EOPSY Lab-3

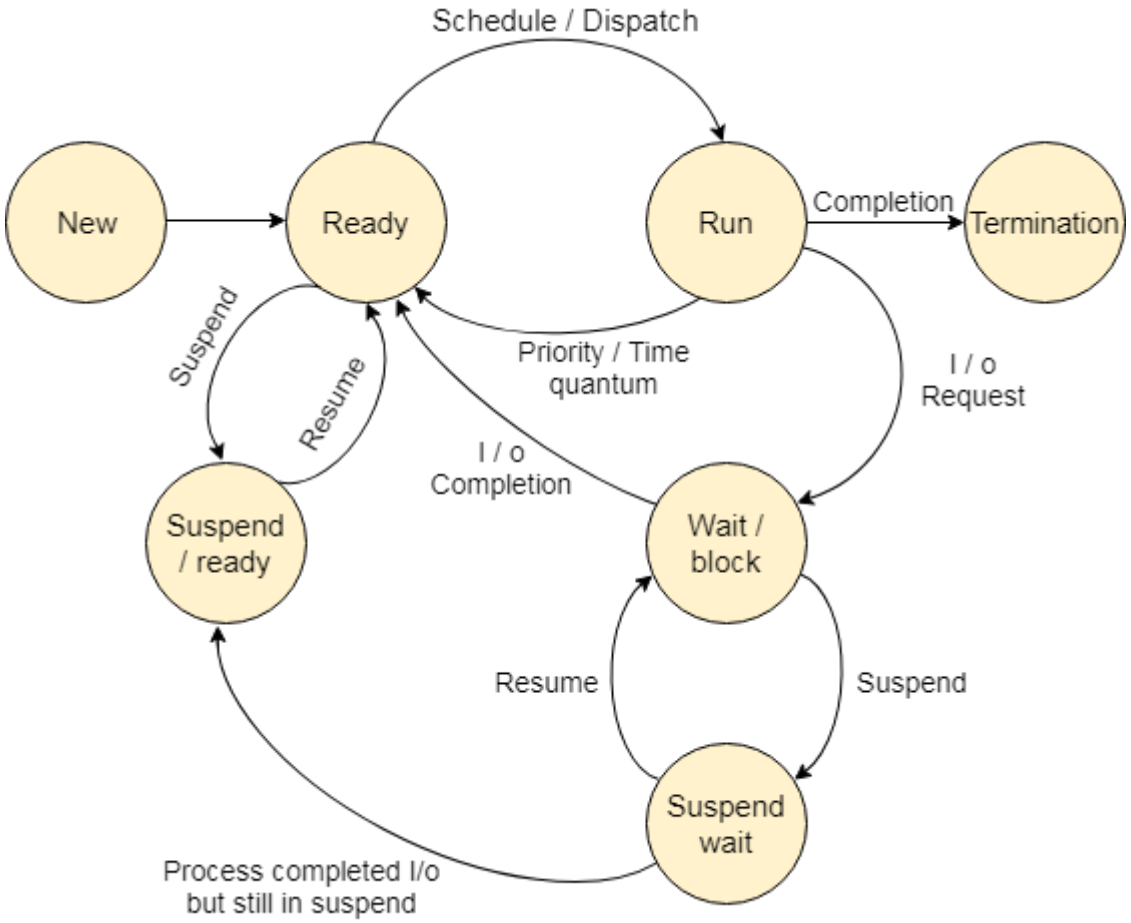
Name: MADHANA GOPALAPURAM RAMESH RATHISH AADIRAJA
ID: 302457
Group: G

Introduction:

The objective of this task is to create multiple processes in MOSS Scheduling simulator with an average run time of 2000 milliseconds, standard deviation of zero and which are blocked for I/O operations for every 500 milliseconds with total simulation time of 10000 milliseconds and observe how the process scheduling works for different number of processes to be executed at same time.

Description:

Generally, a processes goes through several states from its creation to its termination. A diagram representing such states is shown below.



