B.M.S College of Engineering
P.O. Box No.: 1908 Bull Temple Road,
Bangalore-560 019

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

Course –Object oriented programming using C++
Course Code –18IS3PCOOP
AY 2020-21

Title:

LIFECYCLE OF A PROCESS

Submitted to – Dr V Shubha Rao
Submitted by -

H Nidhi 1BM19IS056

Kamal Pratik 1BM19IS067

B.M.S College of Engineering
P.O. Box No.: 1908 Bull Temple Road,
Bangalore-560 019

INDEX

<u>Abstract</u>

This mini project is on the life cycle of a process using dynamic size partitioned memory. The program simulates aspects of the operating system. It simulates the lifecycle of a process with priority scheduling and memory management by "first-fit" approach.

Following are the details of the command by the user.

/——————————————————commands details——————————————————/

A <priority> <size> **-** Create a new process

t - Terminate the process in CPU

d <disk number> - The process that currently uses the CPU requests the hard disk #number.

p <printer number> - The process that currently uses the CPU requests the printer #number.

D <disk number> - The hard disk #number has finished the work for one process.

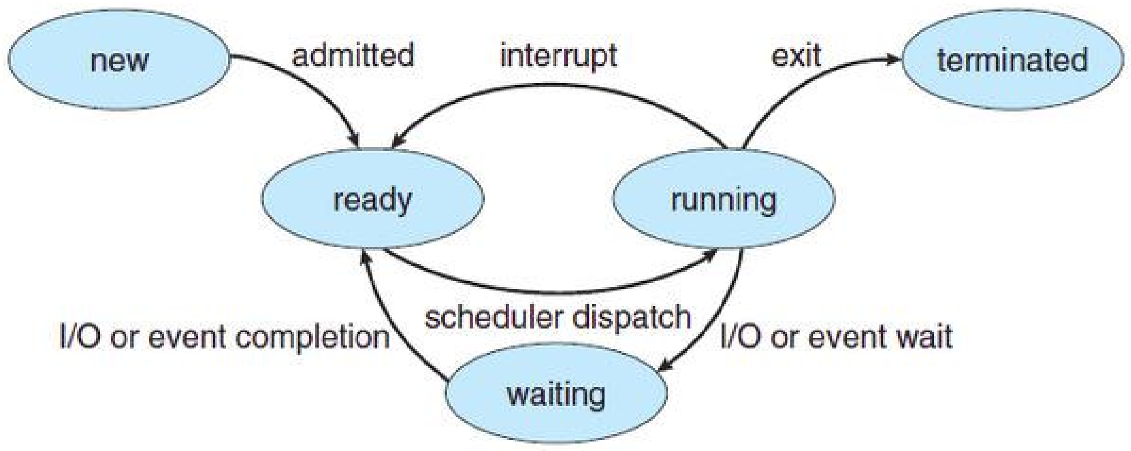
P <printer number> - The printer #number has finished the work for one process.

S r - Shows what process is currently using the CPU and what processes are waiting to use the CPU on each level of the ready-queue.

S i - Shows what processes are currently using the I/O devices and what processes are waiting to use them.

S m - Shows the state of memory.

/——/



INTRODUCTION

There are 5 priority ready-queues, priority levels numbered from 0 to 4 (0 is the lowest priority, and 4 is the highest). Within each level of the ready-queue the FirstCome-FirstServe algorithm is used to send the process from the ready-queue to the CPU. A process from the same level of the ready-queue can be scheduled only if all the processes of higher levels are empty. However, the scheduling is preemptive: When a higher priority process becomes available for execution the lower priority process is preempted from the CPU and sent to the end of its level of the ready queue. A point to note is that in our system architecture, the CPU can work on only one process at a given time.

Inputs are taken from the user, for accepting the size of memory we'll be using, the number of Hard Drive Disks, and the number of Printers needed.The user is then given the option of executing one of the following commands : create a new process, request for IO, finish work at IO, terminate a running process and show status of any part of the program (CPU/IO/Memory)

When the user chooses to create a new process, a block of memory is reserved for the process using "First-Fit" algorithm to find the first suitable hole and a PCB is assigned to the process which basically contains all the information about the process (PID, priority, size, where it's located in the memory). After it is successfully settled in the memory,it is then either scheduled to the CPU or put in the waiting queue depending on factors like - an idle CPU and priority of the currently executing process. This scheduling process follows "Preemptive priority scheduling".

