Experiment No 8
Class: SE Comp
Year: 2020-21
Performed by: Danyl Fernandes, 72

Aim: Write a program to demonstrate the concept of MVT and MFT memory management techniques

Theory:

Fixed Partitioning:

- This is the oldest and simplest technique used to put more than one process in the main memory.
- In this partitioning, the number of partitions (non-overlapping) in RAM are fixed but the size of each partition may or may not be the same.
- As it is a contiguous allocation, hence no spanning is allowed.
- Here partition are made before execution or during system configure

Advantages:

- Easy to implement
- Little OS overhead

Disadvantages:

- Internal Fragmentation
- External Fragmentation
- Limit process size
- Limitation on Degree of Multiprogramming

Variable Partitioning:

It is a part of Contiguous allocation technique. It is used to alleviate the problem faced by Fixed Partitioning. In contrast with fixed partitioning, partitions are not made before the execution or during system configure. Various features associated with variable Partitioning -

- Initially RAM is empty and partitions are made during the run-time according to process's need instead of partitioning during system configure.
- The size of the partition will be equal to the incoming process.
- The partition size varies according to the need of the process so that the internal fragmentation can be avoided to ensure efficient utilisation of RAM.
- Number of partitions in RAM is not fixed and depends on the number of incoming processes and Main Memory's size.

Advantages:

- No Internal Fragmentation
- No restriction on Degree of Multiprogramming
- No Limitation on the size of the process

Disadvantages:

- Difficult Implementation
- External Fragmentation

Conclusion:

- In this experiment, we were successfully able to implement & demonstrate the the concept of MVT & MFT memory management techniques

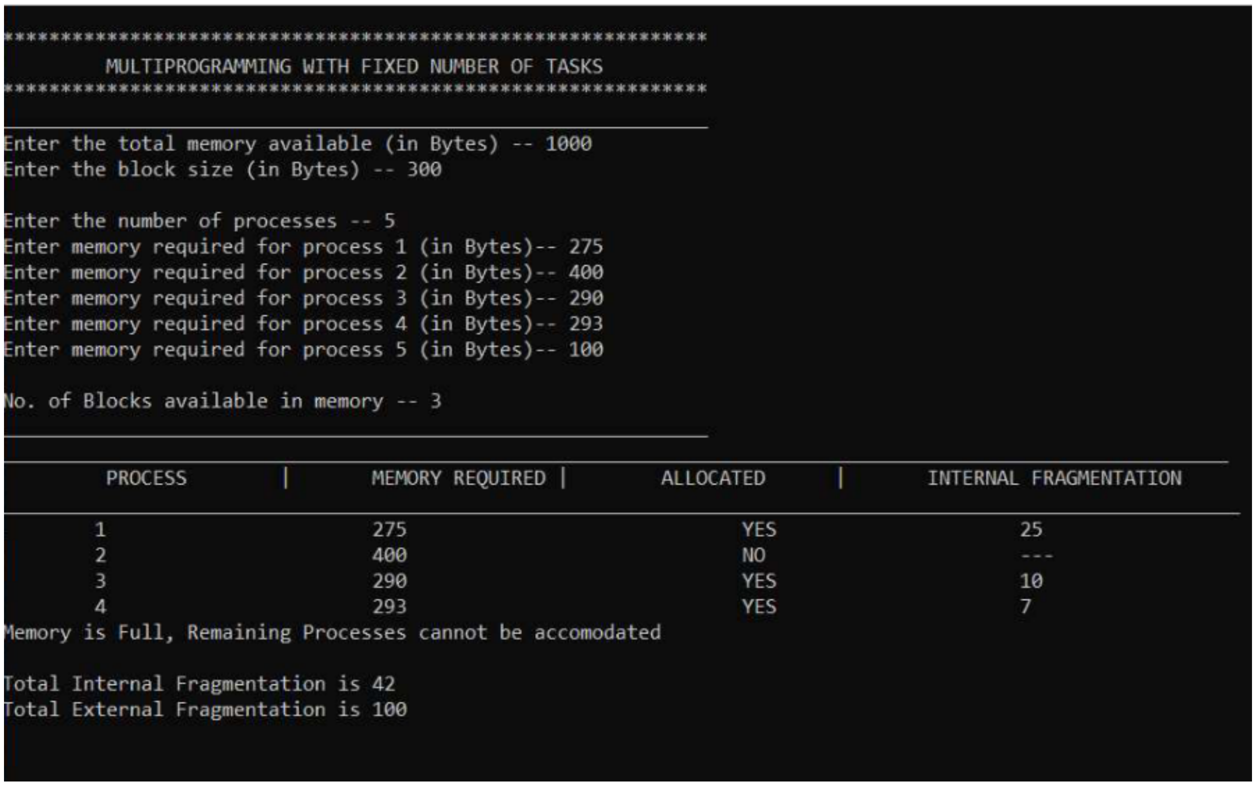
MFT Memory Management Technique:

