CERTIFICATE

Name of the Lab : OPERATING SYSTEMS

Name of the Student : JASWANTH YENDURI

Student Regd. No. : 18BQ1A05M8

CLASS : III B.TECH. I SEM CSE – D

GIT HUB LINK: https://github.com/Jaswanth-yenduri/OS-Lab

INDEX

|  |  |  |
| --- | --- | --- |
| S. No. | Name of the Experiment | Page No. |
| 2 (a) | Simulate Multiprogramming with a fixed number of tasks (MFT) | 03- |
| 2 (b) | Simulate Multiprogramming with a variable number of tasks (MVT) |  |

**EXPERIMENT NO: 3(a)**

**AIM :** Stimulate Multiprogramming with a fixed number of tasks (MFT)

**DESCRIPTION :** In operating systems, Memory Management is the function responsible for allocating and managing computer’s main memory.

Memory Management function keeps track of the status of each memory location, either allocated or free to ensure effective and efficient use of Primary Memory.

There are two Memory Management Techniques**: Contiguous**, and **NonContiguous**.

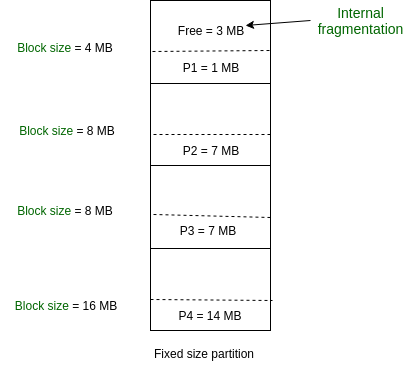
In Contiguous Technique, executing process must be loaded entirely in mainmemory. Contiguous Technique can be divided into:

Fixed (or static) partitioning

Variable (or dynamic) partitioning

Fixed Partitioning:

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



As illustrated in above figure, first process is only consuming 1MB out of 4MB in the main memory.

Hence, Internal Fragmentation in first block is (4-1) = 3MB. Sum of Internal Fragmentation in every block = (4-1)+(8-7)+(8-7)+(16-14)= 3+1+1+2 = 7MB.

Suppose process P5 of size 7MB comes. But this process cannot be accommodated inspite of available free space because of contiguous allocation (as spanning is not allowed). Hence, 7MB becomes part of External Fragmentation.

There are some advantages and disadvantages of fixed partitioning.

Advantages of Fixed Partitioning :

• Easy to implement: Algorithms needed to implement Fixed Partitioning are easy to implement. It simply requires putting a process into certain partition without focussing on the emergence of Internal and External Fragmentation.

• Little OS overhead: Processing of Fixed Partitioning require lesser excess and indirect computational power.

Disadvantages of Fixed Partitioning :

• Internal Fragmentation: Main memory use is inefficient. Any program, no matter how small, occupies an entire partition. This can cause internal fragmentation.

• External Fragmentation: The total unused space (as stated above) of various partitions cannot be used to load the processes even though there is space available but not in the contiguous form (as spanning is not allowed).

• Limit process size: Process of size greater than size of partition in Main Memory cannot be accommodated. Partition size cannot be varied according to the size of incoming process’s size. Hence, process size of 32MB in above stated example is invalid.

**PROGRAMMING LANGUAGE USED:** Python

**LIBRARIES USED:**  No built-in libraries used

**PROGRAM:**

from texttable import Texttable

t = Texttable()

t.add\_row(['Process','Memory acquired','Allocated?','Internal fragmentation'])

process\_memory = []

tif=0

total\_memory = int(input("Enter the total memory available : "))

blocksize = int(input("Enter the blocksize: "))

numOfBlocks = int(total\_memory/blocksize)

processes\_num = int(input("Enter number of processes:"))

externalfrag=total\_memory - (processes\_num\*blocksize)

if(processes\_num <= numOfBlocks):

for i in range(0,processes\_num):