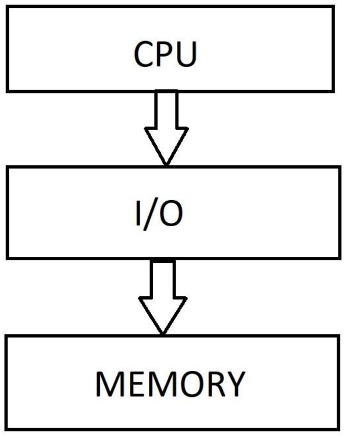
When the user chooses to request for IO(either input through Hard Disk Drive or output through the Printer) the currently executing process is moved from the CPU to the respective IO device for the input/output operation. And the CPU schedules the next process following "Priority scheduling". In case the CPU was idle when the user requests for IO, an error is thrown saying that no process is running and the request for IO is declined.

If the user chooses to finish the work at IO, and the IO devices are idle, an error is thrown indicating the same. Otherwise, if any process is using the devices, then that device gives up the process it was executing and puts it in the ready queue, which again follows "Preemptive priority scheduling". If any other process was waiting to use the respective IO device, then that process is given control over the IO, otherwise the IO remains idle.

If the user chooses to terminate the currently executing process in the CPU, then the process is removed from the CPU and the same process is located in the memory, when found it is freed from the memory, if not found an error is thrown telling the user that process wasn't found. This removal of the process results in creating a hole in the memory which will next be utilized when a new process is created according to the "First-fit" match algorithm.Finally, at each step the user is given the choice to get the current status of the Memory & CPU & IO devices. The process using the CPU/IO along with the processes waiting for CPU and IO are displayed to the output screen. And the processes along with their location in the memory are also displayed when the status of memory is requested by the user.

<u>OOPS concepts used</u>

- Exception Handling - Used to catch and throw errors throughout the program.
- Inheritance - Multilevel Inheritance is used.’CPU’ class has a derived class ‘IO’ which intern has another derived class ‘memory’.NO more changes right?



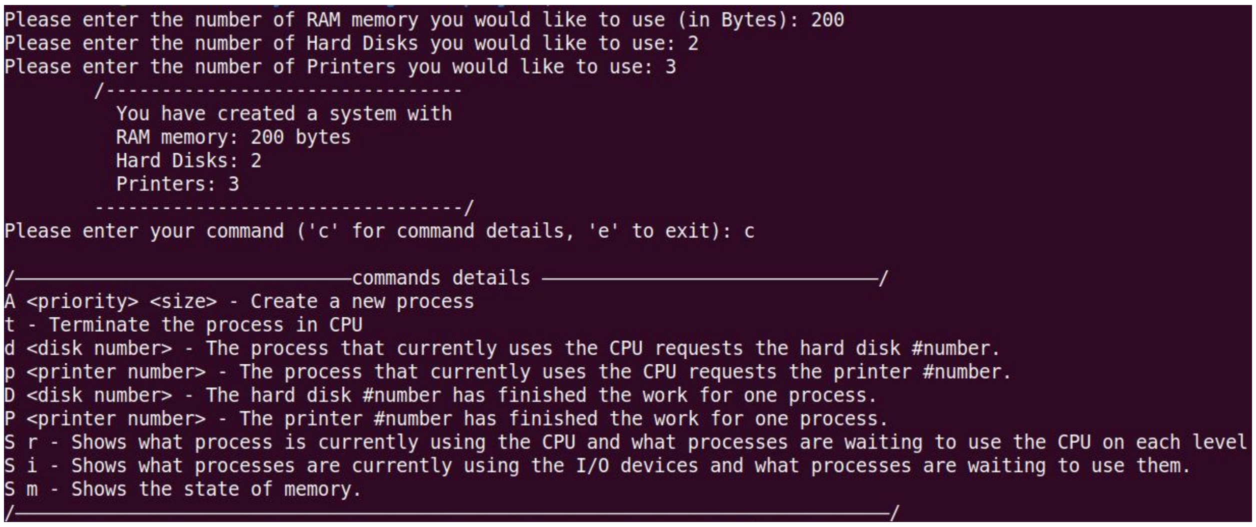
- Modular Programming is used to display the code in an organised manner to make it easier to understand for anybody else working on the code.
- File Handling - Reading from the commands file ‘c.txt’ and displaying the contents in the output screen.
- Classes and objects: We have 3 classes. They are CPU,IO which is derived from CPU and Memory derived from IO. An object mem of class memory is made to demonstrate the lifecycle of a process.
- Standard Template Library
    - ➔ Lists: Used to store processes in the ready queue, hard disks and printer.
    - ➔ Vector: The total number of hard disks and printers are stored in vectors.
    - ➔ Iterators: They are used to iterate through lists of ready queues when we need to print them.

