EOPSY - Task 3

Burak Kaymakci - 317054

Contents

1 Scheduling

Scheduling is a method that is used to distribute valuable computing resources, usually processor time, bandwidth and memory, to the various processes, threads, data flows and applications that need them. Scheduling is done to balance the load on the system and ensure equal distribution of resources and give some prioritization according to set rules. This ensures that a computer system is able to serve all requests and achieve a certain quality of service.

Scheduling is also known as process scheduling. Scheduling in a system is done by the aptly named scheduler, which is mainly concerned with three things:

- Throughput, or how fast it can finish a certain number of tasks from beginning to end per unit of time
- Latency, which is the turnaround time or the time it takes to finish the task from the time of request or submission until finish, which includes the waiting time before it could be served
- Response time, which is the time it takes for the process or request to be served, in short the waiting time

Scheduling is largely based on the factors mentioned above and varies depending on the system and the programming of the system's or user's preferences and objectives. In modern computers such as PCs with large amounts of processing power and other resources and with the ability to multitask by running multiple

threads or pipelines at once, scheduling is no longer a big issue and most times processes and applications are given free reign with extra resources, but the scheduler is still hard at work managing requests.

1.1 Preemptive Scheduling

Preemptive scheduling is used when a process switches from running state to ready state or from waiting state to ready state. The resources (mainly CPU cycles) are allocated to the process for the limited amount of time and then is taken away, and the process is again placed back in the ready queue if that process still has CPU burst time remaining. That process stays in ready queue till it gets next chance to execute.

Algorithms based on preemptive scheduling are: Round Robin (RR),Shortest Remaining Time First (SRTF), Priority (preemptive version), etc.

1.2 Non-preemptive Scheduling

Non-preemptive Scheduling is used when a process terminates, or a process switches from running to waiting state. In this scheduling, once the resources (CPU cycles) is allocated to a process, the process holds the CPU till it gets terminated or it reaches a waiting state. In case of non-preemptive scheduling does not interrupt a process running CPU in middle of the execution. Instead, it waits till the process complete its CPU burst time and then it can allocate the CPU to another process.

Algorithms based on non-preemptive scheduling are: Shortest Job First (SJF basically non preemptive) and Priority (non preemptive version), etc.

1.3 FCFS (First Come First Serve) Scheduling Algorithm

This is the default algorithm used in the simulations. First Come First Serve (FCFS) is an operating system scheduling algorithm that automatically executes queued requests and processes in order of their arrival. It is the easiest and simplest CPU scheduling algorithm. In this type of algorithm, processes which requests the CPU first get the CPU allocation first. This is managed with a FIFO queue. The full form of FCFS is First Come First Serve.

As the process enters the ready queue, its PCB (Process Control Block) is linked with the tail of the queue and, when the CPU becomes free, it should be assigned to the process at the beginning of the queue.

Characteristics of FCFS method

- It supports non-preemptive and pre-emptive scheduling algorithm.
- Jobs are always executed on a first-come, first-serve basis.
- It is easy to implement and use.
- This method is poor in performance, and the general wait time is quite high.

1.3.1 Advantages of FCFS

Here, are pros/benefits of using FCFS scheduling algorithm:

- The simplest form of a CPU scheduling algorithm
- Easy to program
- First come first served

1.3.2 Disadvantages of FCFS

Here, are cons/ drawbacks of using FCFS scheduling algorithm:

- It is a Non-Preemptive CPU scheduling algorithm, so after the process has been allocated to the CPU, it will never release the CPU until it finishes executing.
- The Average Waiting Time is high.
- Short processes that are at the back of the queue have to wait for the long process at the front to finish.

- Not an ideal technique for time-sharing systems.
- Because of its simplicity, FCFS is not very efficient.

2 Task Description

Create a configuration file in which all processes run an average of 2000 milliseconds with a standard deviation of zero, and which are blocked for input or output every 500 milliseconds. Run the simulation for 10000 milliseconds with 2 processes. Examine the two output files. Try again for 5 processes. Try again for 10 processes. Explain what's happening.

3 2-Process Case

3.1 Configuration File

```
// # of Process
numprocess 2

// mean deivation
meandev 2000

// standard deviation
standdev 0

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

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

3.2 Simulation Results

3.2.1 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)
```

3.2.2 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
```

3.3 Conclusions

As we can see from the results, the simulation for two processes took four seconds to complete as each process was only running for two seconds. Each process blocks for I/O after 500 ms burst time, meaning that they enter the blocked or waiting state. The process continues to wait in the main memory and does not require CPU. Once the I/O operation is completed the process goes to the ready state. When a process is in the waiting state the CPU can start processing the next process in the queue, thus, when `Process 0` blocks for I/O, `Process 1` gets registered and starts using CPU resource for 500 ms until it blocks for I/O as well. After `Process 1` blocks for I/O and enters the waiting state, since we only have 2 processes, the CPU starts processing `Process 0` again and so on and so forth until either all the processes gets completed or the defined simulation time (10 seconds in this case) runs out. In 2 process case, we didn't run out of simulation time so our 2 processes were completed successfully.

In this example each process blocked for I/O 3 times because by our configuration file, every process blocks after 500 ms of burst time and every process has 2 seconds of burst time in total. If we do the math we can see that every process will always block for 3 times as long as we have enough simulation time.

4 5-Process Case

4.1 Configuration File

```
// # of Process
numprocess 5

// mean deivation
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
```

4.2 Simulation Results

4.2.1 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)
```

4.2.2 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
```

4.3 Conclusions

In the 5-Process case, we can see that all the processes are successfully completed, however, in the `Summary-Processes` file we do not see a message for Process 4 that it's been completed. That's probably because since the total simulation time defined in the configuration file is 10 seconds and it also takes 10 seconds to process all the 5 processes. The application might have exited before logging that Process 4 was successfully completed. We can be sure that all the processes were successfully completed by taking a look at the `Summary-Results` file. There we see all the processes were run for 2 seconds and all of them blocked for I/O 3 times before they were completed.

5 10-Process Case

5.1 Configuration File

```
// # of Process
numprocess 10

// mean deivation
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
```

5.2 Simulation Results

5.2.1 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)
```

5.2.2 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
```

5.3 Conclusions

The 10-Process case is a special one, because in this case not all the processes get completed. Moreover the simulation ends even before the last 4 processes get a chance to be processed or registered. In this case we can see that first 4 processes (0-3) were successfully completed, each blocking 3 times for I/O and processes 4 and 5 are only run for 1 second. After that we run out of total simulation time and the simulation exits.