Deepesh Tank
1807517

Experiment-8

File Allocation strategies.

Aim:- write a C program to stimulate the file allocation strategies
a) sequential
b) Indexed
c) Linked.

Theory:-

a) sequential:-

In this allocation strategy each file occupies set of contiguously blocks on the disk. This strategy is the best suited for the sequential files, the file allocation table consists of a single entry for each file. It shows the filename, starting block of the file and size of the file. The main problem with this strategy is, it is difficult to find the contiguous free blocks in the disk and some free blocks could happen between two files.

Algorithm:-

Step 1 → Start the program.
Step 2 → Get the number of memory partitions and their sizes
Step 3:- Get the number of processes and value of block size for each process
Step 4 → First fit algorithm searches all the entire memory block until a hole which is big enough is encountered. It allocates that memory block for the requesting processes
Step 5 :- Best-fit algorithm searches the memory blocks for smallest hole which can be allocated to requesting processes and allocates it.

Step 6:- Worst-fit algorithm searches the memory blocks for the largest and allocate it to the process

Step 7:- Analysis all the 3 memory managment technique and display best algorithm which utilize the memory resources effectively and efficiently

Step 8 → Stop the program.

b) Indexed:

Indexed allocations support both sequential and direct access file The file indexes are not physically stored as apart of the file allocation table. Whenever the file size ~~decrease~~ increases, we can easily add more blocks to the index. In this strategy, the file allocation table contains a single entry for each file. The entry consisting of one index block, the index blocks having the pointers to the other blocks No external fragmentation.

Algorithm:-

Step 1:- Start

Step 2:- let n be size of buffer.

Step 3:- check if there are any producer

Step 4:- if yes check whether the buffer is full.

Step 5:- if no the producer item is stored in the buffer.

Step 6:- if the buffer is full the producer has to wait.

Step 7:- check there is any consumer. if yes. check whether buffer is empty

Step 8:- if no the consumer consumes them from the buffer.

Step 9:- if the buffer is empty, the consumer has to wait.

Step 10:- Repeat checking for producer and consumer till required.

Step 11:- Terminate the process

9 Linked:-

in this file allocation strategy, file allocation table contains a field which points to starting block of memory. for each block a pointer a pointer is kept to next successive block. there is no external fragmentation

Algorithm:-

step1:- start, get no of files;

step2:- get memory requirements of each file

step3→ allocate the required location by selecting a location randomly q = random(100);

a) check whether the selected location is free.

b) if location is free allocate and set flag = 1 to allocated location

step 4 - print result file number, length, block allocated

step 5 - stop the program.

Program code :-

a) Sequential :-

```
#include<stdio.h>
#include<conio.h>
void main()
{ int f[50], i, st, len, j, c, k, count=0;
clrscr();
for(i=0;i<50;i++) f[i]=0;
printf("File Allocated are : \n");
x: count=0;
printf("file allocated are : \n");
printf("Enter starting block and length of files:");
scanf("%d %d",&st,&len);
for(k=st;k<(st+len);k++){
if(f[k]==0) count++;}
if(len==count)
{ for(j=st;j<(st+len);j++)
if(f[j]==0)
{ f[j]=1;
printf("%d \t %d \n",j,f[j]);}
if(j!=(st+len-1))
printf("The file is allocated to disk \n"); }
else
printf("The file is not allocated \n");
printf("Do you want to enter more file(Yes-1/No-0)");
scanf("%d",&c);
if(c==1)
goto x;
else
exit();
getch();}
```

Program output :-

File Allocated are

Enter starting block and length of files : 14 3

14 1

15 1

16 1

The file is allocated to disk

Do you want to enter more file (Yes-1/No-0) 1

Enter starting block and length of file : 14 1

The file is not allocated

Do you want to enter more file (Yes-1/No-0) 1

Enter starting block and length of file :- 14 4

The file is not allocated

Do you want to enter more file (Yes-1/No-0) 0

b) Indexed

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
void main()
{ int f[50], index[50], i, n, st, len, j, c, k, ind, count=0;
clrscr();
for(i=0; i<50; i++){
f[i]=0; }
x: printf("Enter the index block: ");
scanf("%d", &ind);
if (f[ind]!=1)
{ printf("Enter no of blocks needed and no of file for index
%d on the disk: \n", ind);
scanf("%d", &n); }
else { printf("%d index is already allocated \n", ind);
goto x;
}
y: count=0;
for(i=0; i<n; i++)
{ scanf("%d", &index[i]);
if (f[index[i]]==0)
count++; }
if (count==n) {
for (j=0; j<n; j++)
f[index[j]]=1;
printf("Allocated \n");
printf("File indexed \n");
for (k=0; k<n; k++)
printf("%d ---> %d: %d\n", ind, index[k], f[index[k]]);
}
```

```
else
{ printf("file in the index is already allocated \n");
prmtf("Enter another file indexed");
goto y; }
prmtf("Do you want to enter more file (Yes-1/No-0));
sconf("%d", &c);
if(c==1)
goto x;
else
exit(0);
getch(0);
}
```

Program output:-

Enter the indexblock: 5

Enter no of blocks needed and no of files for the index 5 on the disk:
4

1 2 3 4

Allocated

File indexed

5 - - - - 71: 1

5 - - - - 72: 1

5 - - - - 73: 1

5 - - - - 74: 1

Do you want to enter more file (Yes-1/No-0)

0

c) linked :-

```
#include<stdio.h>
main()
{
int f[50],p,i,j,k,a,st,len,n,c;
clrscr();
for(i=0;i<50;i++)f[i]=0;
printf("Enter how many blocks that are already allocated");
scanf("%d",&p);
printf("\nEnter the blocks nos that are already allocated");
for(i=0;i<p;i++){
scanf("%d",&a); f[a]=1;}
x:printf("Enter the starting index block and length");
scanf("%d %d",&st,&len);
k=len;
for(j=st;j<(k+st);j++){
if(f[j]==0) { f[j]=1;
printf("\n%d->%d",j,f[j]);}
else {
printf("\n%d -> file is already allocated",j);
k++;}
printf("\n if u want to enter one more file {yes-1/No-0)");
scanf("%d",&c);
if(c==1)
goto x;
else
exit();
getch();
}
```

program output:-

Enter how many blocks that are already allocated 3

Enter the blocks nos that are already allocated 4 7 9.

Enter the starting index block & length 3.

7

3->1

4->1 file is already allocated.

5->1

6->1

7->1 file is already allocated

8->1

9->1 file is already allocated

10->1

11->1

12->1.