LAB 4: FITTING STRATIGIES IN MEMORY MANAGEMENT

OBJECTIVE:

The primary objective of this lab report is to explore and evaluate various fitting strategies utilized in memory management. Memory management is a crucial aspect of computer systems, especially in operating systems, where efficient allocation and deallocation of memory are essential for optimal system performance.

THEORY:

Memory management in computer systems involves allocating memory segments to processes or data structures dynamically. Fitting strategies dictate how memory segments are chosen from the available free memory space for allocation. There are several common fitting strategies, including First Fit, Next Fit, Best Fit, and Worst Fit.

1. First Fit:
   * In First Fit, the system searches for the first available memory segment that is large enough to accommodate the requested memory size.
   * It is simple to implement but can lead to fragmentation, as it may allocate larger segments than necessary, leaving smaller unused spaces.
2. Next Fit:
   * Next Fit is similar to First Fit, but it starts searching for free memory space from the location where the last allocation occurred, rather than the beginning.
   * While it may reduce fragmentation compared to First Fit, it still suffers from potential fragmentation issues.
3. Best Fit:
   * Best Fit aims to find the smallest available memory segment that is large enough to accommodate the requested size.
   * It minimizes wasted memory but can be inefficient due to the time required for searching the entire free memory list for the best fit.
4. Worst Fit:
   * Worst Fit allocates memory from the largest available free memory segment.
   * It tends to leave behind larger fragmented areas, leading to potential fragmentation issues in the long run.

Question 1: First Fit Strategy

Solution:

#include<stdio.h>

#define max 25

void main(){

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

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

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++){

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

if (bf[j] != 1){

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

if (temp >= 0){

ff[i] = j;

break;}}}

frag[i] = temp;

bf[ff[i]] = 1;}

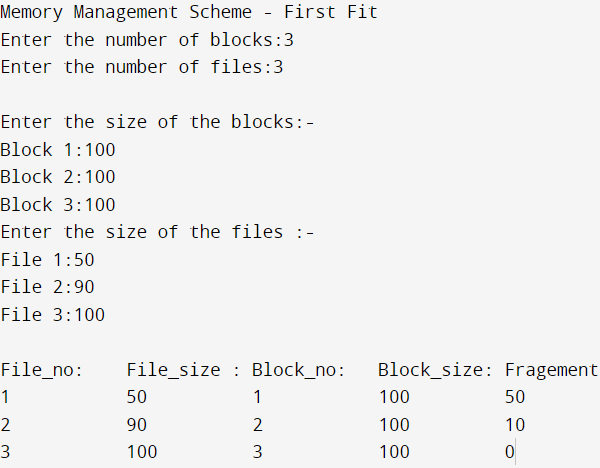
printf ("\nFile\_no:\tFile\_size :\tBlock\_no:\tBlock\_size:\tFragement");

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]);}

Output:



Question 2: Best fit Strategy

Solution:

#include<stdio.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];

printf("\n ----- Best Fit Strategy ---- \n");

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;}}}

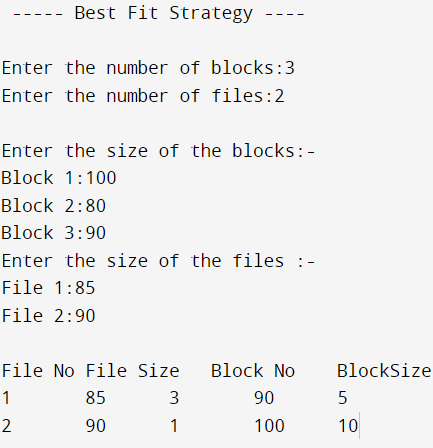
frag[i]=lowest; bf[ff[i]]=1; lowest=10000;}

printf("\nFile No\tFile Size \tBlock No\tBlockSize\tFragment"); for(i=1;i<=nf

&& ff[i]!=0;i++)

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

Output:



Question 3: Worst Fit Strategy

Solution:

#include<stdio.h>

#define max 25

void main (){

int frag[max], b[max], f[max], i, j, nb, nf, temp, highest = 0;

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

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)

if (highest < temp){

ff[i] = j;

highest = temp;}}}

frag[i] = highest;

bf[ff[i]] = 1;

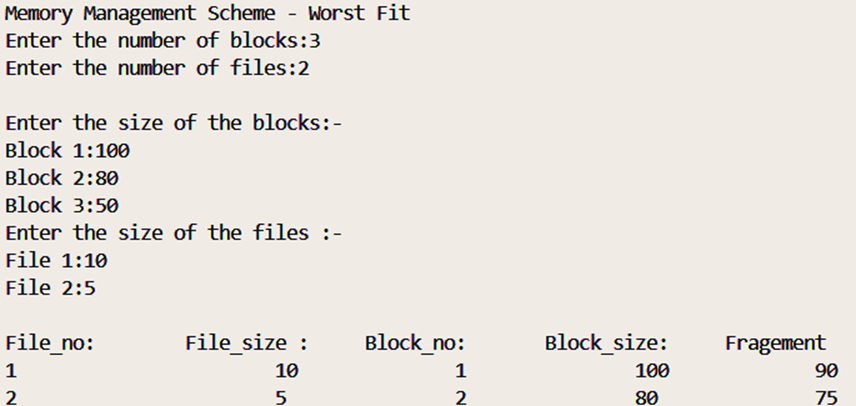
highest = 0;}

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]);}

Output:



CONCLUSION:

In conclusion, through this lab report, we have gained insights into the principles and characteristics of various fitting strategies, including First Fit, Next Fit, Best Fit, and Worst Fit. Each strategy has its advantages and disadvantages, depending on factors such as system workload, memory allocation patterns, and fragmentation concerns.