A program or several lines of code which is to be picked up by the operating system from secondary memory to its main memory to initiate its execution is said to be in its new state. In ready state the process is being picked up in main memory and it waits for the execution by the processor(waits for the CPU to be assigned). From the ready state a process can go to either suspend/ready state or run state. If there is any other process which has a higher priority than the current process comes to the ready state, and at the same time if there is no free space in main memory the current process is suspended and its send to suspend/ready state and it is brought back again to ready state when the main memory is available. If the main memory is free enough it will go to the run state where the process is assigned to the CPU execution. From the run state if the process does not have any I/O operation or any other priorities it will terminate(goes to termination state). If it has any resource to be assigned or any I/O operation it is sent to wait/block state. Again in this state if there is another process with higher priority and the main memory is full the current process is sent to suspend wait state where it waits until the main memory becomes available. When it is available it is sent back to wait/block state. From suspend wait state it can also be sent to suspend ready state when the I/O operation is completed but the main memory is unavailable. After the completion of I/O operation in wait/block state the process is sent to ready state and the same repeats again.

When the CPU is free(not totally occupied by processes) the processes which are in the ready queue should be selected. This selection process is done by the CPU scheduler. After a process is selected the dispatcher gives actual control of CPU to the process. As discussed above when a process goes from running state to waiting state or when a process terminates the only choice left for the CPU scheduler is to schedule the next process which is waiting in the ready queue, this scheduling scheme is called nonpreemptive scheduling. When a process goes from running sate to ready state when it is interrupted or from waiting state to ready state after completing I/O operation they are not yet finished and still waits for CPU to complete its execution. In this situation the CPU schedules using different algorithms and selects the process with highest priority, this scheduling scheme is called preemptive scheduling.

Results:

Two processes:

Summary-Results

```
Scheduling Type: Batch (Nonpreemptive)
Scheduling Name: First-Come First-Served
Simulation Run Time: 4000
Mean: 2000
Standard Deviation: 0
Process #       CPU Time        IO Blocking     CPU Completed   CPU Blocked
0               2000 (ms)       500 (ms)        2000 (ms)       3 times
1               2000 (ms)       500 (ms)        2000 (ms)       3 times
```

Summary-Processes

Format:

Process: process-number process-status… (CPU-time block-time accumulated-time accumulated-time)

```
Process: 0 registered... (2000 500 0 0)
Process: 0 I/O blocked... (2000 500 500 500)
Process: 1 registered... (2000 500 0 0)
Process: 1 I/O blocked... (2000 500 500 500)
Process: 0 registered... (2000 500 500 500)
Process: 0 I/O blocked... (2000 500 1000 1000)
Process: 1 registered... (2000 500 500 500)
Process: 1 I/O blocked... (2000 500 1000 1000)
Process: 0 registered... (2000 500 1000 1000)
Process: 0 I/O blocked... (2000 500 1500 1500)
Process: 1 registered... (2000 500 1000 1000)
Process: 1 I/O blocked... (2000 500 1500 1500)
Process: 0 registered... (2000 500 1500 1500)
Process: 0 completed... (2000 500 2000 2000)
Process: 1 registered... (2000 500 1500 1500)
Process: 1 completed... (2000 500 2000 2000)
```

Five processes

Summary-Results:

```
Scheduling Type: Batch (Nonpreemptive)
Scheduling Name: First-Come First-Served
Simulation Run Time: 10000
Mean: 2000
Standard Deviation: 0
Process #       CPU Time        IO Blocking     CPU Completed   CPU Blocked
0               2000 (ms)       500 (ms)        2000 (ms)       3 times
1               2000 (ms)       500 (ms)        2000 (ms)       3 times
2               2000 (ms)       500 (ms)        2000 (ms)       3 times
3               2000 (ms)       500 (ms)        2000 (ms)       3 times
4               2000 (ms)       500 (ms)        2000 (ms)       3 times
```