Code:

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

main() {
int memory_size, block_size, no_blocks, external_fragmentation,n,
process_memory[100],total_internal_fragmentation=0;
int i,p=0;
printf("\n************************************************************
*");

printf("\nMULTIPROGRAMMING WITH FIXED NUMBER OF TASKS");
printf("\n************************************************************
*");
printf("\n____________________________________________________________
_");
printf("\nEnter the total memory available (in Bytes) -- ");
scanf("%d",&memory_size);
printf("Enter the block size (in Bytes) -- ");
scanf("%d", &block_size);
no_blocks=memory_size/block_size;

external_fragmentation=memory_size - no_blocks*block_size;
printf("\nEnter the number of processes -- ");
scanf("%d",&n);

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

printf("\nNo. of Blocks available in memory -- %d",no_blocks);
printf("\n____________________________________________________________
_");

     printf("\n_______________________________________________________
___________________________________________________");
printf("\n\t PROCESS\t|\tMEMORY REQUIRED\t|\t ALLOCATED\t|\tINTERNAL
FRAGMENTATION");
printf("\n____________________________________________________________
_______________________________________________");

for(i=0;i<n && p<no_blocks;i++) {
```

```
printf("\n \t%d\t\t\t%d\t",i+1,process_memory[i]);
if(process_memory[i] > block_size)
printf("\t\t\tNO\t\t\t---");
else {
printf("\t\t\tYES\t\t\t%d",block_size-process_memory[i]);
total_internal_fragmentation = total_internal_fragmentation +
block_size-process_memory[i];
p++;
}
}

