Department of
Computer Science and Engineering

# Title: Simulating the MFT and MVT Memory Management Techniques.

Operating System Lab
CSE 310

Green University of Bangladesh

## 1 Objective(s)

- To gather knowledge of different memory management techniques.
- To implement Multiprogramming with a Fixed number of Tasks (MFT) and Multiprogramming with a Variable number of Tasks (MVT).

## 2 Problem analysis

In this section, Multiprogramming with a Fixed number of Tasks (MFT) memory management technique is discussed.

### 2.1 MFT Memory Management Technique

MFT (Multiprogramming with a Fixed number of Tasks) is one of the old memory management techniques in which the memory is partitioned into fixed-size partitions and each job is assigned to a partition. The memory assigned to a partition does not change.

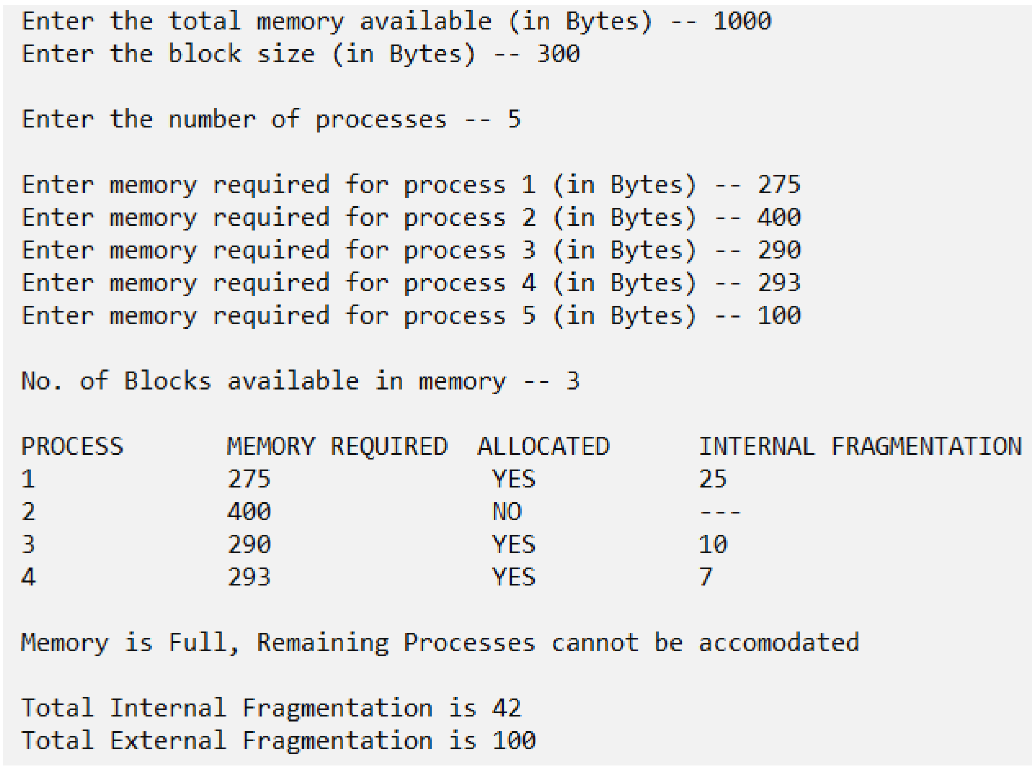
## 3 Implementation in C

```c
/* MFT Memory Management Technique code */
#include<stdio.h>
#include<conio.h>

main()
{
    int  ms, bs, nob, ef,n, mp[10],tif=0;
    int i,p=0;

    printf("  Enter the total memory available (in Bytes) -- ");
    scanf("%d",&ms);

    printf("  Enter the block size (in Bytes) -- ");
    scanf("%d", &bs);

    nob=ms/bs;
    ef=ms - nob*bs;

    printf("\n  Enter the number of processes -- ");
    scanf("%d",&n);
    printf("\n");

    for(i=0; i<n; i++)
    {
        printf("  Enter memory required for process %d (in Bytes) -- ",i+1);
        scanf("%d",&mp[i]);
    }

    printf("\n  No. of Blocks available in memory -- %d",nob);
    printf("\n\n  PROCESS\tMEMORY REQUIRED\t ALLOCATED\tINTERNAL FRAGMENTATION")
        ;

    for(i=0; i<n && p<nob; i++)
    {
        printf("\n  %d\t\t%d",i+1,mp[i]);

        if(mp[i] > bs)
```

```
            printf("\t\t  NO\t\t---");
        else
        {
            printf("\t\t  YES\t\t%d",bs-mp[i]);
            tif = tif + bs-mp[i];
            p++;
        }
    }

    if(i<n)
        printf("\n\n  Memory is Full, Remaining Processes cannot be accomodated");

    printf("\n\n  Total Internal Fragmentation is %d",tif);
    printf("\n  Total External Fragmentation is %d",ef);

    getch();
}
```

