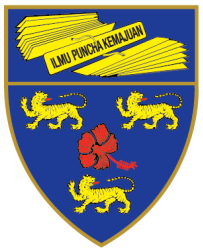


UNIVERSITI MALAYA

WIA2004 Operating Systems

Lab 5: Memory Management Techniques

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

In computer operating systems, memory management refers to the arrangement and distribution of memory resources among processes that are executing simultaneously in a multiprogramming environment. It guarantees that memory is divided across processes in an optimal manner, maximizing allocation while reducing waste and averting fragmentation.

Operating systems assign memory to a process using a technique called first-fit allocation. During First-Fit, the operating system begins its search from the beginning of the list of available memory blocks and continues until it locates a block big enough to satisfy the process's memory need. The operating system divides a block into two halves when it finds one that works: the part that will be assigned to the process and the remaining free block.

Regarding the effectiveness of the First-Fit algorithm, we can summarize that,

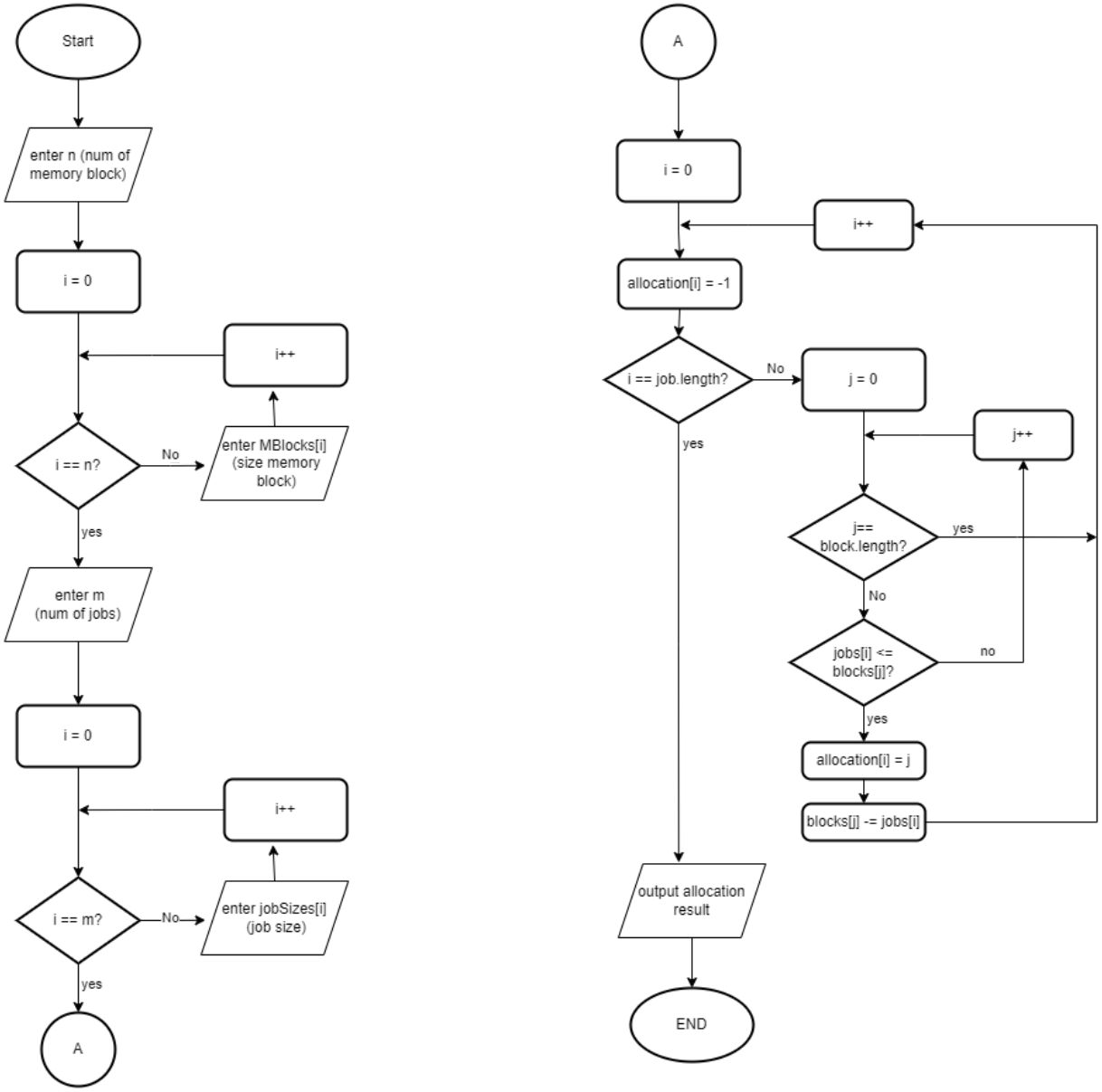
For advantages, First-Fit Allocation includes its simplicity and efficiency, as the search for a suitable block of memory can be performed quickly and easily. Additionally, First-Fit can also help to minimize memory fragmentation, as it tends to allocate memory in larger blocks.