print("Enter the size of process",i+1,'in bytes : ')

mem = int(input())

process\_memory.append(mem)

print("Number of block in memory are:",numOfBlocks)

for i in range(0,processes\_num):

internalfrag = blocksize - process\_memory[i]

if internalfrag >= 0:

tif += internalfrag

else:

internalfrag = blocksize

if(process\_memory[i] > blocksize):

isAllocated = 'NO'

else:

isAllocated = 'YES'

t.add\_row([i+1,process\_memory[i],isAllocated,internalfrag])

print(t.draw())

print("external fragmentation : ",externalfrag)

print("total internal fragmentation : ",tif)

else:

print("Number of processes is more than number of partitions")

**OUTPUT-1:**

Enter the total memory available : 1000

Enter the blocksize: 200

Enter number of processes:5

Enter the size of process 1 in bytes : 123

Enter the size of process 2 in bytes : 45

Enter the size of process 3 in bytes : 200

Enter the size of process 4 in bytes : 201

Enter the size of process 5 in bytes : 456

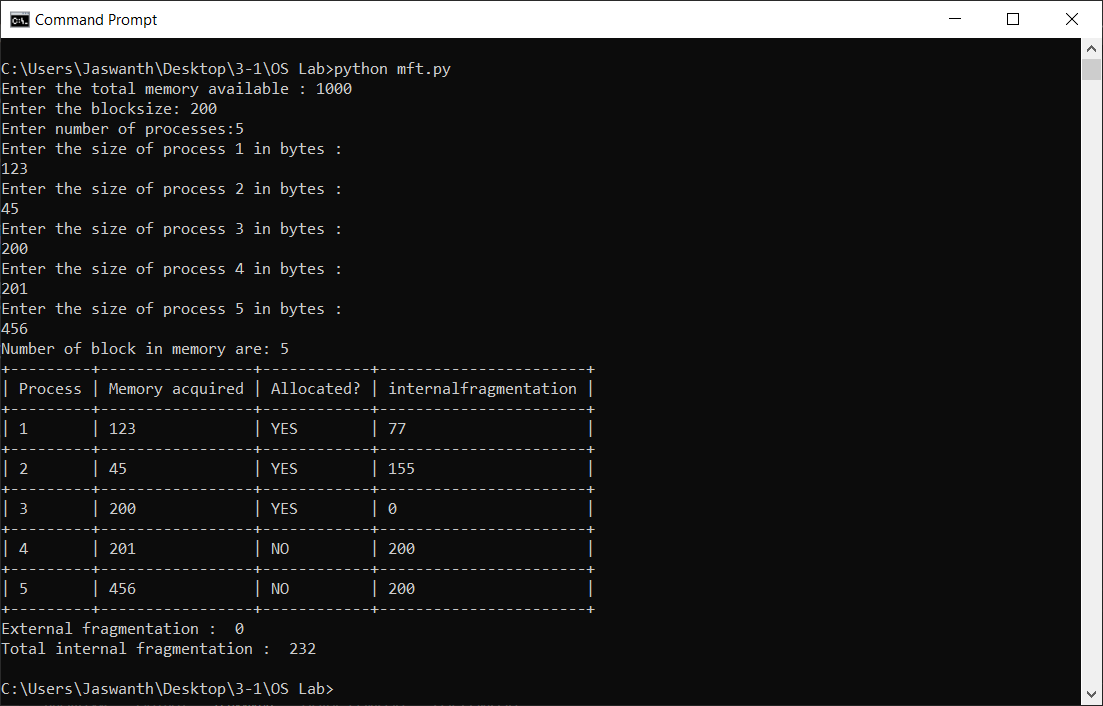
Number of block in memory are: 5

|  |  |  |  |
| --- | --- | --- | --- |
| Process | Memory Acquires | Allocated? | Internam Fragmentation |
| 1 | 123 | YES | 77 |
| 2 | 45 | YES | 155 |
| 3 | 200 | YES | 0 |
| 4 | 201 | NO | 200 |
| 5 | 456 | NO | 200 |

