MOSS Scheduling Simulator
Installation on Unix/Linux/Solaris/HP-UX Systems

Purpose

This document provides instructions for the installation of the MOSS Scheduling Simulator on Unix operating systems. This procedure should be the same or similar on Unix, Linux, Solaris, HP-UX and other Unix-compatible systems. The MOSS software is designed for use with Andrew S. Tanenbaum, Modern Operating Systems, 2nd Edition (Prentice Hall, 2001). The Scheduling Simulator was written by Alex Reeder (*alexr@e-sa.org*). This installation guide was written by Ray Ontko (*rayo@ontko.com*).

Requirements

The following software components are required to install and use the MOSS Scheduling Simulator.

- X-windows environment for running Java Application Window Toolkit (AWT) programs
- Java Development Kit (JDK) 1.0 or greater
- Text program editor (e.g., notepad)

Pre-Installation

Before installation, you should verify:

- that you have a working java runtime environment,
- that you have a working java development environment, and
- that the working directory is in the classpath for the runtime environment.

If you're using a standard command-line java compiler, the following instructions will help determine if your environment is configured correctly.

1. Verify that you have java installed and configured in your environment.

   ```
   $ java -version
   ```

   You should see a message like this with possibly a different version number.

   ```
   java version "1.1.8"
   ```

   If you get a message like:

   ```
   java: Command not found.
   ```

   Then java may not be installed on your system, or may not be configured for your use.

   If you think that Java may already be installed on your system but may not be in your "path", you can find it by

   ```
   $ find /usr -name java -print
   ```

   On my system, for example, the following is returned.

   ```
   /usr/lib/netscape/477/communicator/java
   /usr/lib/netscape/477/netscape/java
   /usr/lib/jdk1.1/bin/java
   /usr/lib/jdk1.1/bin/ia32/green_threads/java
   /usr/share/java
   /usr/bin/java
   /usr/src/kernel-source-2.2.17/include/config/binfmt/java
   ```

   On my system, I also searched for "javac" and found that it exists in /usr/bin/java. I’ll use this jdk for my installation.

   If Java isn’t available on your system, you should check with your instructor or system administrator. If you administer your own system, then you should be able to find a copy of Java for your operating system.

   If you find that java is installed but not configured for your use, then perhaps you need to add it to your path. Consult your instructor or system administrator if you need help adding this to your path.

2. Verify that the java compiler is installed and configured in your environment.

   ```
   $ javac
   ```

   If you’re using a standard java command-line compiler, you should see a message similar to this.

   ```
   use: javac [-g][-O][-debug][-depend][-nowarn][-verbose][-classpath path][-nowrite][-deprecation][-d dir][-J] file.java...
   ```

If you get a message like:

```
javac: Command not found.
```

then the java compiler may not be installed on your system, or may not be configured for your use. Consult your instructor or system administrator.

3. Verify that that the current directory is in your classpath.

   ```
   $ echo $CLASSPATH
   ```

   You should see a list of directories separated by colons (":") or possibly just a blank line. If you don’t see the directory "." (a single period, which stands for the current directory), then you should add it to the claspath.

   Determine which shell you’re using:

   ```
   $ echo $SHELL
   ```

   If you’re using sh, ksh, or bash:

   ```
   $ CLASSPATH=.:$CLASSPATH
   $ export CLASSPATH
   ```

   If you’re using csh, or tcsh:

   ```
   % set CLASSPATH=.:$CLASSPATH
   ```

If you have a working java runtime environment, a working java compiler, and the current directory is in your path, then you’re ready to proceed with the installation.

Installation

Installation of the software can be accomplished with these simple steps:

1. Create a directory in which you wish to install the simulator (e.g., "moss/sched").

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

2.

Download the compressed tar archive (sched.tgz) into the directory. The latest release for this file can always be found at http://www.ontko.com/moss/sched/sched.tgz.

