Operating Systems (EOPSY)

Laboratory no. 3

Scheduling

Laura Ploch

May 2021

1. Introduction

The aim of this laboratory was to get familiar with CPU scheduling by using scheduling simulator, which illustrates the behavior of scheduling algorithms against a simulated mix of process loads. In the configuration file we could modify a couple of different parameters which influenced the final output.

2. Theoretical part

2.1 CPU scheduling

CPU scheduling is a method which allows for having a computer multitasking with a single central processing unit (CPU). It is based on assigning work (processes, data flows etc.) to resources responsible for completing the work (hardware resources such as processors, network links, expansion cards (sound cards, storage cards etc.)). CPU scheduling aims for making the system fast and efficient.

2.2 Scheduling types

The type of CPU scheduling depends on which circumstances the decisions take place under. If the process switches from *running* state to *waiting* state or when the process terminates – the scheduling scheme is said to be **non-preemptive.**

In case an interrupt occurs (so when a process switches from *running* state to *ready* state) or when I/O operation is completed (process switches from *waiting* to *ready* state) then the scheduling type is **preemptive.** In this scheduling type the resources are allocated to the process depending on the process's priority. The resources are also allocated for limited amount of time after which the process is placed back in the queue.

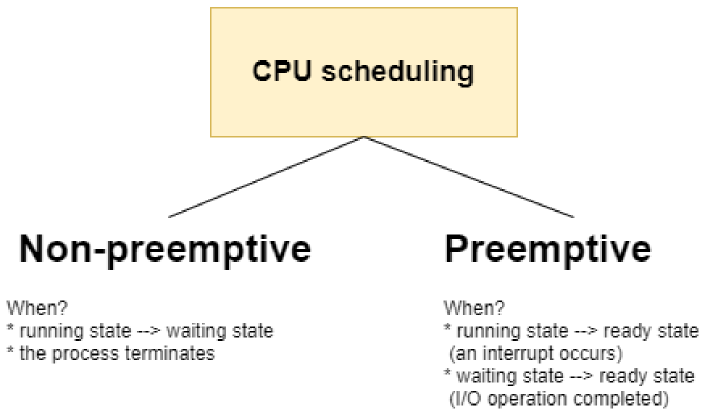

CPU scheduling
Non-preemptive
When?
* running state --> waiting state
* the process terminates
Preemptive
When?
* running state --> ready state
(an interrupt occurs)
* waiting state --> ready state
(I/O operation completed)

Fig. 1 Scheduling types. Source: own elaboration

2.3 Our scheduling type – non-preemptive

In the case of our simulation is non-preemptive. The CPU is allocated to a process for a specified period of time (*CPU time* calculated from standard deviation and mean deviation values set in configuration file – in our case the resulting CPU time is 2000 ms) after which the process switches from *running* to *ready* state and the CPU is allocated to another process. The interchanging of CPU allocation between processes happens in pairs and continues until the two processes terminate.

2.4 Scheduling Algorithm - First Come First Serve

Among six possible scheduling algorithms, the one our scheduler works with is FCFS – "First Come First Serve". In this algorithm the process which requests the CPU first is executed first. When the CPU is freed due to a block or process terminating, it is assigned to the next process from the beginning of the queue.

The FCFS is the reason why the processes are executed in pairs. After the first process has been blocked, the next one in the queue gets the CPU allocation. Then it is blocked too and the previously blocked process gets the CPU allocation, since it has not been terminated yet.

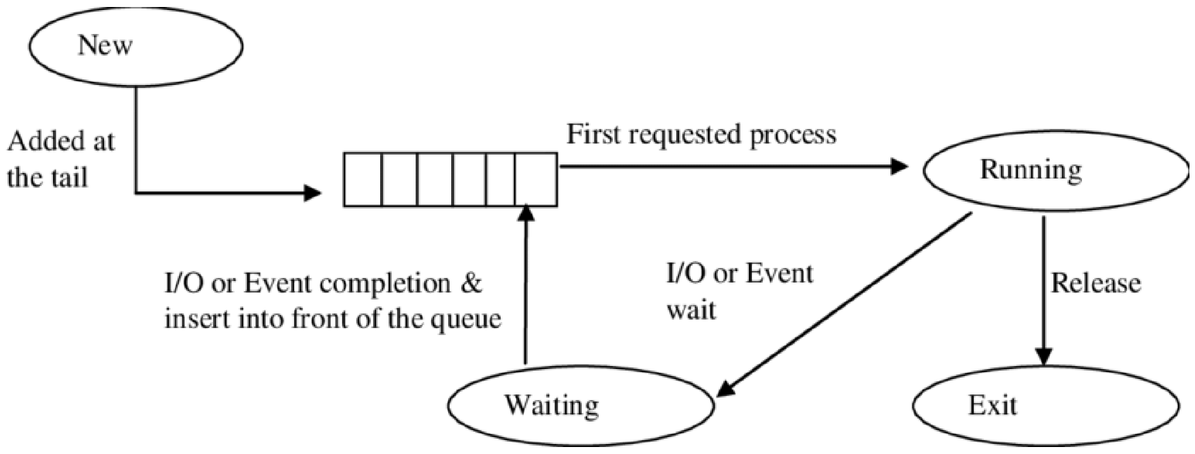


Fig. 2. First Come First Serve algorithm in non-preemptive scheduling

source: https://www.researchgate.net/figure/First-Come-First-Serve-Scheduling-Characteristics-The-lack-of-prioritization-does_fig1_249645533

3. The Task

We were to modify the configuration file in such a way that all processes run an average of 2000 ms with a standard deviation of 0 (the number of standard deviations from the average length of time a process should execute before terminating).

