Experiment no. 10

Write a program to demonstrate various file allocation methods such as Contiguous allocation and Indexed Allocation

By

Harshali Bhuwad

Contiguous File Allocation Program

- In the contiguous File Allocation, the file is stored in sequential memory blocks and they are next to each other.

- This access method also allows us to directly access the blocks of the memory as we can calculate easily where our required information is located.

Algorithm:

STEP 1: Start the program.

STEP 2: Gather information about the number of files.

STEP 3: Gather the memory requirement of each file.

STEP 4: Allocate the memory to the file in a sequential manner.

STEP 5: Select any random location from the available location.

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

STEP 7: If the location is allocated set the flag = 1.

STEP 8: Print the file number, length, and the block allocated.

STEP 9: Gather information if more files have to be stored.

STEP 10: If yes, then go to STEP 2.

STEP 11: If no, Stop the program.

```c
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
void recurse(int files[]){
int flag = 0, startBlock, len, j, k, ch;
printf("Enter the starting block and the length of the files: ");
scanf("%d%d", &startBlock, &len);
for (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;
printf("%d\t%d\n", k, files[k]);
}
}
if (k != (startBlock+len-1))
printf("The file is allocated to the disk\n");
}
else
printf("The file is not allocated to the disk\n");
printf("Do you want to enter more files?\n");
printf("Press 1 for YES, 0 for NO: ");
scanf("%d", &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;
printf("Files Allocated are :\n");
recurse(files);
getch();
return 0;
}
```

Indexed File Allocation method

- The Indexed File Allocation stores the file in the blocks of memory, each block of memory has an address and the address of every file block is maintained in a separate index block.

- These index blocks point the file allocation system to the memory blocks which actually contains the file.

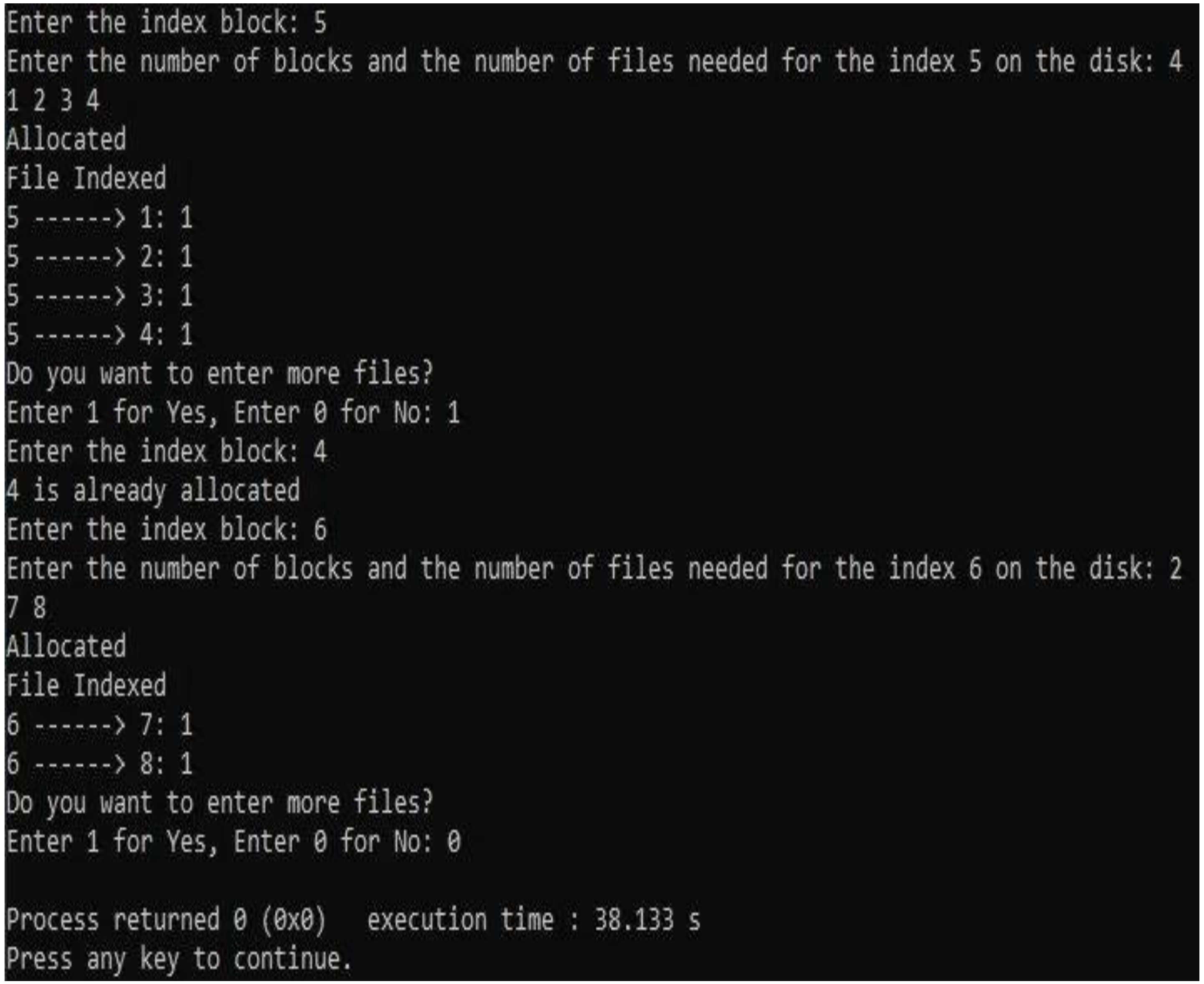
Algorithm:

```
STEP 1: Start the program.
STEP 2: Get information about the number of files.
STEP 3: Get the memory requirement of each file.
STEP 4: Allocate the memory to the file by selecting random locations.
STEP 5: Check if the location that is selected is free or not.
STEP 6: If the location is allocated set the flag = 1, and if free set flag = 0.
STEP 7: Print the file number, length, and the block allocated.
STEP 8: Gather information if more files have to be stored.
STEP 9: If yes, then go to STEP 2.
STEP 10: If no, Stop the program.
```