3. Expand the compressed tar archive.

```
$ tar -xzf sched.tgz
```

or

```
$ gunzip sched.tgz
$ tar xf sched.tar
```

Files

The directory should now contain the following files:

Files	Description
sched.tgz	Compressed tar archive which contains all the other files.
Common.java Process.java Results.java Scheduling.java SchedulingAlgorithm.java sProcess.java	Java source files (*.java)
Common.class Process.class Results.class Scheduling.class SchedulingAlgorithm.class sProcess.class	Compiled Java class files (*.class)
scheduling.conf	Sample configuration file
install_unix.html install_windows.html user_guide.html	Documentation
COPYING.TXT	Gnu General Public License: Terms and Conditions for Copying, Distribution, and Modification

Compilation

The distribution includes compiled class files as well as the source java files. You should not need to recompile unless you decide to change the code. If you wish to compile the code, the following commands should work if you’re using a Java compiler that accepts the normal "javac" command line.

To determine which shell you’re using:

```
$ echo $SHELL
```

If you’re using sh, ksh, bash:

```
$ CLASSPATH=.
$ export CLASSPATH
$ javac -nowarn *.java
```

If you’re using csh, tcsh:

```
% set CLASSPATH=.
% javac -nowarn *.java
```

The `-nowarn` flag supresses warning messges, of which there may be several. For backward compatability we use only those features of Java which have been present from the beginning, some of which are deprecated and are usually reported by the compiler with warning messages.

MOSS Scheduling Simulator User Guide

Purpose

This document is a user guide for the MOSS Scheduling Simulator. It explains how to use the simulator and describes the various input and output files used by the simulator. The MOSS software is designed for use with Andrew S. Tanenbaum, Modern Operating Systems, 2nd Edition (Prentice Hall, 2001). The Scheduling Simulator was written by Alex Reeder (*alexr@e-sa.org*). This user guide was written by Ray Ontko (*rayo@ontko.com*).

Introduction

The scheduling simulator illustrates the behavior of scheduling algorithms against a simulated mix of process loads. The user can specify the number of processes, the mean and standard deviation for compute time and I/O blocking time for each process, and the duration of the simulation. At the end of the simulation a statistical summary is presented. Students may also be asked to write their own scheduling algorithms to be used with process loads defined by the instructor.

Running the Simulator

The program reads a configuration file (`scheduling.conf`) and writes two output files (`Summary-Results` and `Summary-Processes`).

To run the program, enter the following command line.

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

The program will display "Working..." while the simulation is working, and "Completed." when the simulation is complete.

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

The simulator reads parameters from the configuration file ("scheduling.conf"). It creates a specified number of processes, each of which blocks for input or output after a number of milliseconds that can be specified for each process. Each process is allowed to run for a randomly generated amount of time, with the amount of time constrained by a specified average (mean) in milliseconds, and standard deviations from that average. The simulation may also be bounded in the total length of its run.

After reading the configuration file, the scheduling algorithm then "runs" the processes, causing each to block for input or output after the specified interval until all processes have completed their randomly generated amount of runtime, or until the maximum amount of runtime for the simulation is exceeded.

As the simulation proceeds, a log file ("Summary-Processes") is generated which shows the activity of the scheduling algorithm as it considers each process in the process queue.

After the simulation halts, a summary report ("Summary-Results") is generated which shows statistics for each process and for the simulation as a whole.

The Configuration File

The configuration file (`scheduling.conf`) is used to specify various parameters for the simulation, including:

- the number of processes,
- the mean runtime for a process,
- the standard deviation in runtime for a process,
- for each process, how long the process runs before it blocks for input or output, and
- how long the simulation should run.

Configuration File Options

There are a number of options which can be specified in the configuration file. These are summarized in the table below.

Keyword	**Values**	**Description**	
`numprocess`	*n*	The number of processes to create for the simulation.	
`meandev`	*n*	The average length of time in milliseconds that a process should execute before terminating.	
`standdev`	*n*	The number of standard deviations from the average length of time a process should execute before terminating.	
`process`	*n*	The amount of time in milliseconds that the process should execute before blocking for input or output. There should be a separate `process` directive for each process specified by the `numprocess` directive.	
`runtime`	*n*	The maximum amount of time the simulation should run in milliseconds.	