Summary-Processes:

Process: 0 registered... (2000 500 0 0)

Process: 0 I/O blocked... (2000 500 500 500)

Process: 1 registered... (2000 500 0 0)

Process: 1 I/O blocked... (2000 500 500 500)

Process: 0 registered... (2000 500 500 500)

Process: 0 I/O blocked... (2000 500 1000 1000)

Process: 1 registered... (2000 500 500 500)

Process: 1 I/O blocked... (2000 500 1000 1000)

Process: 0 registered... (2000 500 1000 1000)

Process: 0 I/O blocked... (2000 500 1500 1500)

Process: 1 registered... (2000 500 1000 1000)

Process: 1 I/O blocked... (2000 500 1500 1500)

Process: 0 registered... (2000 500 1500 1500)

Process: 0 completed... (2000 500 2000 2000)

Process: 1 registered... (2000 500 1500 1500)

Process: 1 completed... (2000 500 2000 2000)

Process: 2 registered... (2000 500 0 0)

Process: 2 I/O blocked... (2000 500 500 500)

Process: 3 registered... (2000 500 0 0)

Process: 3 I/O blocked... (2000 500 500 500)

Process: 2 registered... (2000 500 500 500)

Process: 2 I/O blocked... (2000 500 1000 1000)

Process: 3 registered... (2000 500 500 500)

Process: 3 I/O blocked... (2000 500 1000 1000)

Process: 2 registered... (2000 500 1000 1000)

Process: 2 I/O blocked... (2000 500 1500 1500)

Process: 3 registered... (2000 500 1000 1000)

Process: 3 I/O blocked... (2000 500 1500 1500)

Process: 2 registered... (2000 500 1500 1500)

Process: 2 completed... (2000 500 2000 2000)

Process: 3 registered... (2000 500 1500 1500)

Process: 3 completed... (2000 500 2000 2000)

Process: 4 registered... (2000 500 0 0)

Process: 4 I/O blocked... (2000 500 500 500)

Process: 4 registered... (2000 500 500 500)

Process: 4 I/O blocked... (2000 500 1000 1000)

Process: 4 registered... (2000 500 1000 1000)

Process: 4 I/O blocked... (2000 500 1500 1500)

Process: 4 registered... (2000 500 1500 1500)

Ten processes

Summary-Results

Scheduling Type: Batch (Nonpreemptive)
Scheduling Name: First-Come First-Served
Simulation Run Time: 10000
Mean: 2000
Standard Deviation: 0

Process #	CPU Time	IO Blocking	CPU Completed	CPU Blocked
0	2000 (ms)	500 (ms)	2000 (ms)	3 times
1	2000 (ms)	500 (ms)	2000 (ms)	3 times
2	2000 (ms)	500 (ms)	2000 (ms)	3 times
3	2000 (ms)	500 (ms)	2000 (ms)	3 times
4	2000 (ms)	500 (ms)	1000 (ms)	2 times
5	2000 (ms)	500 (ms)	1000 (ms)	1 times
6	2000 (ms)	500 (ms)	0 (ms)	0 times
7	2000 (ms)	500 (ms)	0 (ms)	0 times
8	2000 (ms)	500 (ms)	0 (ms)	0 times
9	2000 (ms)	500 (ms)	0 (ms)	0 times

Summary-Processes

Process: 0 registered... (2000 500 0 0)

Process: 0 I/O blocked... (2000 500 500 500)

Process: 1 registered... (2000 500 0 0)

Process: 1 I/O blocked... (2000 500 500 500)

Process: 0 registered... (2000 500 500 500)

Process: 0 I/O blocked... (2000 500 1000 1000)

Process: 1 registered... (2000 500 500 500)

Process: 1 I/O blocked... (2000 500 1000 1000)

Process: 0 registered... (2000 500 1000 1000)

Process: 0 I/O blocked... (2000 500 1500 1500)

