CS 450 take-home midterm exam

1 Instructions

This is an **individual**, open-book/slides, open-notes examination. After starting work on it you should not seek help from or discuss it with anyone else (and this precludes searching for answers on the Internet) — the only exception would be to e-mail the instructor directly if you require clarification of any of the questions/instructions on the exam. I am holding you to your honor on this; please don't disappoint!

You have until **10AM CDT on Monday, October 21, 2013**, to complete the exam. Your answers to all the problems should be clearly written or typed up, and submitted at the beginning of lecture on the due date or as a PDF via BlackBoard.

2 True/False (with justification) Problems

For each of the following statements indicate whether it is true or false, and supply a short ($<$ 75 word) justification. When your answer is "false" you should try to refute the original statement with a detailed example/scenario. (4 points each.)

1. Concurrency and its resulting race conditions are possible even when non-preemptive multitasking takes place on a uniprocessor machine.

2. Reducing the quantum length of a RR scheduler will tend to improve the responsiveness of jobs with relatively short CPU burst lengths.

3. For a given set of jobs, all non-preemptive scheduling policies will require the same amount of context switch time overhead.

4. Priority inversion is not possible in the context of a multilevel feedback queue based scheduling system.

5. Starvation is a potential problem for all preemptive scheduling policies.

3 Numerical Problems

Some of these questions require that you supply either a single numeric result or a table of numeric results. You may use pencil & paper, a calculator, or write a program to come up with your solution. If you choose to write a program, include the source code in your submission; otherwise, you must show all significant steps in your computation. Clearly label your final answer(s).

6. (8 points) Consider a scheduling simulation involving 20 processes, each running for a total duration of 1000ms spread over separate CPU bursts ranging uniformly in duration from 25ms to 75ms.

   The following four scheduling policies are used in the simulation:

   - FCFS
   - RR, q=50
   - SJF (non-preemptive)
   - SRTF

   The following are the results of the simulation (policies not in order):

Policy	Total CST	Avg wait time	Std dev of wait time
#1	1212	17772	21.16
#2	816	12106	235.68
#3	816	17152	19.59
#4	980	12143	240.09

   Which policies correspond to #1, #2, #3 and #4? Justify your answer.

7. (12 points) You are to simulate the execution of a set of processes using a MLFQ scheduler consisting of 3 RR queues with quanta of 3ms, 6ms, and 9ms. All processes initially enter the highest-priority 3ms queue at the beginning of their first burst, after which they are moved between queues according to the semantics covered in class.

   There are 3 processes, with the following arrival times and ordered CPU burst durations:

   - P_0 arriving at $t = 0$, CPU bursts = 5ms, 8ms, 8ms
   - P_1 arriving at $t = 4$, CPU bursts = 5ms, 2ms, 5ms
   - P_2 arriving at $t = 12$, CPU bursts = 10ms, 2ms

Assume that successive CPU bursts of a given process are separated by 10ms I/O bursts, and that any number of I/O bursts can be serviced in parallel. Additionally, the following sub-policies apply:

- if a new job is entering a RR queue at the same time an existing job is, the latter is queued ahead
- if a job is executing from a lower priority queue when a job becomes ready in a higher priority one, the latter preempts the former; when the former resumes execution it will only be scheduled for the remainder of its time quantum

Compute the turnaround and cumulative wait times for each process. Support your answer by including a Gantt chart that visually documents when each process is executing on the CPU.

8. (4 points) A particularly industrious professor has decided to try to implement rapid feedback for exams, whereby after submission of an exam a student need wait no longer than 2 hours, on average, to receive a score. Given that exams arrive in the professor's inbox at an average rate of 3 per hour, how quickly must the professor grade each exam to meet his goal?

9. (4 points) Anxious students petition to decrease the average turnaround time for scores. By how much must the professor increase his average grading rate in order to decrease the average turnaround time by 10%, 25%, 50%, and 90%? Give your answers in percentages.

   Note: decreasing turnaround time by 10% results in a $2-(0.1)(2) = 1.8$ hour turnaround time; given a grading rate of N per hour, increasing grading rate by 50% would result in a $(N + (0.5)(N))$ per hour rate.

10. (4 points) The professor decides, ultimately, to pace himself based on how many students are waiting for their scores to be returned to them. If he is willing to keep 5 students, on average, waiting for their scores, how quickly must he grade the exams? Assume that exams continue to arrive at the same rate of 3 per hour.

4 Discussion Problems

This section asks you to do a bit of critical thinking based on concepts covered in class. For each question you should supply a clear, well organized, properly structured (i.e., as in grammatically sound) written response of less than 500 words in length. (8 points each.)

11. **Kernel organization**. After being hired as a senior software engineer at OSes"R"Us, you are asked to help architect a new kernel for use in embedded systems with limited CPU/RAM. The system must be very robust, yet flexible enough to be tailored to suit a variety of different types of devices. How would you choose to organize your kernel? Why?

12. **Policy vs. Mechanism**. Give a specific example in the context of operating system implementation of the policy vs. mechanism dichotomy. Make a case *for* their separation (in your specific example), and then make a case *against* their separation.

13. **Scheduling for RTSes**. Processes running on a real-time system (RTS) may ask the operating system to help them achieve hard, real-time deadlines for the completion of certain tasks. Describe a scheduling policy that you might choose for a RTS, and another that would *not* be suitable. Clearly explain your reasoning.

5 xv6 Problems

14. (6 points) Detail the steps taken by both the hardware and xv6 to carry out a context switch between two separate user processes. You should identify the functions involved in, and the structures being consulted/manipulated at each step.

15. (4 points) What is the significance of line 2719 (`movl %esp, (%eax)`, in `swtch`)? What would happen if it were omitted?

16. (4 points) Examine the inner loop of `scheduler` (starting at line 2468 in the xv6 source booklet). The policy being implemented appears to be RR, but it turns out that there are circumstances where process selection is less than truly "fair". Explain.

17. (4 points) The `tvinit` function (3067) sets up the entries (aka "gates") of the interrupt descriptor table (IDT) — what is the significance of

the `DPL_USER` constant at line 3073? Why isn't it used to initialize other gates?

18. (6 points) What section of code determines whether or not to return control back to a user process following a trap? Immediately after carrying out a non-terminal system call, is it possible that xv6 will choose to schedule a new process instead of returning to the trapping process? Explain.