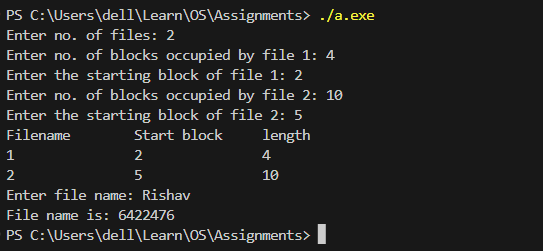
1. **SEQUENTIAL FILE ALLOCATION**

**AIM:** Write a C Program to implement Sequential File Allocation method.



Explanation:

1. The program sequentially allocates blocks to `n` files using the Sequential File Allocation method.

2. It takes the number of files, the number of blocks occupied by each file, and the starting block of each file as input from the user.

3. It then stores the starting block of each file and the sequence of blocks allocated to each file in the array `c`.

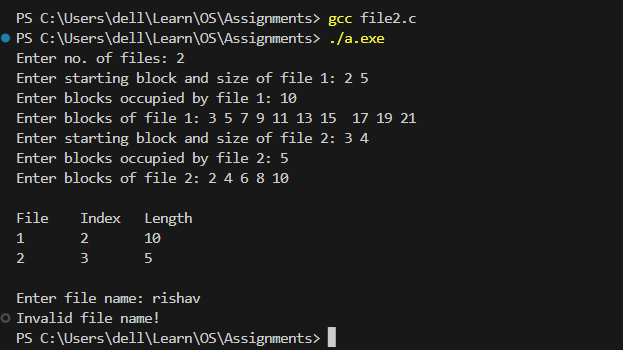
4. After taking the input, it displays a table with the filename, start block, and length for each file.

5. The user is then asked to enter a file name, and the program displays the file's name, length, and the blocks occupied by that file.

Note: The Sequential File Allocation method used in this program allocates blocks sequentially to files without any consideration for optimizing block allocation. In real-world scenarios, more advanced file allocation methods like linked allocation or indexed allocation are used to improve disk space utilization and efficiency. This program is meant for educational purposes to demonstrate the basic concept of Sequential File Allocation.

1. **INDEXED FILE ALLOCATION**

AIM: Write a C Program to implement Indexed File Allocation method.



To fix the code, we need to include the `conio.h` header for `clrscr()` to work. Additionally, it's a good practice to specify the return type of the `main` function as `int`. Here's the corrected code:

Explanation:

1. The program allows the user to input details of multiple files, including starting block, file size, and the blocks occupied by each file.

2. After taking all the inputs, it displays a table with the file number, starting block, and the number of blocks occupied by each file.

3. The user is then prompted to enter a file name, and the program displays the file's index (i.e., the starting block) and the blocks occupied by that file.

Please note that the code assumes that the inputs are valid and doesn't include extensive error handling. In a real-world application, you should include error checking to handle invalid inputs or other exceptional cases.

Enter no. of files: 2

Enter starting block and size of file 1: 2 5

Enter blocks occupied by file 1: 10

Enter blocks of file 1: 3 5 7 9 11 13 15 17 19 21

Enter starting block and size of file 2: 3 4

Enter blocks occupied by file 2: 5

Enter blocks of file 2: 2 4 6 8 10

Enter file name: 2

```

In this example, we have two files:

1. File 1 starts at block 2 and has a size of 5. It occupies 10 blocks starting from blocks 3, 5, 7, 9, 11, 13, 15, 17, 19, and 21.

2. File 2 starts at block 3 and has a size of 4. It occupies 5 blocks starting from blocks 2, 4, 6, 8, and 10.

After taking the input, the program asks the user to enter a file name (in this example, 2). The program then displays the details of File 2, including its index (starting block) and the blocks it occupies.

1. **LINKED FILE ALLOCATION**

AIM: Write a C Program to implement Linked File Allocation method.

ALGORITHM:

Step 1: Create a queue to hold all pages in memory

Step 2: When the page is required replace the page at the head of the queue

Step 3: Now the new page is inserted at the tail of the queue

Step 4: Create a stack

Step 5: When the page fault occurs replace page present at the bottom of the stack

Step 6: Stop the allocation.

### 

CODE AT <https://github.com/rbrox/Operating-Systems/tree/main/file>