Sample Configuration File

The "scheduling.conf" configuration file looks like this:

```
// # of Process
numprocess 3

// mean deivation
meandev 1100

// standard deviation
standdev 510

// process    # I/O blocking
process 100
process 500
process 30

// duration of the simulation in milliseconds
runtime 5000
```

The Summary-Results File

The Summary-Results file contains a summary report describing the simulation and includes one line of summary information for each process. The fields and columns in the report are described in the following table.

Field	Description
Scheduling Type:	The type of the scheduling algorithm used. The value displayed is "hard coded" in the SchedulingAlgorithm.java file.
Scheduling Name:	The name of the scheduling algorithm used. The value displayed is "hard coded" in the SchedulingAlgorithm.java file.
Simulation Run Time:	The number of milliseconds that the simulation ran. This may be less than or equal to the total amount of time specified by the "runtime" configuration parameter.
Mean:	The average amount of runtime for the processes as specified by the "meandev" configuration parameter.
Standard Deviation:	The standard deviation from the average amount of runtime for the processes as specified by the "standdev" configuration parameter.
Process #	The process number assigned to the process by the simulator. The process number is between 0 and n-1, where n is the number specified by the "numprocess" configuration parameter.
CPU Time	The randomly generated total runtime for the process in milliseconds. This is determined by the "meandev" and "standdev" parameters in the configuration file.
IO Blocking	The amount of time the process runs before it blocks for input or output. This is specified for each process by a "process" directive in the configuration file.
CPU Completed	The amount of runtime in milliseconds completed for the process. Note that this may be less than the CPU Time for the process if the simulator runs out of time as specified by the "runtime" configuration parameter.
CPU Blocked	The number of times the process blocked for input or output during the simulation.

Sample Summary-Results File

The output "Summary-Results" file looks something like this:

```
Scheduling Type: Batch (Nonpreemptive)
Scheduling Name: First-Come First-Served
Simulation Run Time: 2750
Mean: 1100
Standard Deviation: 510
Process #       CPU Time        IO Blocking     CPU Completed   CPU Blocked
0               1372 (ms)       100 (ms)        1372 (ms)       13 times
1               689 (ms)        500 (ms)        689 (ms)        1 times
2               689 (ms)        30 (ms)         689 (ms)        22 times
```

The Summary-Processes File

The Summary-Processes file contains a log of the actions taken by the scheduling algorithm as it considers each process in the scheduling queue.

Each line in the log file is of the following form:

`Process:` *`process-number process-status`*`...` `(`*`cpu-time block-time accumulated-time accumulated-time`*`)`

The fields in the line are described in the table below.

Field	Description
process-number	The process number assigned to the process by the simulator. This is a number between 0 and n-1, where n is the value specified for the "numprocess" configuration parameter.
process-status	The status of the process at this point in time. If "registered" then the process is under consideration by the scheduling algorithm. If "I/O blocked", then the scheduling algorithm has noticed that the process is blocked for input or output. If "completed", then the scheduling algorithm has noticed that the process has met or exceeded its allocated execution time.
cpu-time	The total amount of run time allowed for this process. This number is randomly generated for the process based on the "meandev" and "standdev" values specified in the configuration file.
block-time	The amount of time in milliseconds to execute before blocking process. This number is specified for the process by the "process" directive in the configuration file.
accumulated-time	The total amount of time process has executed in milliseconds. (This number appears twice in the log file; one should be removed).

