EOPSY
Lab 3: Scheduling

Julia Czmut 300168

Introduction

Process scheduling creates a possibility of efficient usage of the CPU – multiple processes can share the CPU time. It's an important mechanism in multi programming which ensures that the system is efficient, fast and fair.
Whenever the CPU becomes idle, the OS selects one of the other processes in queue and starts executing it.

If the CPU scheduling decisions occur:
- when a process switches from state running to ready,
- when a process terminates,
then the scheduling scheme is called **non-preemptive** (there is no choice in terms of scheduling – if another process exists in queue, it must be chosen).

If apart from the two situations, the scheduling decisions also occur:
- when a process switches from state running to ready (for example, in case of an interrupt),
- when a process switches from state waiting to ready (for example, in case of completion of I/O),
then the scheduling scheme is called **preemptive** (the processes can have different priorities set).

The goal was to run some processes and analyze the results. The processes were to be configured to run an average of 2000 miliseconds with a standard deviation of zero, be blocked for input or output every 500 miliseconds and with runtime of 10000 miliseconds.

There are three simulations: first with 2 processes, second with 5 and third with 10 processes.

SIMULATION 1

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
```

Output from 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
```

Output from 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)
```

Comments

As can be seen in Summary-Results output, both the processes took 2000 ms to finish. Summed up they took 4000 ms, so less than the limit of 10000 ms. Moreover, we can see the configured values there, for example stating that “IO blocking” occurs every 500 ms, CPU was blocked 3 times for each process and that the CPU took 2000 ms to finish.

In Summary-Processes we can see the steps of the simulation. First the process 0 is registered, then as we stated by the configuration it gets blocked after 500ms. Then the process 1 is registered and is also blocked and gets in a queue after process 0. After that the process 0 is again registered, blocked and the process 1 is registered and then blocked. This cycle repeats 4 times, so until the CPU reaches the run time of 2000ms and is able to complete (so in the 4^{th} cycle instead of IO blocking the processes are completed). When that time comes, the process 0 is registered and completed first, because our simulation is scheduled in First-Come First-Served mode and process 0 came before process 1. Then it’s process 1’s turn and the CPU is able to accept it and complete.

The scheduling type in our simulations is the **non-preemptive batch scheduling**. In non-preemptive scheduling, once a process is taken on by the CPU, the process keeps the CPU to itself until releasing it by either termination or switching to waiting state. It is clearly visible in our Summary-Processes output – no other process is started until the current one is either “IO blocked” (so kind of stalled and waiting) or completed.

We can also talk more about the scheduling algorithm used in our simulations – **First-Come First Served**. In this algorithm, the process which requests the CPU first, gets the CPU to itself first. It is like the FIFO (First In First Out) and it is very easy to implement and understand. However, because it is a non-preemptive algorithm and the process priorities do not matter, there can be problems in case of more important and less important processes. If a low priority long process is first, it will be executed first, even if next in queue is a highly important task – it will have to wait, and in turn, the system may crash. Another problem in FCFS is the Convoy Effect which leads to inefficient resource utilization. The Convoy Effect is a situation when multiple processes which need the CPU for short amounts of time, have to wait for it, because another process is blocking it for a long time.
In general, a good scheduling algorithm has to be fair, reliable (stated rules do actually apply) and balanced (all parts of the system are busy).
Now more specifically, in batch systems, important features are:
- throughput (max. number of jobs per hour)
- turnaround time (min. time between submission and termination)
- CPU utilization (CPU is busy all the time)

Throughput would depend on how long the jobs were. In our case, all of them had the same duration, but it could have been worse, especially considering the algorithm takes the processes from the queue “blindly”, whichever comes first.
Turnaround time also depends on the specific jobs, however FCSF is not an efficient algorithm and average waiting times are high, so being a part of turnaround time – it also influences the turnaround time to be higher than it could be.
As for the CPU utilization, the CPU is in fact always busy, because the processes are holding them occupied until termination or changing state to waiting.

SIMULATION 2

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
```

Output from 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
```

Output from 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)
```

Comments

Simulation 2 was handling 5 processes. All of them still had the runtime of 2000 ms, so because the limit was 10000 ms, the last process did not have enough time to signal its completion.
The processes 0 and 1 performed the same set of steps as in Simulation 1. After they completed, the same situation occurred for processes 2 and 3. The pattern of those pairs of processes in short consists of: first process is registered and then blocked, second process is registered and then blocked (3 times), then first process is registered and completed, and finally the second process is registered and completed.
At this point, there was exactly 2000 ms left until the limit of 10000 ms was reached. Process 4 was registered, then blocked after 500 ms 3 times and did not have enough time to signal its completion, because the time limit was reached.

SIMULATION 3

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
```

Output from 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
```

Output from 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)
```

Comments

In this simulation, the first 8000ms is performed the same as in Simulation 2, so the pairs of processes: 0 and 1, 2 and 3 had their runtimes equal to 2000ms per process. The processes 4 and 5 were executed for 1000ms each, and the rest of the processes did not execute at all.

Now more specifically, after processes 0-3 are completed, we are 8000ms into the simulation time.
- (8000ms) process 4 is registered
- (8500ms) process 4 is blocked and process 5 is registered
- (9000ms) process 5 is blocked and process 4 is registered
- (9500ms) process 4 is blocked and process 5 is registered

Then the simulation ends, because the time limit was reached and process 5 did not get blocked.

It can also be seen in Summary-Results. Processes 0-3 had runtimes equal to 2000ms and were blocked 3 times each. Processes 4 and 5 took 1000ms each, process 4 was blocked 2 times, and process 5 was blocked only once.