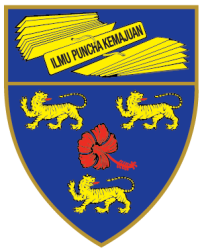


UNIVERSITI MALAYA

WIA2004 Operating Systems

Lab 3: File Allocation Strategies

Group : F5U

Occ: 10

Lecturer: Dr. Fazidah Othman

GROUP MEMBERS	MATRICS NUMBER
ZHANG ZHIYANG	S2193685
MEILIN LI	S2174975
HUMYRA TASMIA	S2176677
HUSNA NADIAH BINTI ROSTHAM	22060027
HUSNA BINTI IHSANUDDIN	U2102305

Table of Content

1.0 Introduction

Sequential file allocation is a fundamental method in file management systems for storing and organizing files on storage devices like disks. Files are assigned consecutive blocks of disk space, with each file occupying contiguous disk blocks. This strategy ensures that file data is stored in physically adjacent blocks, enabling efficient sequential access.

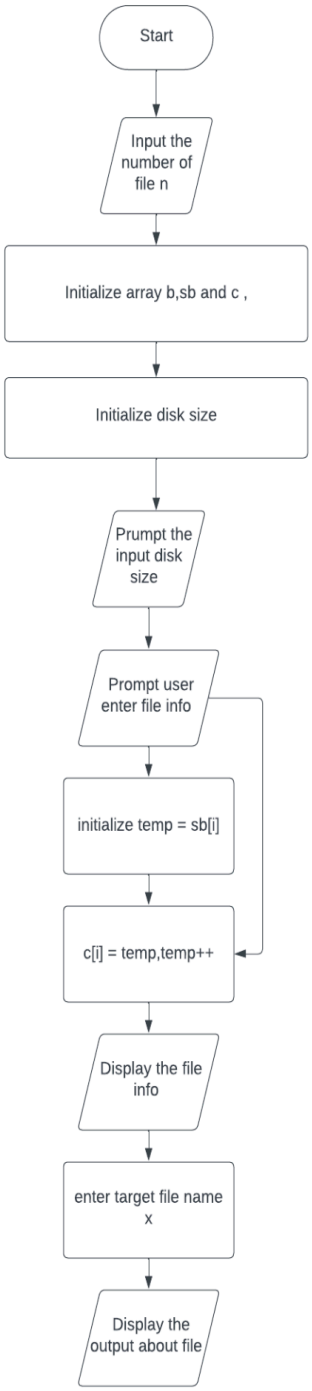
In sequential file allocation:

- Files are stored contiguously on the disk, ensuring that allocated blocks are adjacent to each other.
- An allocation pointer or index is maintained to track the next available block for allocation.
- New files or file extensions allocate consecutive blocks starting from the allocation pointer.
- Deletion of files marks allocated blocks as available, potentially leading to fragmentation.

Advantages include simplicity in implementation and efficient sequential access. However, over time, fragmentation may occur, reducing space utilization and access speed. Additionally, frequent file creation and deletion may lead to inefficient use of fragmented space, limiting flexibility.

2.0 Methodology

Flowchart



Pseudocode

1. 1. Get the number of files (n) from the user

2. Initialize an array b to store the number of blocks occupied by each file

3. Initialize an array sb to store the starting block of each file

4. Initialize a 2D array c to store blocks occupied by each file

5. For each file i from 1 to n:

    5.1 Get the number of blocks occupied by file i (b[i]) from the user
    5.2 Get the starting block of file i (sb[i]) from the user
    5.3 Initialize a temporary variable temp to sb[i]
    5.4 For each block j from 1 to b[i]:
        5.4.1 Add temp to the array c[i]
        5.4.2 Increment temp by 1

6. Print the table header for file information

7. For each file i from 1 to n:
    7.1 Print the filename, starting block, and length of file i

8. Get the filename (x) from the user

9. Print the filename and length

10. Print the blocks occupied by the file

3.0 Implementation

3.1 Coding

```
public class Lab_3 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int n, x;

        System.out.println("Enter the number of files: ");
        n = scanner.nextInt();

        int[] b = new int[n]; // Array to store number of blocks occupied by each file
        int[] sb = new int[n]; // Array to store starting block of each file
        int[][] c = new int[n][]; // 2D array to store blocks occupied by each file

        //if want to limit the size of the disk(array),initialise the variable size
        System.out.println("Enter the disk size:");
        int size = scanner.nextInt();

        for (int i = 0; i < size; i++) {
            System.out.println("Enter the number of blocks occupied by file " + (i + 1) + ": ");
            b[i] = scanner.nextInt();
            System.out.println("Enter the starting block of file " + (i + 1) + ": ");
            sb[i] = scanner.nextInt();
            c[i] = new int[b[i]];
            int temp = sb[i];
            for (int j = 0; j < b[i]; j++) {
                c[i][j] = temp++;
            }
        }
```

```
        System.out.println("Filename\tStarting block\tLength");
        for (int i = 0; i < n; i++) {
            System.out.println((i + 1) + "\t\t" + sb[i] + "\t\t" + b[i]);
        }

        System.out.print("Enter the file name: ");
        x = scanner.nextInt();

        System.out.println("File name is: " + x);
        System.out.println("Length is: " + b[x - 1]);
        System.out.print("Blocks occupied: ");
        for (int i = 0; i < b[x - 1]; i++) {
            System.out.print(c[x - 1][i] + " ");
        }

        scanner.close();
    }

}
```

3.2 Sample output

```
Scanning for projects...

----------------------< com.mycompany:Y2S2_OS >-----------------------
Building Y2S2_OS 1.0-SNAPSHOT
--------------------------------[ jar ]---------------------------------

--- exec-maven-plugin:3.0.0:exec (default-cli) @ Y2S2_OS ---
Enter the number of files:
3
Enter the number of blocks occupied by file 1:
11
Enter the starting block of file 1:
4
Enter the number of blocks occupied by file 2:
22
Enter the starting block of file 2:
5
Enter the number of blocks occupied by file 3:
66
Enter the starting block of file 3:
6
Filename        Starting block  Length
1               4               11
2               5               22
3               6               66
```

4.0 Conclusion

In conclusion, we implemented a program to simulate the sequential file allocation strategy. To access a record, all the previous records must be read sequentially. Each record can only be accessed by reading all the previous records, making it a simple but potentially inefficient method.

Our program first gets the number of files from the user and then initializes arrays to store the number of blocks occupied by each file, the starting block of each file, and the blocks occupied by each file. It then iterates through each file, getting the necessary information from the user and calculating the blocks occupied by each file. Finally, it prints a table showing the file information and allows the user to input a filename to display its details and the blocks it occupies.

5.0 References

- Tanenbaum, A. S., & Bos, H. (2014). Modern operating systems (4th ed.). Pearson.

Chapter 4: File Systems and Directories Modern Operating Systems, Global Edition | Pearson eLibrary

- Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). Operating System Concepts (10th ed.). Wiley.

Chapter 10: File-System Interface
Operating System Concepts 10th : Abraham Silberschatz, Peter B. Galvin, and Greg Gagne : Free Download, Borrow, and Streaming : Internet Archive