External fragmentation : 0

Total internal fragmentation : 232

**OUTPUT-1 SCREENSHOT:**



**OUTPUT-2:**

Enter the total memory available : 500

Enter the blocksize: 75

Enter number of processes:6

Enter the size of process 1 in bytes : 100

Enter the size of process 2 in bytes : 67

Enter the size of process 3 in bytes : 76

Enter the size of process 4 in bytes : 75

Enter the size of process 5 in bytes : 74

Enter the size of process 6 in bytes : 9

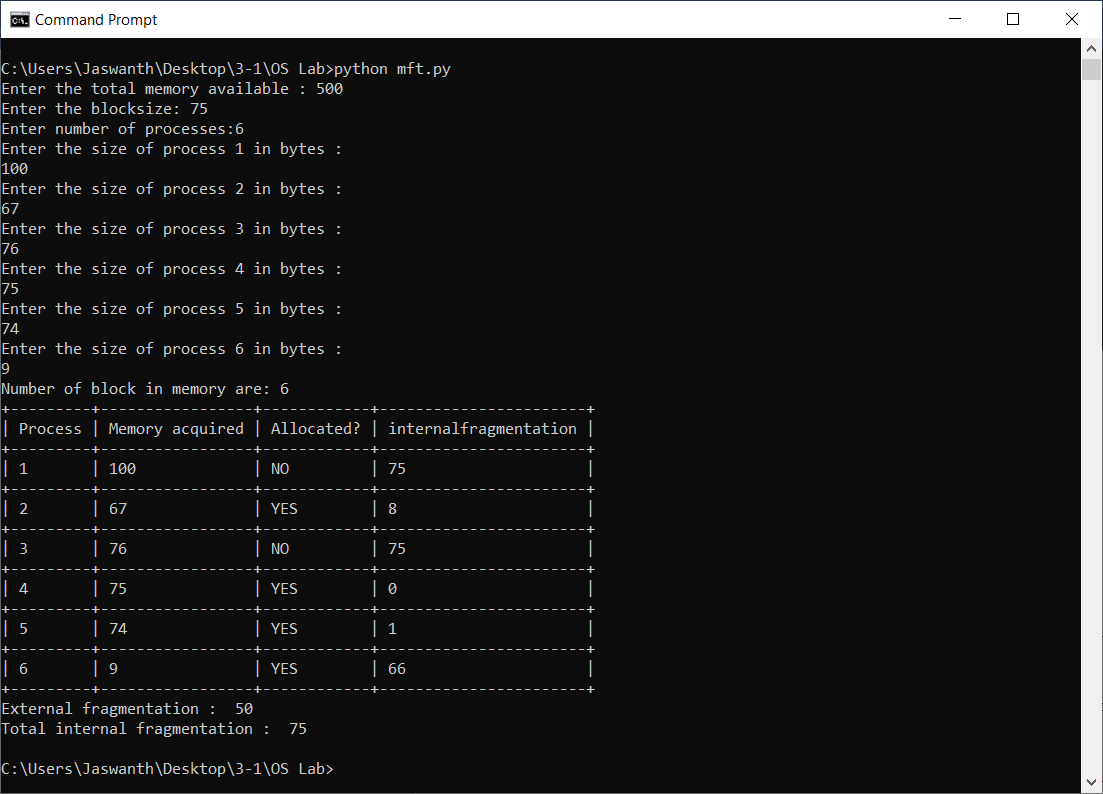
Number of block in memory are: 6

|  |  |  |  |
| --- | --- | --- | --- |
| Process | Memory Acquired | Alloated? | Internal fragmentation |
| 1 | 100 | NO | 75 |
| 2 | 67 | YES | 8 |
| 3 | 76 | NO | 75 |
| 4 | 75 | YES | 0 |
| 5 | 74 | YES | 1 |
| 6 | 9 | YES | 66 |

External fragmentation : 50

Total internal fragmentation : 75

**OUTPUT-2 SCREENSHOT:**



**EXPERIMENT NO: 3(a)**

**AIM :** Stimulate Multiprogramming with a variable number of tasks (MVT)

**DESCRIPTION :** In operating systems, Memory Management is the function responsible for allocating and managing computer’s main memory.

