**CS 475 Operating Systems**

**CPU Scheduler & Dispatch Module Simulation Project**

### Due March 28, 2012

In this assignment, you will design and implement a ***discrete event model*** simulating the CPU dispatch scheduling (short-term and dispatch management) for a multi-core system. You will also perform extensive test runs, collecting data, perform benchmark analysis and report those findings. Finally, you are to report your final system assessment as though you are a systems analyst.

* Work in **teams** of 2, however submit an individual final assessment as noted below
* You may use a **programming language of your choice**, in **Linux**. You’ll likely want to use C++ or Java.

**Objectives:** For you to integrate OS theory and practice in the design of a substantial model/simulation

Gain a better understanding of the process scheduling and dispatch process.

Learn discrete event modeling/simulation concepts

Learn how to thoroughly analyze a system and present a meaningful assessment

Strengthen your Linux proficiency

### NOTE: This project is a rigorous project and requires extensive time for programming, testing and analysis. DO NOT wait until the last minute. will require external time to complete.

### Requirements:

**Part A: Model Design & Implementation [60 Points]**

You are to create a simulation program that models a CPU scheduler and dispatch module, as well as the hand-off to another level of scheduling on multi-cores. Realize any model is a partial representation – you must make assumptions and cleverly design the most accurate representation possible. For instance, it is a **discrete** **event** model (see below in timing).

* Processes are ***model entities***, implemented as objects including the necessary accounting structures. For instance a PCB structure tracking the control and accounting information you deem necessary (i.e. PID, priority, state, CPU burst time, etc.) Note, it is not necessary to include all elements of an actual PCB – only the elements required for your model.
* Process vectors or queues that reference PCBs with PIDs & corresponding references to the PCBs
* Randomly generate processes with necessary information, arrival times, CPU burst times, IO burst times, etc.
* This is where you can be clever. Design and implement your scheduling queue configuration and algorithmic schemes.
  + As processes are created, they first enter a wait queue
  + Implement a central Multilevel Feedback Queue with some type of Aging scheme.
    - Consider the criteria used to assign processes to queues
    - Implement some type of Aging scheme – what criteria will be used to determine when to upgrade or demote a process from queue to queue. Remember this means you’ll need to move processes from queue to queue
    - Implement at least 3 queues with at least 3 different scheduling algorithms:

for instance: Round Robin, Priority, FCFS.

* + Design and implement another level of scheduling at the core thread level. Practically speaking, this should be implemented as threads.
  + Design and implement some form load balancing across cores.
* Model Timing:
  + Implement time quantums as ***discrete events*** – that is just include the designated block of time ***NOT a continuous interval clock module***.
  + Also model context switches as ***discrete events*** – again as a designated block of time. Don’t try to model all the steps of a context switch.
  + Processing of CPU bursts are also treated as ***discrete events***.
  + Set your program to run until the given processes have completed, capturing all necessary time measurements for analysis. You’ll need to capture the data in files for later benchmarking analysis.

**Part B: Benchmarking Analysis & Report [20 points]**

* Testing is an important part of Simulation. **DO NOT underestimate the time required for this portion**. Allow sufficient time to make **many sets of simulation runs** (remember you will be randomly generating process times and deriving averages). Experiment with:
  + Varied time quantums
  + Varied configuration of queues, variations of process allocation to specific queues
* Collect the necessary data during runs to perform thorough evaluation of:
  + Throughput time
  + Average turnaround times
  + Average wait times
  + Average response times
  + Average context switch time
* Not required, but an extra could be queueing analysis. For instance, knowing arrival rates and service rates, you can compute CPU utilization, average queue length, etc. You are already required to monitor wait times.
* **Team Benchmarking Report** with graphs, numerical data collection and stated assumptions.

Make it look PROFESSIONAL! Not

* + As with any simulation model - You will likely need to make **assumptions** for your system such as timing issues, arrival policies, etc? State those clearly in this document.
  + Create **graphs** (i.e. Excel) that plot the required analysis of **average Turnaround(Tt ), average Wait(Wt ) and average Response (Rt )** given a **significant number of runs** with **different time quantums**. These will be plotted in 3 graphs with metrics (e.g. average Tt , Wt , Rt on they-axes, respectively) per time quantums on the x-axis.
  + **Report data collection** for the graphs. You’ll want to collect data in files and then calculate averages.

**Part C: Individual write-ups and discussion of your findings [20 points]**

A written discussion from your **individual perspective**. This is where you step into an Analyst role.

**a.)** Give a **summary** discussing your design, implementation methods, observations, findings, conclusions.

**b.) Given the analytical findings, what were your observations?** For instance, how did you handle such things as queue distribution, switching, etc. What quantum resulted in the best average turnaround times, did it favor shorter versus longer jobs? What configuration resulted in the best response time, in the least wait time? What effect did varied quantum times have? What differences did you see with predominantly short burst time processes with more context switches versus predominantly longer burst time processes? Other important observations are what time quantum resulted in the best average turnaround times, did it favor shorter versus longer jobs? What effect did varied quantum times have? How did you handle the multi-core scheduling?

**b.) Systems Assessment and suggestions**. You are the **Systems Analyst.** After all of this, what is your personal assessment of the design and implementation given your findings? What are strengths/weaknesses of your system? What are potential improvements that should be made to the system?

**c.) What role did you play on the project?** What did you contribute? What did you learn from this project?

### Deliverables March 28:

* **Walkthrough/Demonstration** of your Simulation Model
* **Informal discussion of your findings**
* **Group Benchmarking Report**
* **Individual assessment write-ups**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Grading Rubric/Criteria**

**This project is weighted the same as an exam.**

**Part A: Simulation Model and Walkthrough (60 points)**

30 points Extent to which requirements are met

20 points Model – Design and implementation solution of algorithms, queue management

10 points SW Engineering – Quality of code, clean code, design, comments

**Part B: Group Benchmarking Analysis Report (20 points)**

20 points Graphs of T, W, and R; Data collections, Statement of Assumptions

**Part C: Individual Writeup (20 points)**

20 points Observations and conclusive system assessment

Role on the project

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**100 Total Possible**