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
int files[50], indexBlock[50], indBlock, n;
void recurse1();
void recurse2();
void recurse1(){
printf("Enter the index block: ");
scanf("%d", &indBlock);
if (files[indBlock] != 1){
printf("Enter the number of blocks and the number of files
needed for the index %d on the disk: ", indBlock);
scanf("%d", &n);
}
else{
printf("%d is already allocated\n", indBlock);
recurse1();
}
recurse2();
}
```

```
void recurse2(){
int ch;
int flag = 0;
for (int i=0; i<n; i++){
scanf("%d", &indexBlock[i]);
if (files[indexBlock[i]] == 0)
flag++;
}
if (flag == n){
for (int j=0; j<n; j++){
files[indexBlock[j]] = 1;
}
printf("Allocated\n");
printf("File Indexed\n");
for (int k=0; k<n; k++){
printf("%d ------> %d : %d\n", indBlock, indexBlock[k],
files[indexBlock[k]]);
}
}
```

```c
else{
printf("File in the index is already allocated\n");
printf("Enter another indexed file\n");
recurse2();
}
printf("Do you want to enter more files?\n");
printf("Enter 1 for Yes, Enter 0 for No: ");
scanf("%d", &ch);
if (ch == 1)
recurse1();
else
exit(0);
return;
}
int main()
{
for(int i=0;i<50;i++)
files[i]=0;
recurse1();
return 0;
}
```



```
Enter the index block: 5
Enter the number of blocks and the number of files needed for the index 5 on the disk: 4
1 2 3 4
Allocated
File Indexed
5 ------> 1: 1
5 ------> 2: 1
5 ------> 3: 1
5 ------> 4: 1
Do you want to enter more files?
Enter 1 for Yes, Enter 0 for No: 1
Enter the index block: 4
4 is already allocated
Enter the index block: 6
Enter the number of blocks and the number of files needed for the index 6 on the disk: 2
7 8
Allocated
File Indexed
6 ------> 7: 1
6 ------> 8: 1
Do you want to enter more files?
Enter 1 for Yes, Enter 0 for No: 0

Process returned 0 (0x0)   execution time : 38.133 s
Press any key to continue.
```