Memory Management function keeps track of the status of each memory location, either allocated or free to ensure effective and efficient use of Primary Memory.

There are two Memory Management Techniques**: Contiguous**, and **NonContiguous**.

In Contiguous Technique, executing process must be loaded entirely in mainmemory. Contiguous Technique can be divided into:

Fixed (or static) partitioning

Variable (or dynamic) partitioning

Variable Partitioning :

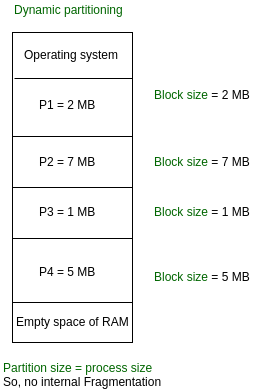
It is a part of Contiguous allocation technique. It is used to alleviate the problem faced by Fixed Partitioning. In contrast with fixed partitioning, partitions are not made before the execution or during system configure. Various features associated with variable Partitioning.

1. Initially RAM is empty and partitions are made during the run-time according to process’s need instead of partitioning during system configure.

2. The size of partition will be equal to incoming process.

3. 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.

4. Number of partitions in RAM is not fixed and depends on the number of incoming process and Main Memory’s size.



There are some advantages and disadvantages of variable partitioning over fixed partitioning as given below.

Advantages of Variable Partitioning :

* No Internal Fragmentation: In variable Partitioning, space in main memory is allocated strictly according to the need of process, hence there is no case of internal fragmentation. There will be no unused space left in the partition.
* No restriction on Degree of Multiprogramming:
  + More number of processes can be accommodated due to absence of internal fragmentation. A process can be loaded until the memory is not empty.
* No Limitation on the size of the process:
  + In Fixed partitioning, the process with the size greater than the size of the largest partition could not be loaded and process can not be divided as it is invalid in contiguous allocation technique. Here, In variable partitioning, the process size can’t be restricted since the partition size is decided according to the process size.

Disadvantages of Variable Partitioning :

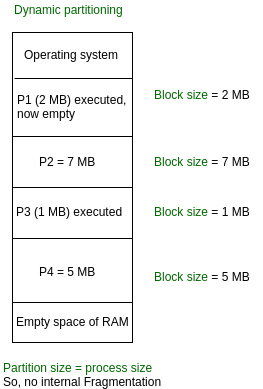
1. Difficult Implementation:

Implementing variable Partitioning is difficult as compared to Fixed Partitioning as it involves allocation of memory during run-time rather than during system configure.

1. External Fragmentation:

There will be external fragmentation inspite of absence of internal fragmentation.

For example, suppose in above example- process P1(2MB) and process P3(1MB) completed their execution. Hence two spaces are left i.e. 2MB and 1MB. Let’s suppose process P5 of size 3MB comes. The empty space in memory cannot be allocated as no spanning is allowed in contiguous allocation. The rule says that process must be contiguously present in main memory to get executed. Hence it results in External Fragmentation.



Now P5 of size 3 MB cannot be accommodated in spite of required available space because in contiguous no spanning is allowed.