SOFTWARES USED

- VS CODE EDITOR
- MINGW COMPILER
- TEXT EDITOR

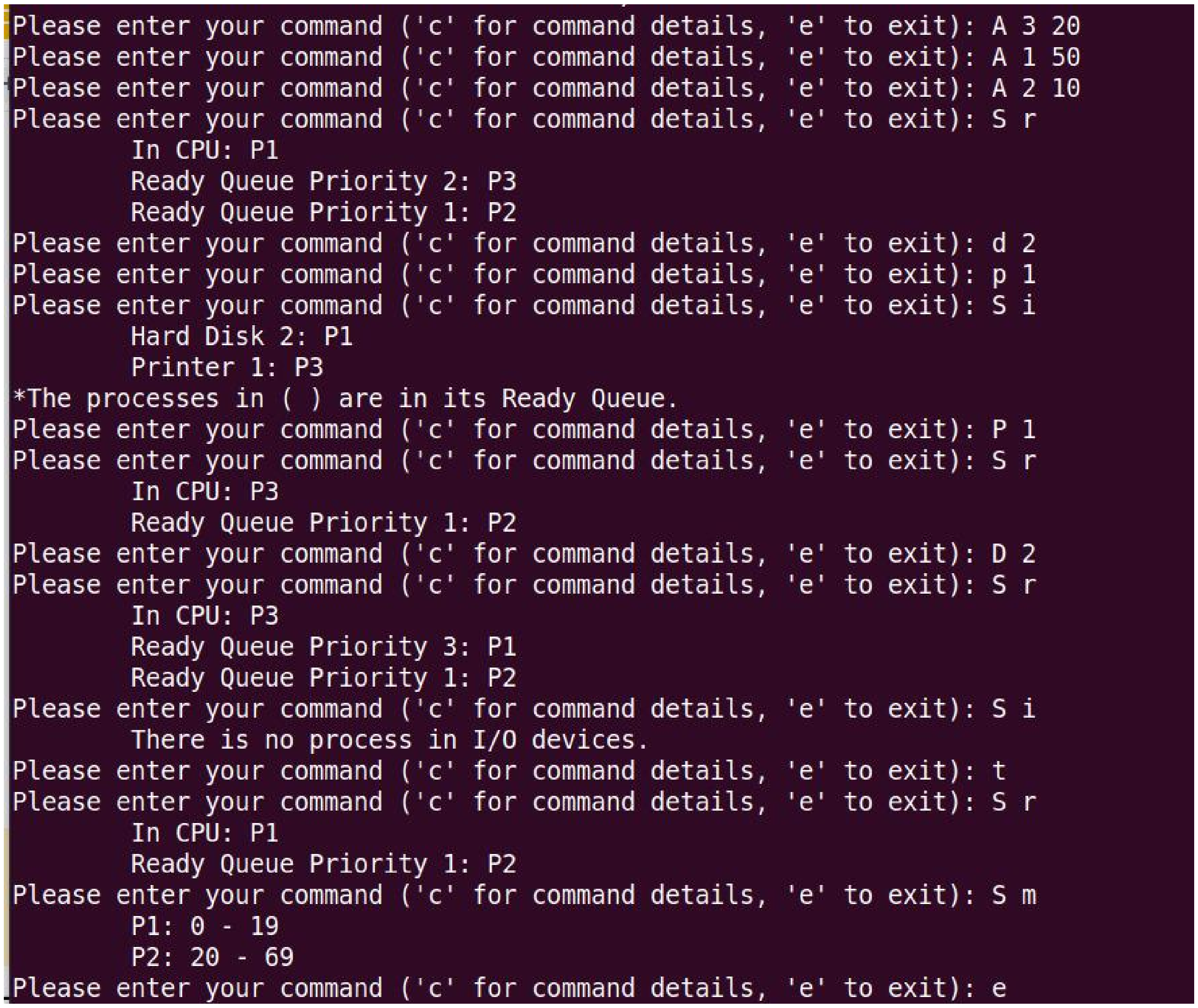
SNAPSHOTS

Output window:



```
Please enter the number of RAM memory you would like to use (in Bytes): 200
Please enter the number of Hard Disks you would like to use: 2
Please enter the number of Printers you would like to use: 3
        /--------------------------------
          You have created a system with
          RAM memory: 200 bytes
          Hard Disks: 2
          Printers: 3
        ---------------------------------/
Please enter your command ('c' for command details, 'e' to exit): c

/——————————————————————————————commands details ——————————————————————————————/
A <priority> <size> - Create a new process
t - Terminate the process in CPU
d <disk number> - The process that currently uses the CPU requests the hard disk #number.
p <printer number> - The process that currently uses the CPU requests the printer #number.
D <disk number> - The hard disk #number has finished the work for one process.
P <printer number> - The printer #number has finished the work for one process.
S r - Shows what process is currently using the CPU and what processes are waiting to use the CPU on each level
S i - Shows what processes are currently using the I/O devices and what processes are waiting to use them.
S m - Shows the state of memory.
/——————————————————————————————————————————————————————————————————————————————/
```



```
Please enter your command ('c' for command details, 'e' to exit): A 3 20
Please enter your command ('c' for command details, 'e' to exit): A 1 50
Please enter your command ('c' for command details, 'e' to exit): A 2 10
Please enter your command ('c' for command details, 'e' to exit): S r