if(i<n)
printf("\nMemory is Full, Remaining Processes cannot be accomodated");
printf("\n\nTotal Internal Fragmentation is
%d",total_internal_fragmentation);
printf("\nTotal External Fragmentation is %d",external_fragmentation);
getch();
}
```

Output:



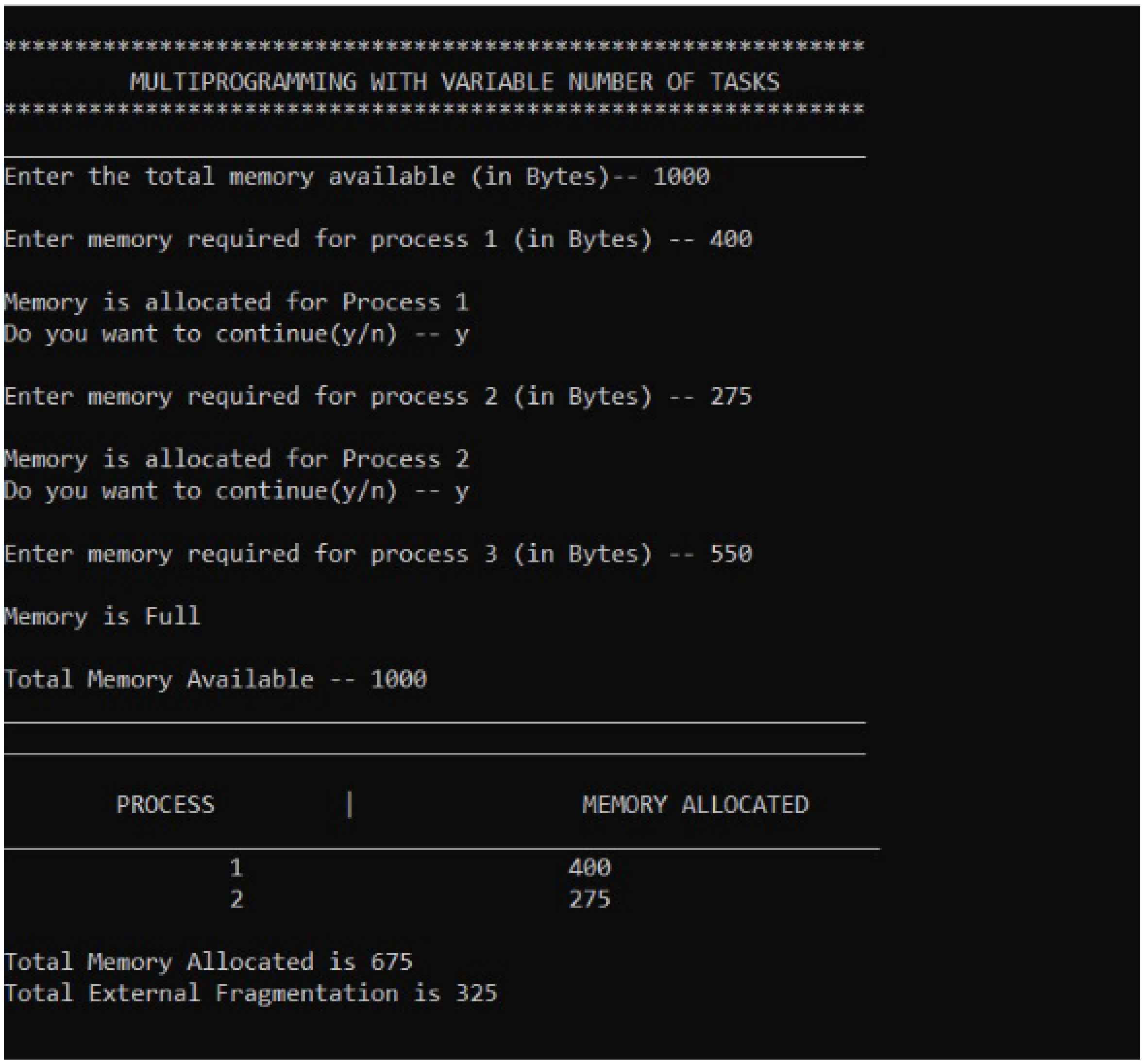
```
*************************************************************
        MULTIPROGRAMMING WITH FIXED NUMBER OF TASKS
*************************************************************
_____________________________________________________________
Enter the total memory available (in Bytes) -- 1000
Enter the 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
```

MVT Memory Management Technique:

Code:

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

main() {
int available_memory,process_memory[100],i, temp,n=0; char ch = 'y';
printf("\n************************************************************
*");
printf("\n      MULTIPROGRAMMING WITH VARIABLE NUMBER OF TASKS
");
printf("\n************************************************************
*");

printf("\n____________________________________________________________
_");
printf("\nEnter the total memory available (in Bytes)-- ");
scanf("%d",&available_memory);
temp=available_memory;

for(i=0;ch=='y';i++,n++) {
printf("\nEnter memory required for process %d (in Bytes) -- ",i+1);
scanf("%d",&process_memory[i]);
if(process_memory[i]<=temp) {
printf("\nMemory is allocated for Process %d ",i+1);
temp = temp - process_memory[i];
} else {
printf("\nMemory is Full");
break;
}

printf("\nDo you want to continue(y/n) -- ");
scanf(" %c", &ch);
}

printf("\n\nTotal Memory Available -- %d", available_memory);
printf("\n____________________________________________________________
_");
printf("\n____________________________________________________________
_");
printf("\n\n\tPROCESS\t\t|\t\t MEMORY ALLOCATED ");
printf("\n____________________________________________________________
__");
for(i=0;i<n;i++)
printf("\n \t\t%d\t\t\t%d",i+1,process_memory[i]);
printf("\n\nTotal Memory Allocated is %d",available_memory-temp);
printf("\nTotal External Fragmentation is %d",temp);

getch();
}
```

Output:



```
**************************************************************
         MULTIPROGRAMMING WITH VARIABLE NUMBER OF TASKS
**************************************************************
______________________________________________________________
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

Memory is Full

Total Memory Available -- 1000
______________________________________________________________
______________________________________________________________

        PROCESS         |                MEMORY ALLOCATED
______________________________________________________________
                1                       400
                2                       275

Total Memory Allocated is 675
Total External Fragmentation is 325
```