EOPSY Lab 3

Shailesh Kumar 289008

*Introduction*

## Goal

The task for us is to compare different number of processes (2,5 and 10) in a schedule.conf file and explain the phenomena.

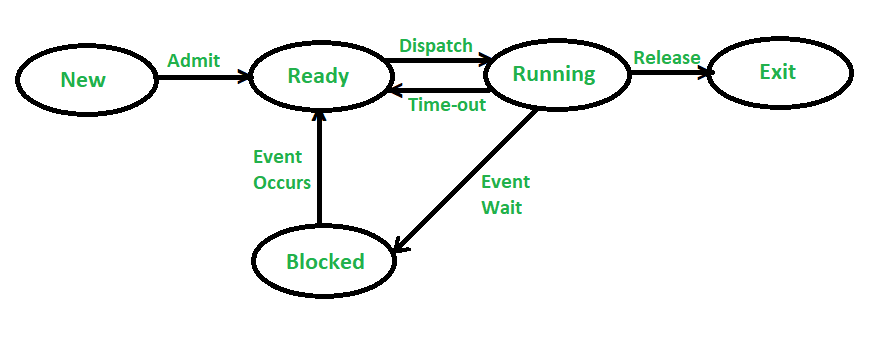
The system seems to work with one process at a time and the amount of time taken by execution can be handled. (2000 milliseconds in our case)

There is an I/O blocking variable that we can fiddle with, it tells you the amount of time one process will be blocked for. It also tells you after how long you will block the input. (both are the same value)

so for 500 ms, it will run for 500ms and then wait for 500 ms. For 100ms, it will run for 100ms and then wait for 100ms.

The system is first come and first serve, so the moment the previous process is halted, the next process takes it place, but if the previous process becomes ready again, when the current process is halted/completed, the waiting process is given priority even if other processes are ready.

## Five states of a process

This is a photo of 5 state model that a process follows:

New – a new process, it can be admitted and then goes to ready state.

Ready - In our scenario, where there can not be more than one process running in parallel. The process will stay in ready state until the running gets a time-out or finishes the execution.

Running – self-explanatory, it is currently running a process and can be timed-out or go to blocked which we do by I/O blocking or finish the execution.

Blocked – In this task, we go to blocked, if a process is waiting for a specific event, in our case, not allowed to start again until a specific amount of time has passed (500ms), once that is over, the process goes to ready state.

Exit – After finishing a process, or if the process was aborted, it exits. This is the final state of a process.

## Scheduling algorithms

There are many types of scheduling algorithms

* Shortest-Job-Next (SJN) Scheduling
* First-Come, First-Served (FCFS) Scheduling
* Priority Scheduling
* Shortest Remaining Time

SJN – This takes the approach of taking the least amount of waiting time. If we know the amount of time a process will take, this is the best approach.

FCFS – This is the one that is used in our task. It is easy to understand and uses the FIFO algorithm. Whatever came first to ready state would be scheduled first.

Priority Scheduling – Each process can be given a priority and if a priority is higher, the process is handled first.

Shortest remaining time – In this algorithm, the processor is allocated a job that is closest to completion.

Reference: <https://www.tutorialspoint.com/operating_system/os_process_scheduling_algorithms.htm>

## Preemptive and non-preemptive Scheduling

Preemptive scheduling means that the cpu gives a process a set amount of time to run and makes the process jump to waiting state.

Non-preemptive scheduling means that the process will hold the cpu until the process gets terminated or reaches a waiting state.

The one that we are doing in our task is non-preemptive.

Reference: <https://www.geeksforgeeks.org/preemptive-and-non-preemptive-scheduling/>

## Reading Summary

**Scheduling type** is nonpreemptive and the algorithm is FCFS. (read above for definitions)

**Simulation run time** gives the run time of the program.

**Mean** gives the average amount of time it takes for a process to burst.

**Standard Deviation** means, the amount of deviation is allowed for an average process to burst.

**Process #** - process number

**CPU Time** – The CPU time taken

**IO Blocking** – amount of time after a process goes to waiting and stays there.

**CPU completed** – the CPU time taken to complete the processes

**CPU blocked** – Amount of time the process jumped to waiting before being terminated

## Reading Summary

**Process** → The process number  
**Registered** - > process jumped from ready to running.  
**I/O Blocked** → process jumped from running to waiting  
**Completed** → process jumped from running to exit

(#1 #2 #3 #4)  
**#1 Mean** – The amount of time taken for process to complete.

**#2 I/O Blocking** - amount of time after a process goes to waiting and stays there.

**#3 and #4 Current time** – Current time in running state.

*Case 1: Two processes.*

First situation gives a performance of running two processes: 0 and 1.

Process 0 is being admitted first and runs until it goes to “blocked” state because I/O interrupt occurred. Then OS will serve another process (Process 1) onto the CPU to be run. Since running is free for now.

This process is executing its job, while in the meantime Process 0 goes to the ready state and expects to be dispatched.

It will have to wait until process 1 exits the running state.

Process 1 finally (after each 500ms) has its own I/O blocking and switches into “blocked” state. This allows OS to restore Process 0 and continue its execution.  
The task of a single process takes 2000 ms. Both task will manage to execute before runtime limit will be reached (which equals to 10000ms).

**Summary:**

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

**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)

*Case 2: Five processes*

This situation is extension of the previous example. Execution time for all processes is equal to the runtime limit. This means that all processes should be executed in without exceeding the limit.

It is worth to note that running or expecting to be dispatched (ready) processes will block spawning of new processes. This means that when Process 0 and Process 1 are created and after some time Process 0 is in ready state and Process 1 is executing, none of the the processes can be spawned.

Then we see that processes will be handled in pairs (0 and 1; 2 and 3) with one exception – Process 4 which does not have its pair, so it will be handled alone. After its waiting-ready cycle, OS will serve it onto the CPU again as there is no other process to be handled.

**Summary:**

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

**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)

*Case 3: Ten processes*

This is another evolution, this time of case 2. The amount of processes exceeds given runtime limit by 100%. It means that half of the processes won’t be handled. Simulation confirms this statement.

We see that last handled pair will be Process 4 and Process 5. Here, however, Process 4 will be interrupted twice and Process 5 will be interrupted only once (compared to other pairs where very process was interrupted 3 times). At this point program has no more time to use, thus it is getting terminated.

**Summary:**

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

**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)