        In CPU: P1
        Ready Queue Priority 2: P3
        Ready Queue Priority 1: P2
Please enter your command ('c' for command details, 'e' to exit): d 2
Please enter your command ('c' for command details, 'e' to exit): p 1
Please enter your command ('c' for command details, 'e' to exit): S i
        Hard Disk 2: P1
        Printer 1: P3
*The processes in ( ) are in its Ready Queue.
Please enter your command ('c' for command details, 'e' to exit): P 1
Please enter your command ('c' for command details, 'e' to exit): S r
        In CPU: P3
        Ready Queue Priority 1: P2
Please enter your command ('c' for command details, 'e' to exit): D 2
Please enter your command ('c' for command details, 'e' to exit): S r
        In CPU: P3
        Ready Queue Priority 3: P1
        Ready Queue Priority 1: P2
Please enter your command ('c' for command details, 'e' to exit): S i
        There is no process in I/O devices.
Please enter your command ('c' for command details, 'e' to exit): t
Please enter your command ('c' for command details, 'e' to exit): S r
        In CPU: P1
        Ready Queue Priority 1: P2
Please enter your command ('c' for command details, 'e' to exit): S m
        P1: 0 - 19
        P2: 20 - 69
Please enter your command ('c' for command details, 'e' to exit): e
```

REFERENCES

https://www.guru99.com/fcfs-scheduling.html
https://www.geeksforgeeks.org/first-fit-allocation-in-operating-systems/
https://zitoc.com/process-life-cycle/
https://prepinsta.com/operating-systems/process-life-cycle/
https://www.geeksforgeeks.org/process-table-and-process-control-block-pcb/
https://www.geeksforgeeks.org/file-handling-c-classes/
https://www.tutorialspoint.com/cplusplus/cpp_exceptions_handling.htm