Practical No. - 8

Aim: Write a program to simulate the following file allocation strategies:

(Sequential, Linked , Indexed)

Tools/Software Required: Turbo C++/VS Code/ Dev C++

Theory:

File allocation refers to the way disk blocks are assigned to store files. The Operating System must efficiently manage these blocks to ensure fast access, minimal fragmentation, and flexible file growth. The main file allocation methods are **Sequential**, **Linked**, and **Indexed** allocation.

Sequential File Allocation

In this method, each file occupies a set of consecutive blocks on the disk. Only the starting block and length are stored.

Linked File Allocation

Each file is stored as a linked list of disk blocks, scattered anywhere on the disk. Every block contains a pointer to the next block.

Indexed File Allocation

A special block, called an index block, stores all block numbers of a file. File data blocks can be placed anywhere on the disk.

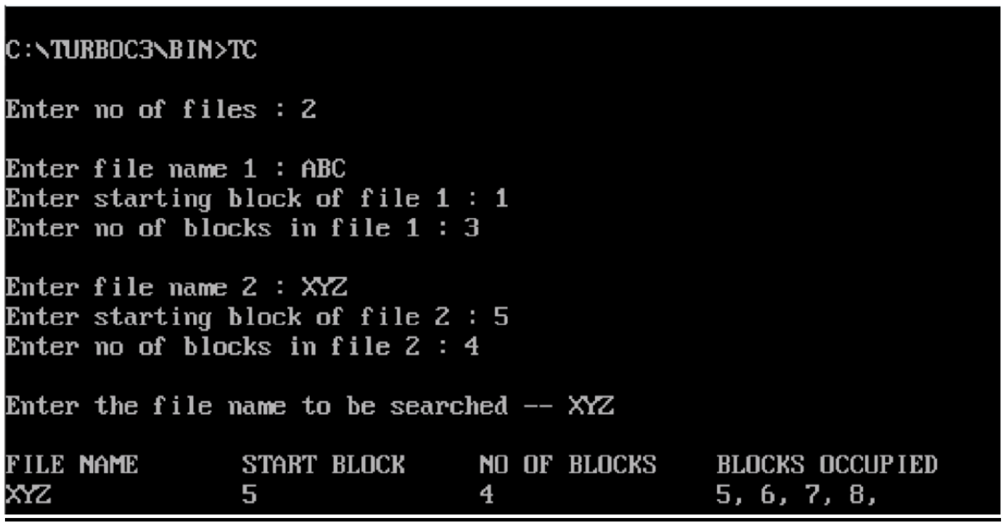
Code :

Sequential Allocation

```
#include <stdio.h>
#include <string.h>

struct fileTable
{
   char name[20];
int sb;     int nob;
} ft[30]; int
main()
{     int i, j, n;     char s[20];
printf("Enter no of files : ");
scanf("%d", &n);
   for (i = 0; i < n; i++)
   {
       printf("\nEnter file name %d : ", i + 1);
scanf("%s", ft[i].name);         printf("Enter starting
block of file %d : ", i + 1);         scanf("%d",
&ft[i].sb);         printf("Enter no of blocks in file %d :
", i + 1);         scanf("%d", &ft[i].nob);
   }
   printf("\nEnter the file name to be searched -- ");
scanf("%s", s);

   for (i = 0; i < n; i++)
   {
       if (strcmp(s, ft[i].name) == 0)
break;
   }     if (i
== n)
   {
       printf("\nFile Not Found");
   }
else
   {
       printf("\nFILE NAME\tSTART BLOCK\tNO OF BLOCKS\tBLOCKS
OCCUPIED\n");
       printf("%s\t\t%d\t\t%d\t\t",
ft[i].name, ft[i].sb, ft[i].nob);         for (j =
0; j < ft[i].nob; j++)
           {
               printf("%d, ", ft[i].sb + j);
```

```
        }
    }
    return 0;
}
```

Output:



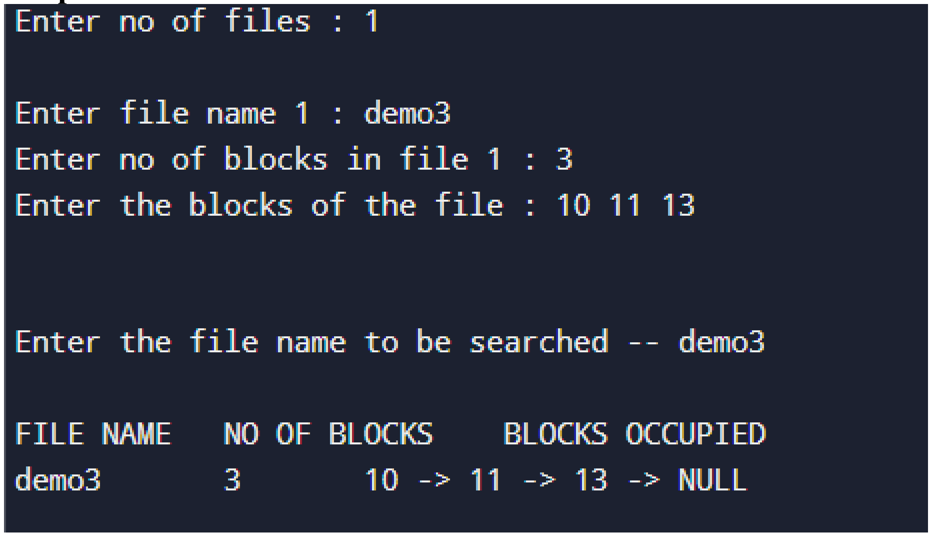
```
C:\TURBOC3\BIN>TC

Enter no of files : 2

Enter file name 1 : ABC
Enter starting block of file 1 : 1
Enter no of blocks in file 1 : 3

Enter file name 2 : XYZ
Enter starting block of file 2 : 5
Enter no of blocks in file 2 : 4

Enter the file name to be searched -- XYZ

FILE NAME       START BLOCK     NO OF BLOCKS    BLOCKS OCCUPIED
XYZ             5               4               5, 6, 7, 8,
```

Linked List Allocation

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct block    //A block has block-level information (e.g., block number, link to next block)
{
   int bno;
   struct block *next;
};

struct fileTable     // A file has file-level information (e.g., name, number of blocks, starting
block)
{
   char name[20];
   int nob;
   struct block *sb; }//sb is a pointer used to store the address of the first block in the linked
list for that file.}
ft[30];
int main()
{
   int i, j, n;
   char s[20];
   struct block *temp; //is used as a temporary pointer to help build and traverse the linked
list of blocks.

   printf("Enter no of files : ");
   scanf("%d", &n);

   for (i = 0; i < n; i++)
   {
      printf("\nEnter file name %d : ", i + 1);
      scanf("%s", ft[i].name);

      printf("Enter no of blocks in file %d : ", i + 1);
      scanf("%d", &ft[i].nob);
// Allocate first block

      ft[i].sb = (struct block *)malloc(sizeof(struct block)); // dynamically allocate memory
for one block
       temp = ft[i].sb;
```

```
        printf("Enter the blocks of the file : ");
        scanf("%d", &temp->bno);
        temp->next = NULL;


    // Create remaining blocks

        for (j = 1; j < ft[i].nob; j++)
        {
            temp->next = (struct block *)malloc(sizeof(struct block));
            temp = temp->next;
            scanf("%d", &temp->bno);
        }
        temp->next = NULL;
    }
    printf("\nEnter the file name to be searched -- ");
    scanf("%s", s);

    // Search for the file
    for (i = 0; i < n; i++)
    {
        if (strcmp(s, ft[i].name) == 0)
            break;
    }
// check if file is found
    if (i == n)
    {
        printf("\nFile Not Found\n");
    }
    else
    {
        printf("\nFILE NAME\tNO OF BLOCKS\tBLOCKS OCCUPIED\n");
        printf("%s\t\t%d\t\t", ft[i].name, ft[i].nob);

        temp = ft[i].sb;

        for (j = 0; j < ft[i].nob; j++)
        {
            printf("%d -> ", temp->bno);
            temp = temp->next;
        }
        printf("NULL");
    }