## 4 Input/Output

Output of the MFT Memory Management Technique program is given below.



```
Enter the total memory available (in Bytes) -- 1000
Enter the block size (in Bytes) -- 300

Enter the number of processes -- 5

Enter memory required for process 1 (in Bytes) -- 275
Enter memory required for process 2 (in Bytes) -- 400
Enter memory required for process 3 (in Bytes) -- 290
Enter memory required for process 4 (in Bytes) -- 293
Enter memory required for process 5 (in Bytes) -- 100

No. of Blocks available in memory -- 3

PROCESS       MEMORY REQUIRED  ALLOCATED      INTERNAL FRAGMENTATION
1             275               YES           25
2             400               NO            ---
3             290               YES           10
4             293               YES           7

Memory is Full, Remaining Processes cannot be accomodated

Total Internal Fragmentation is 42
Total External Fragmentation is 100
```

## 5 Discussion & Conclusion

Based on the focused objective(s) to understand about MFT Memory Management Technique, the additional lab exercises made me more confident towards the fulfilment of the objectives(s).

## 6 Lab Task (Please implement yourself and show the output to the instructor)

1. Implement MVT Memory Management Technique.

### 6.1 Problem analysis

MVT (Multiprogramming with a Variable number of Tasks) is the memory management technique in which each job gets just the amount of memory it needs. That is, the partitioning of memory is dynamic and changes as jobs enter and leave the system. MVT is a more "efficient" user of resources. MFT suffers from the problem of internal fragmentation and MVT suffers from external fragmentation.

### 6.2 Sample Input/Output

**INPUT:**

```
Enter the total memory available (in Bytes) – 1000
Enter memory required for process 1 (in Bytes) – 400
Memory is allocated for Process 1
Do you want to continue(y/n):y
Enter memory required for process 2 (in Bytes) – 275
Memory is allocated for Process 2
Do you want to continue(y/n):y
Enter memory required for process 3 (in Bytes) – 550
```

**OUTPUT:**

```
Memory is Full
Total Memory Available – 1000
PROCESS ———MEMORY-ALLOCATED
1 ——————————400
2 ——————————275
Total Memory Allocated is 675
Total External Fragmentation is 325
```

## 7 Lab Exercise (Submit as a report)

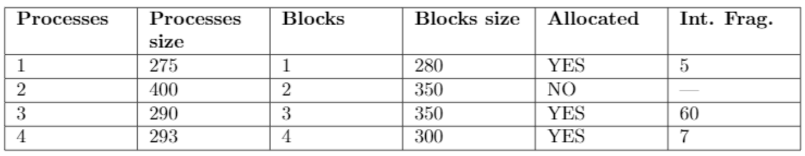
- Implement a code to solve the Memory Management technique problem.

  **Input:**

  ```
  Enter the number of Blocks– 4
  Block 1 size: 280
  Block 2 size: 350
  Block 3 size: 300
  Block 4 size: 320
  Enter the number of processes – 4
  Enter memory required for process 1 – 275
  Enter memory required for process 2 – 400
  Enter memory required for process 3 – 290
  Enter memory required for process 4 – 293
  ```

  **Output:**

Table 1: **Sample Output**



| **Processes** | **Processes size** | **Blocks** | **Blocks size** | **Allocated** | **Int. Frag.** |
|---|---|---|---|---|---|
| 1 | 275 | 1 | 280 | YES | 5 |
| 2 | 400 | 2 | 350 | NO | — |
| 3 | 290 | 3 | 350 | YES | 60 |
| 4 | 293 | 4 | 300 | YES | 7 |

## 8 Policy

Copying from the Internet, classmate, seniors, or from any other source is strongly prohibited. 100% marks will be *deducted* if any such copying is detected.