For disadvantages, First-Fit Allocation includes poor performance in situations where the memory is highly fragmented, as the search for a suitable block of memory can become time-consuming and inefficient. Additionally, First-Fit can also lead to poor memory utilization, as it may allocate larger blocks of memory than are actually needed by a process.

Overall, First-Fit Allocation is a commonly used memory allocation strategy in operating systems; yet, its efficacy varies based on the workload and system characteristics.

2.0 Methodology

Flowchart



Pseudocode

1. Input number of memory blocks (n)
2. For each memory block i from 0 to n-1
    2.1 Input sizes of each memory block (MBlocks)
3. Input number of jobs (m)
4. For each job i from 0 to m-1:
    4.1 Input sizes of each job (jobSizes)
5. Initialize allocation array
6. For each job i from 0 to m-1:
    6.1 Set allocation[i] = -1
    6.2 For each memory block j from 0 to n-1:
        6.2.1 If jobSizes[i] <= MBlocks[j]:
            6.2.1.1 Set allocation[i] = j
            6.2.1.2 Reduce MBlocks[j] by jobSizes[i]
            6.2.1.3 Break (Exit inner loop)

7. Output allocation results
    7.1 if allocation[i] = -1, print “not allocated”

3.0 Implementation

3.1 Coding

```java
import java.util.Scanner;

public class L5 {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the number of memory blocks: ");
        int n = sc.nextInt();


        int[] MBlocks = new int[n];
        System.out.println("Enter the size of each memory block:");
        for (int i = 0; i < n; i++) {
            System.out.print("Block " + (i + 1) + " size: ");
            MBlocks[i] = sc.nextInt();
        }


        System.out.println("Enter the number of jobs:");
        int m = sc.nextInt();

        int[] jobSizes = new int[m];
        System.out.println("Enter the size of each job:");
        for (int i = 0; i < m; i++) {
            System.out.print("Job " + (i + 1) + " size: ");
            jobSizes[i] = sc.nextInt();
        }

        int[] allocation = firstFit(jobSizes, MBlocks);

        System.out.println("\nProcess No. \tJob Size \tBlock no.");
        for (int i = 0; i < m; i++) {
            System.out.println((i + 1) + " \t\t" + jobSizes[i] + " \t\t" + (allocation[i] != -1 ? allocation[i] + 1 : "not allocated"));
        }
    }
```

```java
    public static int[] firstFit(int[] jobs, int[] blocks) {
        int[] allocation = new int[jobs.length];
        for (int i = 0; i < jobs.length; i++) {
            allocation[i] = -1;
            for (int j = 0; j < blocks.length; j++) {
                if (jobs[i] <= blocks[j]) {
                    allocation[i] = j;
                    blocks[j] -= jobs[i];
                    break;
                }
            }
        }
        return allocation;
    }
}
```

3.2 Sample output

```
run:
Enter the number of memory blocks:
5
Enter the size of each memory block:
Block 1 size: 7
Block 2 size: 6
Block 3 size: 40
Block 4 size: 58
Block 5 size: 5
Enter the number of jobs:
7
Enter the size of each job:
Job 1 size: 2
Job 2 size: 75
Job 3 size: 20
Job 4 size: 45
Job 5 size: 96
Job 6 size: 12
Job 7 size: 5

Process No.     Job Size        Block no.
1               2               1
2               75              not allocated
3               20              3
4               45              4
5               96              not allocated
6               12              3
7               5               1
BUILD SUCCESSFUL (total time: 31 seconds)
```

4.0 Conclusion

One of the most straightforward and popular methods for allocating memory is First Fit. It splits memory into many fixed-sized segments, with the possibility of just one process running in each partition. The operating system locates a memory region big enough for the process when one arrives and demands memory. First Fit is fast in terms of processing time since it selects the first block that is large enough that is available. External fragmentation, on the other hand, is a condition in which there are tiny free memory blocks dispersed across the memory space that are too small to be utilized for process allocation. In spite of this, First Fit is still a widely utilized memory allocation method due to its ease of use and respectable speed.

5.0 References

- https://www.geeksforgeeks.org/first-fit-allocation-in-operating-systems/
- First Fit algorithm in Memory Management | GeeksforGeeks
- Carter Bays. 1977. A comparison of next-fit, first-fit, and best-fit. Commun. ACM 20, 3 (March 1977), 191–192. https://doi.org/10.1145/359436.359453