    return 0;
}
```

Output:

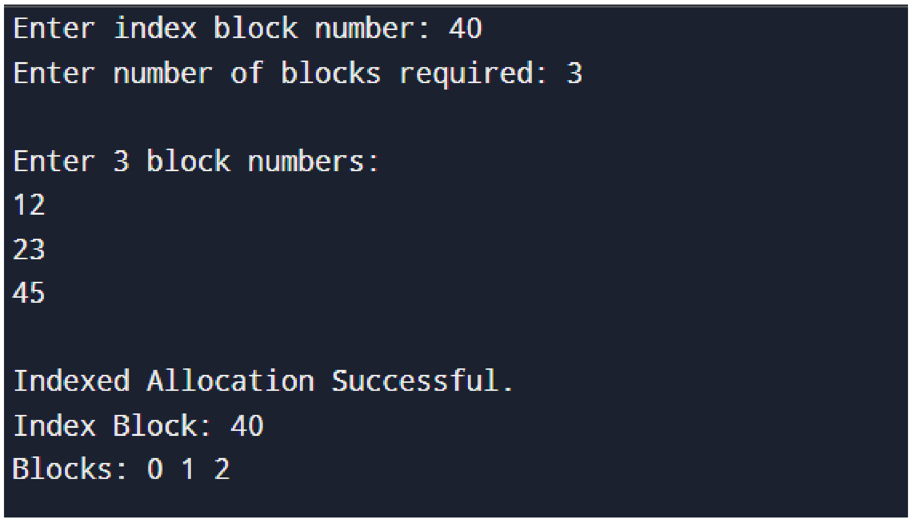


```
Enter no of files : 1

Enter file name 1 : demo3
Enter no of blocks in file 1 : 3
Enter the blocks of the file : 10 11 13



Enter the file name to be searched -- demo3

FILE NAME   NO OF BLOCKS    BLOCKS OCCUPIED
demo3       3       10 -> 11 -> 13 -> NULL
```

Code: **Indexed File Allocation**

```
#include <stdio.h>

int main() {
    int disk[50], indexBlock, n, block, i;
    for (i = 0; i < 50; i++)
        disk[i] = 0;

    printf("Enter index block number: ");
    scanf("%d", &indexBlock);
    if (disk[indexBlock] == 1) { printf("Index block
        already allocated. Try again.\n"); return 0;
    }
    printf("Enter number of blocks required: ");
    scanf("%d", &n);
    printf("\nEnter %d block numbers:\n", n);

    for (i = 0; i < n; i++) {
        scanf("%d", &block);
        if (disk[block] == 0) {
        disk[block] = 1;
        } else { printf("Block %d already allocated. Allocation
           stopped.\n", block); return 0;
        } }
    disk[indexBlock] = 1; printf("\nIndexed Allocation Successful.\nIndex Block:
    %d\nBlocks: ", indexBlock); for (i = 0; i < n; i++) printf("%d ", i);
    printf("\n")
; return 0; }
```

<u>Output:</u>



```
Enter index block number: 40
Enter number of blocks required: 3

Enter 3 block numbers:
12
23
45

Indexed Allocation Successful.
Index Block: 40
Blocks: 0 1 2
```

Course Outcomes:

Understand the concept of file allocation and the importance of organizing disk blocks efficiently in an operating system.

Analyse how different file allocation techniques (Sequential, Linked, Indexed) store files on disk and how these methods affect access time, fragmentation, and file growth.

Apply C programming concepts to implement and simulate Sequential, Linked, and Indexed file allocation strategies using user-defined inputs.

Evaluate the performance and limitations of each allocation method by comparing access methods, storage overhead, and block management.

Create a simulation program that demonstrates the working of different file allocation methods and visualizes how disk blocks are assigned for each technique.

Conclusion:

In this experiment, we successfully simulated Sequential, Linked, and Indexed file allocation methods and understood how operating systems manage disk storage. Each method has its own advantages: Sequential allocation provides fast access, linked allocation allows flexible file growth, and Indexed allocation supports direct access without fragmentation. By implementing these techniques through a program, we learned how disk blocks are assigned, how file access works in each method, and how different allocation strategies affect performance, efficiency, and file management. This practical strengthened our understanding of storage allocation and file system organization in operating systems.

Viva Questions:

Q1. What is file allocation in operating systems?

Ans: File allocation is the process by which an operating system assigns disk blocks to a file. It helps the system keep track of where file data is stored on the disk so that it can be accessed and managed properly.

Q2. Why do we need file allocation methods?

Ans: File allocation methods are required to organize files on the disk efficiently. They help in saving storage space, improving access speed, and allowing easy creation, deletion, and modification of files.

Q3. What is sequential allocation?

Ans: Sequential allocation is a technique in which a file is stored in consecutive disk blocks. The operating system stores the starting address and the length of the file to access its data.

Q4. Mention one advantage and one disadvantage of sequential allocation.

Ans: Advantage: File access is fast because blocks are stored continuously.

Disadvantage: File size cannot be increased easily once allocated.

Q5. What is linked allocation?

Ans: Linked allocation is a file allocation method where a file consists of a list of disk blocks linked together. Each block contains a pointer to the next block, allowing files to be stored in noncontiguous locations.

For Faculty Use only:

Correction Parameters	**Formative Assessment [40%]**	**Timely completion of Practical [40%]**	**Attendance / Learning Attitude [20%]**	**Total**
Marks Obtained				