**OPERATING SYSTEMS**

LAB EXPERIMENT - 4

Submitted To: Prashant Giridhar Shambharkar Sir.

Submitted By: Manav Agrawal (2K19/CO/216)

---------------------------------------------------------------------

Aim:

Write a C program to simulate all file allocation strategies for a) Sequential,

b) Indexed.

Introduction:

### File Allocation Methods:

The allocation methods define how the files are stored in the disk blocks. We must select the best method for the file allocation because it will directly affect the system performance and system efficiency. With the help of the allocation method, we can utilize the disk, and also files can be accessed.The main idea behind these methods is to provide efficient disk space utilization and fast access to the file blocks.

There are three main disk space or file allocation methods.

1. Sequential Allocation
2. Linked Allocation
3. Indexed Allocation
4. **Sequential Allocation:**

In Sequential Allocation, each file occupies a contiguous set of blocks on the disk. For example, if a file requires n blocks and is given a block b as the starting location, then the blocks assigned to the file will be: b, b+1, b+2,……b+n-1. This means that given the starting block address and the length of the file (in terms of blocks required), we can determine the blocks occupied by the file. The directory entry for a file with contiguous allocation contains the address of the starting block and length of the allocated portion.

1. **Linked Allocation:**

In the linked list allocation method, it overcomes the drawbacks of the contiguous allocation method. In this file allocation method, each file is treated as a linked list of disks blocks. In the linked list allocation method, it is not required that disk blocks assigned to a specific file are in the contiguous order on the disk. The directory entry comprises a pointer for starting file block and also for the ending file block. Each disk block that is allocated or assigned to a file consists of a pointer, and that pointer points to the next block of the disk, which is allocated to the same file.

1. **Indexed Allocation:**

The Indexed allocation method is another method that is used for file allocation. In the index allocation method, we have an additional block, and that block is known as the index block. For each file, there is an individual index block. In the index block, the ith entry holds the disk address of the ith file block. We can see in the below figure that the directory entry comprises the address of the index block.

We will be simulating two of these algorithms which are Sequential Allocation and Indexed Allocation.

Algorithms:

1. **Sequential Allocation:**

1- Start the program.

2- Gather information about the number of files.

3- Gather the memory requirement of each file.

4- Allocate the memory to the file in a sequential manner.

5- Select any random location from the available location.

6- Check if the location that is selected is free or not.

7- If the location is allocated set the flag = 1.

8- Print the file number, length, and the block allocated.

9- Gather information if more files have to be stored.

10- If yes, then go to point 2.

11- If no, Stop the program.

1. **Indexed Allocation:**

1- Start the program.

2- Get information about the number of files.

3- Get the memory requirement of each file.

4- Allocate the memory to the file by selecting random locations.

5- Check if the location that is selected is free or not.

6- If the location is allocated set the flag = 1, and if free set flag = 0.

7- Print the file number, length, and the block allocated.

8- Gather information if more files have to be stored.

9- If yes, then go to point 2.

10- If no, Stop the program.

Implementation:

**A) Sequential Allocation:**

#include <iostream>

#include <conio.h>

#include <stdlib.h>

using namespace std;

void recurse(int files[]){

int flag = 0, startBlock, len, k;

cout << "Enter the starting block and the length of the files: ";

cin >> startBlock >> len;

for (int j=startBlock; j<(startBlock+len); j++){

if (files[j] == 0)

flag++;

}

if(len == flag){

for (int k=startBlock; k<(startBlock+len); k++){

if (files[k] == 0){

files[k] = 1;

cout << k <<"\t" << files[k] << endl;

}

}

if (k != (startBlock+len-1))

cout << "The file is allocated to the disk" << endl;

}

else

cout << "The file is not allocated to the disk" << endl;

cout << "Do you want to enter more files?" << endl;

int ch;

cout << "Press 1 for YES, 0 for NO: ";

cin >> ch;

if (ch == 1)

recurse(files);

else

exit(0);

return;

}

int main()

{

int files[50];

for(int i=0;i<50;i++)

files[i]=0;

recurse(files);

getch();

return 0;

}

**B) Indexed Allocation:**

#include <iostream>

#include <conio.h>

#include <stdlib.h>

using namespace std;

int files[50], indexBlock[50], indBlock, n;

void recurse1();

void recurse2();

void recurse1(){

cout << "Enter the index block: ";

cin >> indBlock;

if (files[indBlock] != 1){

cout << "Enter the number of blocks and the number of files needed for the index " << indBlock << " on the disk: ";

cin >> n;

}

else{

cout << indBlock << " is already allocated" << endl;

recurse1();

}

recurse2();

}

void recurse2(){

int flag = 0;

for (int i=0; i<n; i++){

cin >> indexBlock[i];

if (files[indexBlock[i]] == 0)

flag++;

}

if (flag == n){

for (int j=0; j<n; j++){

files[indexBlock[j]] = 1;

}

cout << "Allocated" << endl;

cout << "File Indexed" << endl;

for (int k=0; k<n; k++){

cout << indBlock << " ------> " << indexBlock[k] << ": " << files[indexBlock[k]] << endl;

}

}

else{

cout << "File in the index is already allocated" << endl;

cout << "Enter another indexed file" << endl;

recurse2();

