# 1 OPERATING SYSTEMS LABORATORY VII - Scheduling

## 1.1 MOSS Simulator

MOSS (Modern Operating Systems Simulators) is a collection of Java-based simulation programs which illustrate key concepts presented in the text Andrew S. Tanenbaum, Modern Operating System, Second Edition (Prentice-Hall, 2001). The software is designed for students and instructors using this text.

- Read the file index.html carefully.
- Each simulator is packaged separately and available for
  - Scheduling Simulator
  - Deadlocking Simulator
  - Memory Management Simulator
  - File System Simulator
- Study the user guide for Scheduling Simulator
- Follow the steps below for installation of the software
  1. Create a directory in which you wish to install the simulator (e.g., "moss/sched").

     ```
     $ mkdir moss
     $ cd moss
     $ mkdir sched
     $ cd sched
     ```

  2. Download the compressed tar archive ( sched.tgz) into the directory.
  3. Expand the compressed tar archive.

     ```
     $ tar -zxvf sched.tgz
     ```

  4. Export the classpath

     ```
     $ export CLASSPATH=.
     ```

  5. To test the program, enter the following commands.

     ```
     $ java Scheduling scheduling.conf
     ```

The program will simply run the simulation based on the information provided in *scheduling.conf* and write its output to the *Summary-Results* and *Summary-Processes* files. You should see the following output.

```
Working...
Completed.
```

6. Hint: If you want to compile java codes as given in the manual files, you may be faced with some errors. A suggested solution is that: If there is any import statement that is used for including user-defined class, erase it. (i.e. //import Common;)

Lab work:

1. 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.

2. Consider changing the configuration parameter "meandev" to "run_time_average". The word "dev" doesn't belong here. It would be nice if this were the average amount of time a process runs before blocking for input or output instead of the total runtime.

3. Consider changing the configuration parameter "standdev" to "run_time_stddev". It would be nice if this were the number of standard deviations from the average time a process runs before blocking for input or output instead of the total runtime.

## 1.2 Assignment III

1. Design a cpu scheduler of your own choice. The idea is to schedule the CPU for several simulated processes that alternately compute and request I/O.

   - You are to read a file describing **n** processes and then simulate the **n** processes until they all terminate. The way to do this is to keep track of the state of each process and advance time making any state transitions needed. The data file has the following format:

| Process Arrival Time | CPU burst | I/O Operation | ... | CPU burst | Line End |
|---|---|---|---|---|---|
| 0 | 100 | 2 | ... | 200 | -1 |
| 5 | 6 | 4 | ... | 25 | -1 |
| ⋮ | ⋮ | ⋮ | ... | ⋮ | -1 |

- Assume that there is only one CPU and one I/O device in the system. The I/O device serves only one process at a time.
- Recall that in a real operating system, the scheduler only executes whenever an interrupt or trap occurs. Similarly, your simulation should be event based.
- Each event should be emitted along with relevant process states. For example:

```
 ...
Time : 13
- process 4 io started (blocked until 15, remaining = 3, burst = 2)
- process 5 running    (remaining = 12, burst = 10)

Time : 15
 - process 4 i/o completed (now ready, remaining=3, burst=2)
 - process 5 preempted (remaining = 10, burst = 12)
 - process 4 running   (remaining = 3, burst = 2)
...
```

  (preemption depends on the chosen scheduling algorithm.)
- Ignore the context switching overhead and i/o latency.
- At the end of the run you should first print an identification of the run including the scheduling algorithm used, any parameters (e.g. the quantum for RR), and the number of processes simulated. You should then analyze the behavior of your scheduler and report the followings;
  - Finishing time (i.e., when all the processes have finished).
  - CPU Utilization (i.e., percentage of time some job is running).
  - I/O Utilization (i.e., percentage of time some job is blocked).
  - Throughput, expressed in processes completed per hundred time units.
  - Average turnaround time.
  - Average waiting time.

An example process data file

```
0 5 -1
1 3 2 3 2 3 -1
6 10 2 60 2 30 3 70 2 10 2 10 -1
23 3 2 3 2 3 -1
24 70 2 70 2 40 3 70 2 20 2 10 -1
25 3 2 3 2 3 -1
26 80 2 80 2 50 3 70 2 40 2 10 -1
27 3 2 3 2 3 -1
28 25 2 10 -1
29 3 2 3 2 3 -1
31 3 2 3 2 3 -1
33 3 2 3 2 3 -1
35 3 2 3 2 3 -1
40 3 2 3 2 3 -1
40 3 2 3 2 3 -1
42 3 2 3 2 3 -1
43 3 2 3 2 3 -1
45 3 2 3 2 3 -1
```