# CSCI 3323 (Principles of Operating Systems), Fall 2011

# Homework 5

**Credit:** 20 points.

## 1 Reading

Be sure you have read Chapter 3, sections 3.4 through 3.9.

## 2 Problems

Answer the following questions. 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. (5 points) Consider (imagine?) a very small computer system with only four page frames. Suppose you have implemented the aging algorithm for page replacement, using 4-bit counters and updating the counters after every clock tick, and suppose the $R$ bits for the four pages are as follows after the first four clock ticks.

| Time | $R$ bit (page 0) | $R$ bit (page 1) | $R$ bit (page 2) | $R$ bit (page 3) |
|---:|---:|---:|---:|---:|
| after tick 1 | 0 | 1 | 1 | 1 |
| after tick 2 | 1 | 0 | 1 | 1 |
| after tick 3 | 1 | 0 | 1 | 0 |
| after tick 4 | 1 | 1 | 0 | 1 |

   What are the values of the counters (in binary) for all pages after these four clock ticks? If a page needed to be removed at that point, which page would be chosen for removal?

2. (5 points) A computer at Acme Company used as a compute server (i.e., to run non-interactive jobs) is observed to be running slowly (turnaround times longer than expected). The system uses demand paging, and there is a separate disk used exclusively for paging. The sysadmins are puzzled by the poor performance, so they decide to monitor the system. It is discovered that the CPU is in use about 20% of the time, the paging disk is in use about 98% of the time, and other disks are in use about 5% of the time. For each of the following, say whether it would be likely to increase CPU utilization (i.e., the percentage of time the CPU is in use) and why.

   (a) Installing a faster CPU.

   (b) Installing a larger paging disk.

   (c) Increasing the number of processes (degree of multiprogramming).

   (d) Decreasing the number of processes (degree of multiprogramming).

   (e) Installing more main memory.

   (f) Installing a faster paging disk.

## 3 Programming Problems

Do the following programming problems. You will end up with at least one code file per problem. Submit your program source (and any other needed files) by sending mail to `bmassing@cs.trinity.edu`, with each file as an attachment. Please use a subject line that mentions the course number and the assignment (e.g., "csci 3323 homework 5"). You can develop your programs on any system that provides the needed functionality, but I will test them on one of the department's Linux machines, so you should probably make sure they work in that environment before turning them in.

1. (10 points) The starting point for this problem is a Java program that simulates execution of a page replacement algorithm. Currently the program simulates only the FIFO algorithm. Your mission is to make it simulate one or more of the other algorithms mentioned in the text (and listed in commented-out code in the main program). You will get full credit for simulating one algorithm, extra points for simulating additional algorithms. The program consists of several classes collected in a package called `pagingsimulator`.

   - API documentation[1].
   - Code[2]. (Class `PageReplacerTest` contains a `main` method to test all algorithms.)
   - Sample input[3]. (Output from my sample solution[4].)
   - Another sample input[5]. (Output from my sample solution[6].)

   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, and it needs to accept input from command-line arguments and files — no GUIs, Web-based programs, etc. The latter requirement is to make it easier for me to automate testing your code. If you make changes to the format of the input — and I prefer that you don't — change the comments so they describe the changed requirements.

   Make the following assumptions:

   - Initially memory is empty.
   - All memory references are valid — if the page is not in memory, it can be read in from disk. (You don't have to simulate that part, just count how often it happens.)

[1] `http://www.cs.trinity.edu/~bmassing/Classes/CS3323_2011fall/Homeworks/HW05/Problems/pagingsimulator/docs/`

[2] `http://www.cs.trinity.edu/~bmassing/Classes/CS3323_2011fall/Homeworks/HW05/Problems/pagingsimulator/source/pagingsimulator/`

[3] `http://www.cs.trinity.edu/~bmassing/Classes/CS3323_2011fall/Homeworks/HW05/Problems/pagingsimulator/sample.in`

[4] `http://www.cs.trinity.edu/~bmassing/Classes/CS3323_2011fall/Homeworks/HW05/Problems/pagingsimulator/sample.out`

[5] `http://www.cs.trinity.edu/~bmassing/Classes/CS3323_2011fall/Homeworks/HW05/Problems/pagingsimulator/another.in`

[6] `http://www.cs.trinity.edu/~bmassing/Classes/CS3323_2011fall/Homeworks/HW05/Problems/pagingsimulator/another.out`