Sample Summary-Processes File

The output "Summary-Processes" file looks something like this:

```
Process: 0 registered... (1372 100 0 0)
Process: 0 I/O blocked... (1372 100 100 100)
Process: 1 registered... (689 500 0 0)
Process: 1 I/O blocked... (689 500 500 500)
Process: 0 registered... (1372 100 100 100)
Process: 0 I/O blocked... (1372 100 200 200)
Process: 1 registered... (689 500 500 500)
Process: 1 completed... (689 500 689 689)
Process: 0 registered... (1372 100 200 200)
Process: 0 I/O blocked... (1372 100 300 300)
Process: 2 registered... (689 30 0 0)
Process: 2 I/O blocked... (689 30 30 30)
Process: 0 registered... (1372 100 300 300)
Process: 0 I/O blocked... (1372 100 400 400)
```

```
Process: 2 registered... (689 30 30 30)
Process: 2 I/O blocked... (689 30 60 60)
Process: 0 registered... (1372 100 400 400)
Process: 0 I/O blocked... (1372 100 500 500)
Process: 2 registered... (689 30 60 60)
Process: 2 I/O blocked... (689 30 90 90)
Process: 0 registered... (1372 100 500 500)
Process: 0 I/O blocked... (1372 100 600 600)
Process: 2 registered... (689 30 90 90)
Process: 2 I/O blocked... (689 30 120 120)
Process: 0 registered... (1372 100 600 600)
Process: 0 I/O blocked... (1372 100 700 700)
Process: 2 registered... (689 30 120 120)
Process: 2 I/O blocked... (689 30 150 150)
Process: 0 registered... (1372 100 700 700)
Process: 0 I/O blocked... (1372 100 800 800)
Process: 2 registered... (689 30 150 150)
Process: 2 I/O blocked... (689 30 180 180)
Process: 0 registered... (1372 100 800 800)
Process: 0 I/O blocked... (1372 100 900 900)
Process: 2 registered... (689 30 180 180)
Process: 2 I/O blocked... (689 30 210 210)
Process: 0 registered... (1372 100 900 900)
Process: 0 I/O blocked... (1372 100 1000 1000)
Process: 2 registered... (689 30 210 210)
Process: 2 I/O blocked... (689 30 240 240)
Process: 0 registered... (1372 100 1000 1000)
Process: 0 I/O blocked... (1372 100 1100 1100)
Process: 2 registered... (689 30 240 240)
Process: 2 I/O blocked... (689 30 270 270)
Process: 0 registered... (1372 100 1100 1100)
Process: 0 I/O blocked... (1372 100 1200 1200)
Process: 2 registered... (689 30 270 270)
Process: 2 I/O blocked... (689 30 300 300)
Process: 0 registered... (1372 100 1200 1200)
Process: 0 I/O blocked... (1372 100 1300 1300)
Process: 2 registered... (689 30 300 300)
Process: 2 I/O blocked... (689 30 330 330)
Process: 0 registered... (1372 100 1300 1300)
Process: 0 completed... (1372 100 1372 1372)
Process: 2 registered... (689 30 330 330)
Process: 2 I/O blocked... (689 30 360 360)
Process: 2 registered... (689 30 360 360)
Process: 2 I/O blocked... (689 30 390 390)
Process: 2 registered... (689 30 390 390)
Process: 2 I/O blocked... (689 30 420 420)
Process: 2 registered... (689 30 420 420)
Process: 2 I/O blocked... (689 30 450 450)
Process: 2 registered... (689 30 450 450)
Process: 2 I/O blocked... (689 30 480 480)
Process: 2 registered... (689 30 480 480)
Process: 2 I/O blocked... (689 30 510 510)
Process: 2 registered... (689 30 510 510)
Process: 2 I/O blocked... (689 30 540 540)
Process: 2 registered... (689 30 540 540)
Process: 2 I/O blocked... (689 30 570 570)
Process: 2 registered... (689 30 570 570)
```

```
Process: 2 I/O blocked... (689 30 600 600)
Process: 2 registered... (689 30 600 600)
Process: 2 I/O blocked... (689 30 630 630)
Process: 2 registered... (689 30 630 630)
Process: 2 I/O blocked... (689 30 660 660)
Process: 2 registered... (689 30 660 660)
Process: 2 completed... (689 30 689 689)
```

The program and its input and output files are described more fully in the *MOSS Scheduling Simulator*.

Please send suggestions, corrections, and comments to Ray Ontko (*rayo@ontko.com*).

Last updated: July 7, 2001