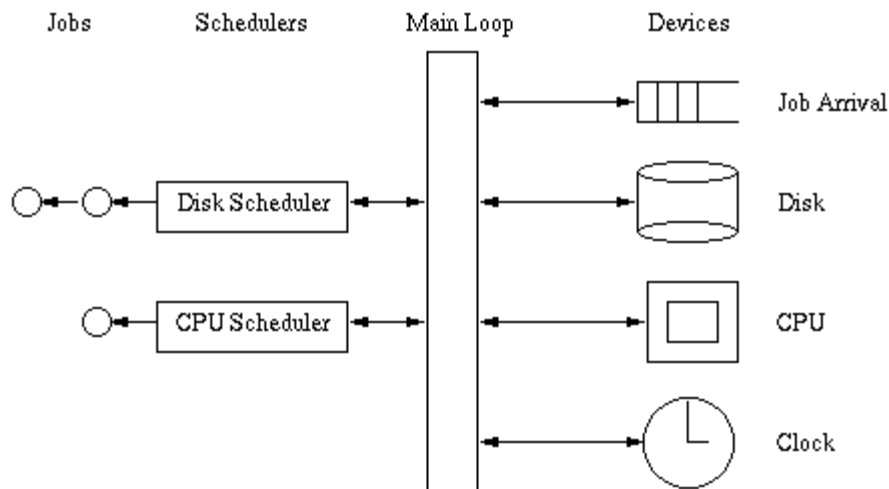


Term Project “Scheduler Simulator”	2
<i>System Overview</i>	2
<i>Input File</i>	2
<i>Statistics</i>	3
<i>Experiments</i>	3
<i>Presentation/Report</i>	4
<i>Grading</i>	4
<i>Deliverables</i>	4
<i>Due Date</i>	4
<i>Useful Links</i>	4

Term Project “Scheduler Simulator”

Working in groups of **MAX THREE**, design and develop an OO generic simulator that allows implementation of different Scheduling algorithms. You will then use this simulator to experiment with different work loads and different scheduling algorithms and prepare a comparison report.

System Overview



- The Simulator essentially consists of Jobs, Devices, and Schedulers -- all coordinated by a single loop of code.
- A **Job** is a customer of services: it is a process that needs to use system resources during its execution.
- A **Device** represents a resource in the system. The devices available to a job are the CPU and the disk.
- You will also have a **clock** device and a **pseudo-device** that interrupts whenever a new job arrives in the system.
- A **Scheduler** coordinates access to a device. It queues Jobs that are waiting to use a device and will choose which job is the next to access.

The overall execution of the simulator occurs like this: Jobs arrive at the JobArrival device and are entered into the system. A job's lifetime consists of alternating periods of using the CPU (called a burst) and performing I/O. The Main Loop is responsible for moving jobs around the system. It sends them to a scheduler, takes the next job from a scheduler, and starts and stops jobs running on a device. The Disk Scheduler and the CPU Scheduler decide which job should be the next to run on their respective devices. They also buffer jobs that are waiting to run but have not yet been given access. The clock device is used to enable preemption.

Input File

The simulation is driven by a randomly generated *input file*. Each line contains $n + 3$ integers, where n is the number of I/O operations performed by the job:

- the arrival time of the job, in milliseconds from the start,
- the number n of I/O operations, and
- $n + 1$ numbers indicating the lengths of “bursts”. There is one burst between each pair of consecutive I/O operations, as well as one burst before the first I/O and one after the last I/O.

The lines are sorted by arrival time.

Statistics

On termination, the simulator prints a large number of statistics about the run, including

- The *utilization* of each device.
- Information about each scheduler queue, including the maximum and average queue lengths, and the percentage of the time it had various lengths.
- Information about jobs, including the number of jobs run and throughput, and summary statistics (min, max, and mean) about the per-job elapsed time, CPU time, I/O operations, waiting time, and penalty ratio. Waiting time includes both *queuing delay* (time the burst sits in the CPU Scheduler queue) and *scheduling overhead* (2ms for each process switch, in this simulation).

Experiments

You need to compare the performance of **three** scheduling algorithms. You should also compare the performance for various values of the parameters **quantum**. Note that if **quantum** is very large, RR becomes *First-Come-First-Served* (FCFS).

1. Run your simulator for each scheduling policy, with a variety of quantum values. For each version of the simulator, for each input data file, for each quantum value, plot the
 - completion time,
 - throughput,
 - average job elapsed time,
 - average job waiting time.
2. From the data in part 1, determine (for each scheduling algorithm) the range of values for the quantum that have a noticeable effect on performance. In other words, what is the smallest and largest quantum that will affect performance?
3. From the data in part 1, note any performance differences and similarities between the scheduling algorithms. Are the results consistent for different input data files? Are the results consistent for different quantum values? Is one algorithm always better than another? Try to figure out if these differences make sense to you and explain these differences and similarities.

Use the “scientific method.” You should form some hypotheses before you start experimenting and use the experiments to confirm or reject these hypotheses based on observed results. Give

careful thought to the correct choice of parameters for the programs. Try a few trial runs with various parameters, print out the results, and think about the results. These preliminary results should help you decide on better parameters for a second round of trials. Remember: It's not the quantity but the quality of data that dictates the quality of the experiments.

If the program is not printing out all the statistics you would like to see, feel free to modify it to produce better output. You may find additional statistics-reporting code can help explain some of the behavior you observe.

Presentation/Report

You are to prepare a presentation describing the results of your experiments. You should carefully describe what experiments you did and what the results showed you about the different scheduling policies. We want to see a correlation between the experiments you run and the conclusions you draw. You must supply *quantitative* data to support your conclusions. Your presentation should not exceed 9 minutes.

Grading

You grade will be determined as follows:

- 40% - Report (experiments, conclusions, presentation)
- 60% - Implementation (correctness, style, documentation)

Deliverables

Upload the following **before** your presentation:

1. Source code of your simulator
2. Word document describing how to run your program, your experiments, and your results.
3. Power Point slides that you will use in your presentation.

Due Date

All presentations will take place during week 16 (May 28 -> Jun 1).

Useful Links

- Animated Process Life Cycle Flash
http://globalbts.com/qu/csd/el/510405af2/document/OS_Animation/OS%20animation/process.swf
- Visualization of Process Scheduling Applet
<http://globalbts.com/qu/csd/el/5104058c6/document/Animations/ProcSchedApplet/ProcSched.htm>