**6. Write a C program to simulate the following contiguous memory allocation techniques**

**a) Worst-fit b) Best-fit c) First-fit**

**DESCRIPTION**

One of the simplest methods for memory allocation is to divide memory into several fixed-sized partitions. Each partition may contain exactly one process. In this multiple-partition method, when a partitionis free, a process is selected from the input queue and is loaded into the free partition. When the process terminates, the partition becomes available for another process. The operating system keeps a table indicating which parts of memory are available and which are occupied. Finally,when a process arrives and needs memory,a memory section large enough for this process isprovided. When it is time to load or swap a process into main memory, and if there is more than onefree block of memory of sufficient size, then the operating system must decide which free block to allocate. Best-fit strategy chooses the block that is closest in size to the request.First-fit chooses the first available block that is large enough. Worst-fit chooses the largest available block.

**PROGRAM**

***WORST-FIT***

#include<stdio.h>

//#include<conio.h>

#define max 25

void main()

{

int frag[max], b[max], f[max], bf[max], ff[max], i, j, nb, nf, temp;

// Initializing arrays to zero

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

frag[i] = 0;

bf[i] = 0;

ff[i] = 0;

}

//clrscr();

printf("\n\tMemory Management Scheme - Worst Fit");

printf("\nEnter the number of blocks:");

scanf("%d", &nb);

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

scanf("%d", &nf);

printf("\nEnter the size of the blocks:-\n");

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

{

printf("Block %d:", i);

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

}

printf("Enter the size of the files :-\n");

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

{

printf("File %d:", i);

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

}

for(i = 1; i <= nf; i++) {

for(j = 1; j <= nb; j++) {

if(bf[j] != 1) {

temp = b[j] - f[i];

if(temp >= 0) {

ff[i] = j;

break;

}

}

}

frag[i] = b[ff[i]] - f[i]; // Calculate fragmentation correctly

bf[ff[i]] = 1;

}

printf("\nFile\_no:\tFile\_size:\tBlock\_no:\tBlock\_size:\tFragement");

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

{

printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d", i, f[i], ff[i], b[ff[i]], frag[i]);

}

//getch();

}

**INPUT**

Memory Management Scheme - Worst Fit

Enter the number of blocks: 3

Enter the number of files: 2

Enter the size of the blocks:- Block 1: 5

Block 2: 2

Block 3: 7

Enter the size of the files:- File 1: 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| File 2: 4  **OUTPUT** | | | | |
| File No | File Size | Block No | Block Size | Fragment |
| 1 | 1 | 1 | 5 | 4 |
| 2 | 4 | 3 | 7 | 3 |

**BEST-FIT**

#include<stdio.h>

//#include<conio.h>

#define max 25

void main()

{

int frag[max], b[max], f[max], i, j, nb, nf, temp, lowest = 10000;

static int bf[max], ff[max];

//clrscr();

printf("\n\tMemory Management Scheme – Best Fit");

printf("\nEnter the number of blocks:");

scanf("%d", &nb);

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

scanf("%d", &nf);

printf("\nEnter the size of the blocks:-\n");

for (i = 1; i <= nb; i++) {

printf("Block %d:", i);

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

}

printf("Enter the size of the files :-\n");

for (i = 1; i <= nf; i++) {

printf("File %d:", i);

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

}

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

{

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

{

if(bf[j]!=1)

{

temp=b[j]-f[i];

if(temp>=0)

if(lowest>temp)

{

ff[i]=j;

lowest=temp;

}

}

}

frag[i]=lowest; bf[ff[i]]=1; lowest=10000;

}

printf("\nFile No\tFile Size \tBlock No\tBlockSize\tFragment");

for(i=1;i<=nf && ff[i]!=0;i++)

printf("\n%d\t%d\t\t%d\t\t%d\t\t%d",i,f[i],ff[i],b[ff[i]],frag[i]);

//getch();

}

***INPUT***

Enter the number of blocks: 3

Enter the number of files: 2

Enter the size of the blocks:- Block 1: 5

Block 2: 2

Block 3: 7

Enter the size of the files:- File 1: 1

File 2: 4

|  |  |  |  |
| --- | --- | --- | --- |
| ***OUTPUT***  File No  Size | Block No | Block Size | Fragment |
| 1 1 | 2 | 2 | 1 |
| 2 4 | 1 | 5 | 1 |

***FIRST-FIT***

#include<stdio.h>

//#include<conio.h>

#define max 25

void main()

{

int frag[max], b[max], f[max], i, j, nb, nf, temp;

static int bf[max], ff[max];

//clrscr();

printf("\n\tMemory Management Scheme - First Fit");

printf("\nEnter the number of blocks:");

scanf("%d", &nb);

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

scanf("%d", &nf);

printf("\nEnter the size of the blocks:-\n");

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

{

printf("Block %d:", i);

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

}

printf("Enter the size of the files :-\n");

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

{

printf("File %d:", i);

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

}

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

{

temp = f[i]; // Size of the current file

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

{

if(bf[j] != 1 && b[j] >= temp) // If bf[j] is not allocated and block size is sufficient

{

ff[i] = j;

frag[i] = b[j] - temp;

bf[j] = 1; // Mark the block as allocated

break; // Break out of the loop once the file is allocated

}

}

}

printf("\nFile\_no:\tFile\_size:\tBlock\_no:\tBlock\_size:\tFragement");

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

{

printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d", i, f[i], ff[i], b[ff[i]], frag[i]);

}}

Memory Management Scheme - First Fit

Enter the number of blocks:3

Enter the number of files:2

Enter the size of the blocks:-

Block 1:5

Block 2:2

Block 3:7

Enter the size of the files :-

File 1:1

File 2:4

File\_no: File\_size: Block\_no: Block\_size: Fragement

1 1 1 5 4

2 4 3 7 3