Process: 1 registered... (2000 500 1000 1000)

Process: 1 I/O blocked... (2000 500 1500 1500)

Process: 0 registered... (2000 500 1500 1500)

Process: 0 completed... (2000 500 2000 2000)

Process: 1 registered... (2000 500 1500 1500)

Process: 1 completed... (2000 500 2000 2000)

Process: 2 registered... (2000 500 0 0)

Process: 2 I/O blocked... (2000 500 500 500)

Process: 3 registered... (2000 500 0 0)

Process: 3 I/O blocked... (2000 500 500 500)

Process: 2 registered... (2000 500 500 500)

Process: 2 I/O blocked... (2000 500 1000 1000)

Process: 3 registered... (2000 500 500 500)

Process: 3 I/O blocked... (2000 500 1000 1000)

Process: 2 registered... (2000 500 1000 1000)

Process: 2 I/O blocked... (2000 500 1500 1500)

Process: 3 registered... (2000 500 1000 1000)

Process: 3 I/O blocked... (2000 500 1500 1500)

Process: 2 registered... (2000 500 1500 1500)

Process: 2 completed... (2000 500 2000 2000)

Process: 3 registered... (2000 500 1500 1500)

Process: 3 completed... (2000 500 2000 2000)

Process: 4 registered... (2000 500 0 0)

Process: 4 I/O blocked... (2000 500 500 500)

Process: 5 registered... (2000 500 0 0)

Process: 5 I/O blocked... (2000 500 500 500)

Process: 4 registered... (2000 500 500 500)

Process: 4 I/O blocked... (2000 500 1000 1000)

Process: 5 registered... (2000 500 500 500)

Observations:

From the results we can see that the CPU runs processes that start from 0,1,2… and so on. This is because of the "First-come First-Served" scheduling algorithm used in this simulation. We can also notice that when the 0^{th} process is registered it runs for 500ms before it is blocks for the I/O operation at the time when its blocked it is sent to the wait/block state and it is still in the main memory. Once it is sent to wait/block state it will be sent again to ready state and wait for its turn to execute. So the next process, "process 1" which waits in the ready state before this new process, is registered and it runs for 500ms. When this process is blocked after 500ms the 0^{th} process which has been blocked before is again brought back and it is executed for 500 more milliseconds. This repeats until the completion of these two processes. After this the next two processes that waits in the queue follows the same steps and gets executed.

We can also observe the priority of the processes, when 0th process waits for I/O operation the higher priority goes to the 1^{st} process not the subsequent process. Also when 1^{st} process waits, the first priority goes back again to the 0^{th} process not the 2^{nd} process. This "priority scheduling" takes place in this way because of the "First-come First-served" scheduling algorithm. No matter how much time each process takes to complete the execution, in "First-come First-Served" algorithm the process which waits first will get the first chance to execute. But, what happens if the process takes so much time but there are other processes waiting in the queue which takes really less time for complete execution. There is also another scheduling algorithm called "Shortest-Job-First Scheduling". In this algorithm the process will not be checked for the first time of execution. It will be checked for the next time when the same process uses the CPU, how much time it will take to finish execution. So the process which takes less time to complete execution on its next CPU burst is given the first priority. If two processes have same time of execution then "First-come First-served" algorithm is used.

We can notice that with ten processes running at the same time there is not enough time to complete all the processes which are ready to be executed, since the total simulation time is set to 10000ms. When tried running with five processes it executed similarly to the two processes but after completing four processes by switching back and forth the fifth process ran continuously since there are no more processes waiting to be executed by the CPU.

From the summary results we can clearly observe if a process completed its execution. For two and four processes we see that the CPU completed 2000ms successfully for each process and it blocked for I/O operation 3 times. In ten processes we can see that the process 5 and process 6 completed only 1000ms(executed only for 1000ms) and the rest of the process did not even execute. This is because of the total simulation time; it is not enough for all the process to complete executing. When the simulation time ends all the processes are stopped.