CSCI 4320 (Principles of Operating Systems), Fall 2003

Homework 3

Assigned: October 7, 2003.

Due: October 14, 2003, at 5pm. *Not accepted late.*

Credit: 20 points.

> *Note:* The HTML version of this document may contain hyperlinks. In this version, hyperlinks are represented by showing both the link text, formatted like this, and the full URL as a footnote.

1 Reading

Be sure you have read chapters 2 and 3.

2 Problems

Do the following problems. You may write out your answers by hand or using a word processor or other program, but please submit hard copy, either in class or in my mailbox in the department office.

1. (8 points) Five batch jobs (call them A through E) arrive at a computer center at almost the same time. Their estimated running times (in minutes) and priorities are as follows, with 5 indicating the highest priority:

job	*running time*	*priority*
A	10	3
B	6	5
C	2	2
D	4	1
E	8	4

   For each of the following scheduling algorithms, determine the turnaround time for each job and the average turnaround time. Assume that all jobs are completely CPU-bound (i.e., they do not block). (Before doing this by hand, see the optional programming program.)

   - First-come, first-served (run them in alphabetic order by job name).
   - Shortest job first.
   - Round robin, using a time quantum of 1 minute.
   - Round robin, using a time quantum of 2 minutes.
   - Priority scheduling.

2. (4 points) Recall that some proposed solutions to the mutual-exclusion problem (e.g., Peterson's algorithm) involve busy waiting. Do such solutions work if priority scheduling is being used and one of the processes involved has higher priority than the other(s)? Why or why not? How about if round-robin scheduling is being used? Why or why not?

3. (4 points) Suppose that a scheduling algorithm favors processes that have used the least amount of processor time in the recent past. Why will this algorithm favor I/O-bound processes yet not permanently starve CPU-bound processes?

4. (4 points) Suppose you are designing an electronic funds transfer system, in which there will be many identical processes that work as follows: Each process accepts as input an amount of money to transfer, the account to be credited, and the account to be debited. It then locks both accounts (one at a time), transfers the money, and releases the locks when done. Many of these processes could be running at the same time. A friend proposes a simple scheme for locking the accounts: First lock the account to be credited; then lock the account to be debited. Can this scheme lead to deadlock?

   If you think it cannot, briefly explain why not. If you think it can, first give an example of a possible deadlock situation, and then design a scheme that avoids deadlocks, but in such a way that once an account is locked, it is not released until the funds transfer is complete (i.e., a design that relies on repeatedly locking one account, trying the other, and releasing the first is not allowed).

3 Optional Programming Problem

For extra credit, do the following programming problem. Turn in your code (all files needed) by sending mail to `cs4320@cs.trinity.edu`, with each of your code files as an attachment. Please use a subject line such as "homework 3" or "hw3". You can develop your program on any system that provides the needed functionality, but I will test it on one of the department's RedHat 9 Linux machines, so you should probably make sure it works in that environment before turning it in.

1. (Up to 5 extra-credit points) The starting point for this problem is a program scheduler.cpp[1] that simulates execution of a scheduler, i.e., generates solutions to problem 1. Currently the program simulates only the FCFS algorithm. Your mission is to make it simulate one or more of the other algorithms mentioned in problem 1. (Feel free to rewrite anything about this program, including starting over in a language of your choice. Just remember that the program has to run on one of the department Linux machines. If you change the input requirements, change the comments so they describe the changed requirements.)

[1] `http://www.cs.trinity.edu/~bmassing/CS4320_2003fall/Homeworks/HW03/Problems/scheduler.cpp`