17BCE0581
SATYAM SINGH CHAUHAN

OPERATING SYSTEMS
CSE2005

AIM:
Memory Management

Algorithm:

First Fit
1- Input memory blocks with size and processes with size.
2- Initialize all memory blocks as free.
3- Start by picking each process and check if it can
   be assigned to current block.
4- If size-of-process <= size-of-block if yes then
   assign and check for next process.
5- If not then keep checking the further blocks.

Best Fit
1- Input memory blocks and processes with sizes.
2- Initialize all memory blocks as free.
3- Start by picking each process and find the
   minimum block size that can be assigned to
   current process i.e., find min(bockSize[1],
   blockSize[2],.....blockSize[n]) >
   processSize[current], if found then assign
   it to the current process.
5- If not then leave that process and keep checking
   the further processes.

Worst Fit
1- Input memory blocks and processes with sizes.
2- Initialize all memory blocks as free.
3- Start by picking each process and find the
   maximum block size that can be assigned to
   current process i.e., find max(bockSize[1],
   blockSize[2],.....blockSize[n]) >
   processSize[current], if found then assign
   it to the current process.
5- If not then leave that process and keep checking

the further processes.

CODE

First-fit:

```
#include<stdio.h>
void main()
{ int bsize[10], psize[10], bno, pno, flags[10], allocation[10], i, j;
for(i = 0; i < 10; i++)
{ flags[i] = 0; allocation[i] = -1;
}
printf("Enter no. of blocks: "); scanf("%d", &bno);
printf("\nEnter size of each block: "); for(i = 0; i < bno; i++) scanf("%d",
&bsize[i]);
printf("\nEnter no. of processes: "); scanf("%d", &pno);
printf("\nEnter size of each process: "); for(i = 0; i < pno; i++) scanf("%d",
&psize[i]); for(i = 0; i < pno; i++) //allocation as per first fit for(j = 0; j <
bno; j++) if(flags[j] == 0 && bsize[j] >= psize[i])
{ allocation[j] = i; flags[j] = 1; break;
}
//display allocation details printf("\nBlock no.\tsize\t\tprocess no.\t\tsize");
for(i = 0; i < bno; i++)
{
printf("\n%d\t\t%d\t\t", i+1, bsize[i]); if(flags[i] == 1)
printf("%d\t\t\t%d",allocation[i]+1,psize[allocation[i]]); else printf("Not
allocated");
}
```

OUTPUT

```
$ vi firstfit.c
$ cc firstfit.c
$ ./a.out
```

```
Enter size of each block: 100
500
200
300
600

Enter no. of processes: 4

Enter size of each process: 212
417
112
426

Block no.       size            process no.             size
1               100             Not allocated
2               500             1                       212
3               200             3                       112
4               300             Not allocated
```

CODE

Best-fit :

```
#include<stdio.h>
void main()
{
int fragment[20],b[20],p[20],i,j,nb,np,temp,lowest=9999; static int
barray[20],parray[20];
printf("\n\t\t\tMemory Management Scheme - Best Fit"); printf("\nEnter the number
of blocks:"); scanf("%d",&nb); printf("Enter the number of processes:");
scanf("%d",&np);
printf("\nEnter the size of the blocks:-\n"); for(i=1;i<=nb;i++)
{ printf("Block no.%d:",i); scanf("%d",&b[i]);
}
printf("\nEnter the size of the processes :-\n"); for(i=1;i<=np;i++)
{
printf("Process no.%d:",i); scanf("%d",&p[i]);
}
for(i=1;i<=np;i++)
{
for(j=1;j<=nb;j++)
{ if(barray[j]!=1)
{ temp=b[j]-p[i]; if(temp>=0) if(lowest>temp)
{ parray[i]=j; lowest=temp;
}
}
}
fragment[i]=lowest; barray[parray[i]]=1; lowest=10000;
}
printf("\nProcess_no\tProcess_size\tBlock_no\tBlock_size\tFragment");
for(i=1;i<=np && parray[i]!=0;i++)
printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,p[i],parray[i],b[parray[i]],fragment[i]);
}
```

OUTPUT

```
./a.out
Enter the number of blocks:5
Enter the number of processes:4

Enter the size of the blocks:-
Block no.1:100
Block no.2:500
Block no.3:200
Block no.4:300
Block no.5:600

Enter the size of the processes :-
Process no.1:212
Process no.2:417
Process no.3:112
Process no.4:426

Process_no      Process_size    Block_no        Block_size      Fragme
1               212             4               300             88
2               417             2               500             83
3               112             3               200             88
```

CODE

Worst-fit:

```
#include<stdio.h>
int main()
{ int fragments[10], blocks[10], process[10]; int m, n, number_of_blocks,
number_of_files, temp, top = 0; static int block_arr[10], file_arr[10];
printf("\nEnter the Total Number of Blocks:\t"); scanf("%d",&number_of_blocks);
printf("Enter the Total Number of process:\t"); scanf("%d",&number_of_files);
printf("\nEnter the Size of the Blocks:\n"); for(m = 0; m < number_of_blocks;
m++)
{
printf("Block No.[%d]:\t", m + 1); scanf("%d", &blocks[m]);
}
printf("Enter the Size of the process:\n"); for(m = 0; m < number_of_files; m++)
{
printf("process No.[%d]:\t", m + 1); scanf("%d", &process[m]);
}
for(m = 0; m < number_of_files; m++)
{
for(n = 0; n < number_of_blocks; n++)
{
if(block_arr[n] != 1)
{
temp = blocks[n] - process[m]; if(temp >= 0)
{ if(top < temp)
{ file_arr[m] = n; top = temp;
```

```
}
}
}
fragments[m] = top; block_arr[file_arr[m]] = 1; top = 0;
}
}
printf("\nFile Number\tFile Size\tBlock Number\tBlock Size\tFragment"); for(m =
0; m < number_of_files; m++)
{
printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d", m, process[m], file_arr[m],
blocks[file_arr[m]], fragments[m]);
} printf("\n"); return 0;
}
```

OUTPUT

```
cc worst-fit.c
./a.out
Enter the Total Number of Blocks:       5
Enter the Total Number of process:      4

Enter the Size of the Blocks:
Block No.[1]:   100
Block No.[2]:   500
Block No.[3]:   200
Block No.[4]:   300
Block No.[5]:   600
Enter the Size of the process:
process No.[1]: 212
process No.[2]: 417
process No.[3]: 112
process No.[4]: 426

File Number     File Size       Block Number    Block Size      Fragment
0               212             4               600             388
1               417             0               100             0
2               112             2               200             0
3               426             0               100             0
```