}

cout << "Do you want to enter more files?" << endl;

cout << "Enter 1 for Yes, Enter 0 for No: ";

int ch;

cin >> ch;

if (ch == 1)

recurse1();

else

exit(0);

return;

}

int main()

{

for(int i=0;i<50;i++)

files[i]=0;

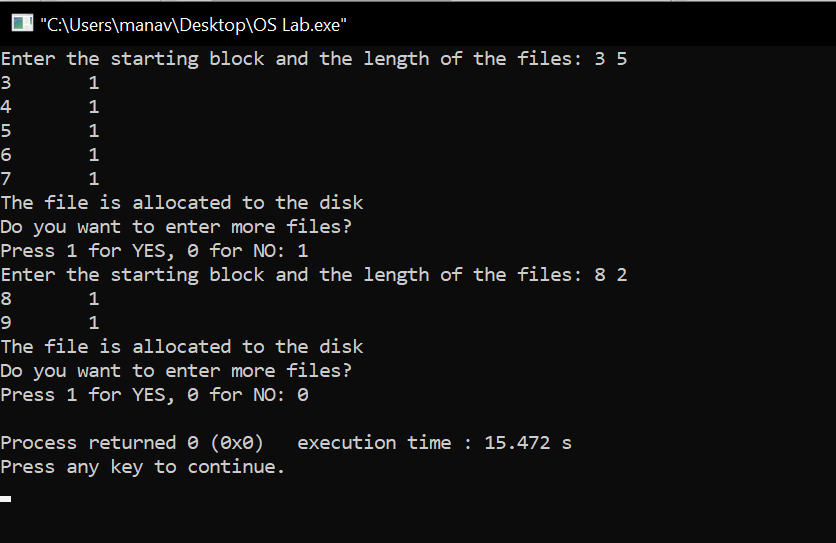
recurse1();

return 0;

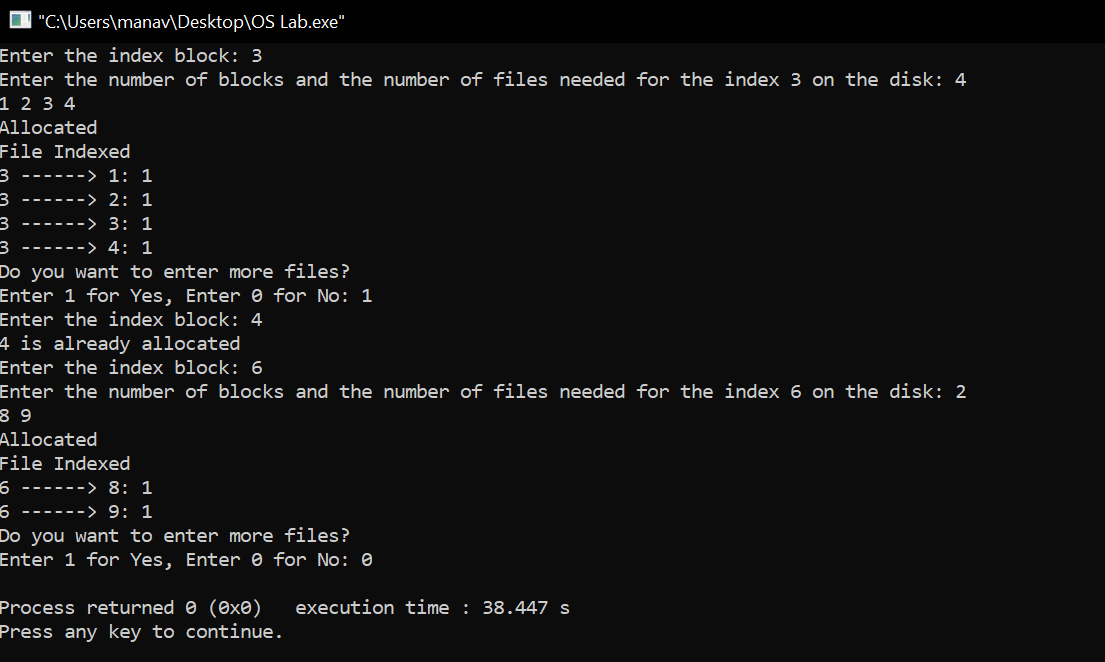
}

Output:

**A) Sequential Allocation:**

****

**B) Indexed Allocation:**

****

Learning From The Experiment:

The algorithms are full of pros and also cons for some cases (given below) which we learned from this experiment;

The Sequential Allocation supports both the Sequential and Direct Accesses. For direct access, the address of the kth block of the file which starts at block b can easily be obtained as (b+k).This is extremely fast since the number of seeks are minimal because of contiguous allocation of file blocks. However, this method suffers from both internal and external fragmentation. This makes it inefficient in terms of memory utilization. Increasing file size is difficult because it depends on the availability of contiguous memory at a particular instance.

The Indexed Allocation supports direct access to the blocks occupied by the file and therefore provides fast access to the file blocks.It overcomes the problem of external fragmentation. Still, The pointer overhead for indexed allocation is greater than linked allocation.For very small files, say files that expand only 2-3 blocks, the indexed allocation would keep one entire block (index block) for the pointers which is inefficient in terms of memory utilization. However, in linked allocation we lose the space of only 1 pointer per block.

***THANK YOU!***