CS 340 Spring 2014 Programming Assignment II and Paper

Paper due 10 April, Program due 24 April

The Program

Add scheduling for your disk queues, a preemptive history based scheduler for your ready queue and CPU usage accounting to your OS.

Along with all the information you got in the first program's SysGen, you should prompt the installer (me) for the *history parameter*, $\alpha, 0 \leq \alpha \leq 1$, the initial burst estimate (the initial value of τ, in milliseconds, for all new processes) and the number of cylinders each disk has.

This program should do *everything* your first program did as well as the following: when an interrupt occurs or a process in the CPU issues a system call the OS (your program) should query the timer (me) for the amount of time the process has used the CPU. The OS should then compute all the appropriate values and implement a history-based SJF approximation algorithm with $\tau_{\text{next}} = \alpha * \tau_{\text{previous}} + (1-\alpha) * t_{\text{previous}}$ (where t_{previous} is the actual previous CPU burst and τ is the estimated burst). A call for a disk read or write should prompt for the cylinder to access as well as all the other information. (Yes, real files on real systems are scattered across many cylinders. I'm trying to keep this reasonably simple.)

The disk queue should no longer be a FIFO queue, but one ordered so that the head seek time is minimized while no requests are starved. See "The Paper" for more details.

An S-{r|c|d|p} should show all the appropriate information well as the total time each process in the displayed queue has used the CPU, each processes average burst time and the systems average total CPU time of competed processes. Be sure the queues are displayed "in order", that is in the order the processes will be removed from the queue if no new processes are added. When a process terminates, its PID, total CPU time, and average burst time are reported to the accounting module (me) and the system's average total CPU time is updated.

Email source code and compile instructions *before* class on the due date. If you are sending one or two files, there is no need to package them. If you are sending more than two you should package them. Tar and zip are fine. Please make sure that the files unpack into the SAME directory the source file is in. Please make sure you don't include `__MACOSX` files. If you are sending c++ code, and I can compile it with `g++ -o ~/temp/run.me *cpp` then I don't need a makefile. If you do supply a makefile, please make sure the output executable is `~/temp/run.me`. Also be aware

that I will count the readability of your output in the grade.

The Paper

In order to decide on your disk queue scheduling algorithm you will need to do some research. Your goal is to find a fast algorithm that won't starve any waiting processes. Research the known algorithms and write a 500-1000 word paper (2-3 pages, double spaced, 1 inch margins, 12 point times roman) analyzing the choices and justifying your choice to implement one of them. I will compare what you choose in the paper to what you implement in the program, but success in that will only effect your program grade. This paper will be graded on readability (grammar, sentence and paragraph structure, understandability) completeness and documentation.

Hand in a printed, stapled document at the beginning of class. No cover or title pages, but your name should be somewhere. All your work should be documented; remember that undergraduates may have opinions but are not supposed to know anything. Fact sources should be referenced. I don't care if you use foot-notes, in-notes or end-notes or whether you stick to Chicago, MLA, APA or some other documentation standard, but all the "right" info has to be in the references and bibliography. The bibliography does not count to the size of the paper.