**PROGRAMMING LANGUAGE USED:** Python

**LIBRARIES USED:**  TextTable from texttable package

**PROGRAM:**

from texttable import Texttable

t = Texttable()

t.add\_row(['Process','Memory allocated'])

total\_memory = int(input("Enter the total memory available in bytes : "))

temp=total\_memory

flag= True

i=1

while flag:

print("Enter memory of process ",i,"in bytes : ",end="")

mem = int(input())

if mem <= temp:

temp -= mem

t.add\_row([i,mem])

i+=1

else:

print("Total memory execeeded")

break

inp = input("Do you want add another process? (y/n) : ")

if inp=='n':

flag=False

print(t.draw())

print("Total memory used is:",total\_memory-temp)

print("External fragmentation is :",temp)

**OUTPUT - 1:**

Enter the total memory available in bytes : 1000

Enter memory of process 1 in bytes : 456

Do you want add another process? (y/n) : y

Enter memory of process 2 in bytes : 239

Do you want add another process? (y/n) : y

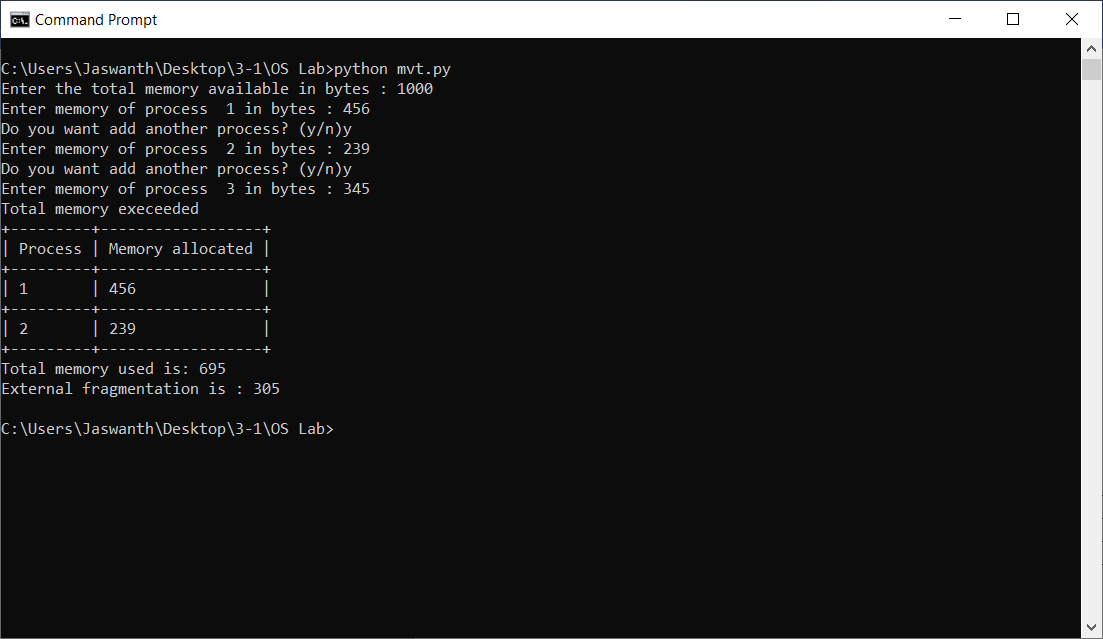
Enter memory of process 2 in bytes : 345

Total memory exceeded

|  |  |
| --- | --- |
| Process | Memory allocated |
| 1 | 456 |
| 2 | 239 |

Total memory used is: 695

External fragmentation is : 305

**OUTPUT -1 SCREENSHOT :**

**OUTPUT - 2:**

Enter the total memory available in bytes : 5000

Enter memory of process 1 in bytes : 256

Do you want add another process? (y/n) : y

Enter memory of process 2 in bytes : 1256

Do you want add another process? (y/n) : y

Enter memory of process 3 in bytes : 2467

Do you want add another process? (y/n) : n

|  |  |
| --- | --- |
| Process | Memory allocated |
| 1 | 256 |
| 2 | 1256 |
| 3 | 2467 |

Total memory used is: 3979

External fragmentation is : 1021

**OUTPUT-2 SCREENSHOT:**