The processes were to be blocked for input or output every 500 ms and the whole simulation was to be run for 10 000 ms.

The simulation was to be repeated in such configuration for 2, 5 and 10 processes.

3.1 Number of processes run: 2

Configuration file

```
// # of Process
numprocess 2

// mean deviation
meandev 2000

// standard deviation
standdev 0

// process    # I/O blocking
process 500
process 500

// duration of the simulation in milliseconds
runtime 10000
```

Summary-results file

```
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 file:

```
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)
```

3.2 Number of processes run: 5

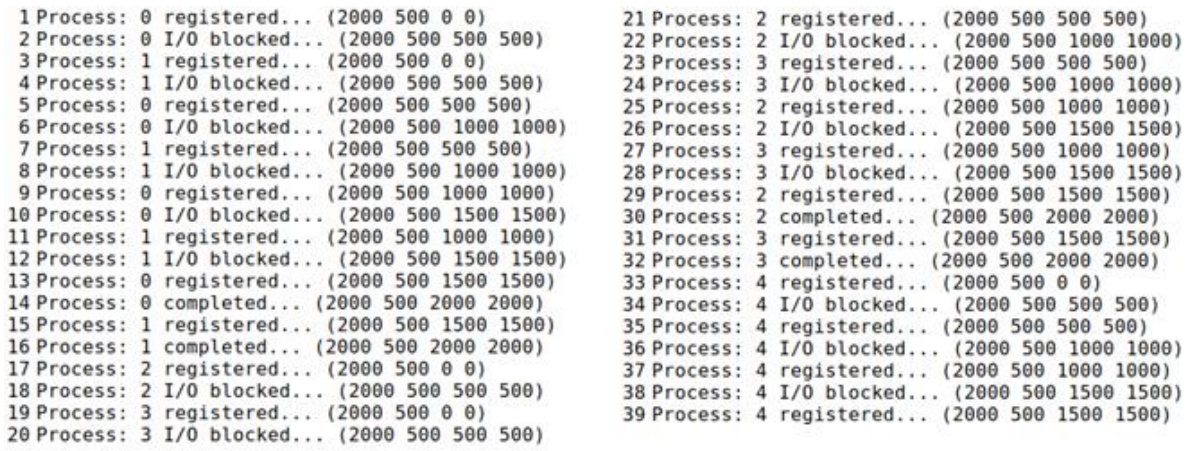
Configuration file:

```
// # of Process
numprocess 5

// mean deviation
meandev 2000

// standard deviation
standdev 0

// process    # I/O blocking
process 500
process 500
process 500
process 500
process 500

// duration of the simulation in milliseconds
runtime 10000
```

Summary-results file

```
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 file:



```
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)
```

3.3 Number of processes run: 10

Configuration file:

```
// # of Process
numprocess 10

// mean deviation
meandev 2000

// standard deviation
standdev 0

// process    # I/O blocking
process 500
process 500
process 500
process 500
process 500
process 500
process 500
process 500
process 500
process 500

// duration of the simulation in milliseconds
runtime 10000
```

Summary-results file:

```
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 file:

```
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)
```

4. Observations

4.1 Two processes run

The simulation run time was equal to 4000 ms which results from running 2 processes, each one for 2000 ms. Each process is blocked after every 500 ms, which gives us a total of 3 blocks during one process' runtime (after 500, 1000 and 1500 ms of running).

4.2 Five processes run

We can see that all processes run in pairs as explained, except the fifth one (process no. 4) since it was the last one left. In the *summary-processes* file we can't see the line informing about fifth process being completed, but we can read in *summary-results* that all processes have runtime equal to the CPU time needed for terminating, that is 2000 ms – so all processes have been completed (the simulator just didn't have enough time to write the last information in the file)**.**

4.3 Ten processes run

We can see that only 6 processes had been run before the simulation reached the runtime of 10 000 ms. Out of those 6, only 4 managed to be completed.. The last two processes (no. 4 (5th process) and no. 5 (6th process)) only managed to be run for 1000 ms - in the last 2000 the 5th process had been blocked twice and the 6th process had been blocked only once before the runtime ended. The 6th process didn't manage to be blocked twice, because the block happens after 500ms, if we provide 10 001 ms runtime, the scheduler manages to block the 6th process twice.

Resources:

- https://en.wikipedia.org/wiki/Scheduling_(computing)#Types_of_operating_system_schedulers
- https://www.studytonight.com/operating-system/first-come-first-serve
- https://www.guru99.com/cpu-scheduling-algorithms.html#8
- https://www.geeksforgeeks.org/preemptive-and-non-preemptive-scheduling/