Operating Systems | **CSI3131** **_Assignment 3_** | *Summer 2014*

Memory Management- Paging Algorithms

You must submit your assignment on-line with BlackBoard. This is the only method by which we accept assignment submissions. Do not send any assignment by email. We will not accept them. We are not able to enter a mark if the assignment is not submitted on Blackboard! The deadline date is firm since you cannot submit an assignment passed the deadline. You are responsible for the proper submission of your assignments and you cannot appeal for having failed to do so. A mark of 0 will be assigned to any missing assignment.

Assignments must be done individually. Any team work, and any work copied from a source external to the student (including solutions of past year assignments) will be considered as an academic fraud and will be forwarded to the Faculty of Engineering for imposition of sanctions. Hence, if you are judged to be guilty of an academic fraud, you should expect, at the very least, to obtain an F for this course. Note that we will use sophisticated software to compare your assignments (with other student's and with other sources...). This implies that you must take all the appropriate measures to make sure that others cannot copy your assignment (hence, do not leave your workstation unattended).

Goal: Gain experience with manipulating a file system.

Marks: 50
Posted: June 23, 2014
Due: July 4, 2014

Overview

This assignment consists of a programming exercise and an experimental part you have to briefly explain. You are provided a memory management simulator simulating FIFO page replacement algorithm. Your goal is to implement LRU and CLOCK algorithms as well. To submit the assignment, you have to submit two files: **Paging.java** (programming part) and **Discussion.xxx** (explanation of experimental results; xxx can be a txt, doc, pdf, ps, i.e. any easily readable file). Remember that you can have only one public class per java source file. Any student not conforming the submission specification will loose points. Upload these files for your submission. Do not forget to click on the submit button after (and only after) you have uploaded your files.

Instructions

1. Open **Assign3_codes** folder to get the java and configuration files of a simple java based simulator of a memory management system. Note that while the provided simulator is loosely based on the simulator from http://www.ontko.com/moss/ - memory, the implementation details are quite different. In particular, the syntax of the commands file has been expanded. For more detail, see **MOSSchanges**.
2. Compile the java files.
3. Run the simulator as it is (type at the command prompt)
   java MemoryManagement cmdFile1 confFile1
4. Take a look at the user guide at http://www.ontko.com/moss/memory/user_guide.html and **MOSSchanges**, then examine the provided configuration files 'cmdFileX' and 'confFileX', where .X is a number. Create your own configuration files and play around with the simulator to learn how to use them and how the simulator performs. Examine the log files to see the actions of the simulator documented.
5. Now is time to look at the code you have to modify. Almost all code can (and should) be left as it is. The only class you have to modify is the class **Paging**, which implements most of the page table functionality. You might want to look as well at the methods Kernel.readAddr() and Kernel.writeAddr() executed when read/write instruction is stepped over in the 'commands' sequence. Do not modify these methods - you are submitting only Paging.java, so your modifications would be lost.
6. The method Paging.replacePage() replaces a page when there is a page fault. The replacement algorithm is either FIFO, LRU or CLOCK, depending on the command line arguments (default is FIFO). Only the FIFO algorithm has been implemented in the provided code, your goal is to implement LRU and CLOCK algorithms as well (fill in the methods replacePageLRU() and replacePageClock()). Note that all fields necessary for all algorithms are already included in the PageTableEntry class. These fields still need to be appropriately updated - most of the work is already done in Kernel.readAdrr() and Kernel.writeAddr(), but you might sill need to add some updates (in an appropriate method of Paging class, do not modify the Kernel class). Compile your solutions and try them. Launching the memory simulator as follows:

java MemoryManagement cmdFile1 confFile1 LRU
will use LRU page replacement algorithm, typing on the command line
java MemoryManagement cmdFile1 confFile1 CLOCK
will use the CLOCK replacement algorithm.

7. Examine the log files to compare the performance (number of page faults) for the FIFO vs LRU vs CLOCK. Try several of the provided command files (cmdFileX) with several replacement algorithms. Discuss the results.
8. Take a fixed command file (cmdFile1 or your own command file) and try it with different memory configuration files (confFile1 ... confFile5) to examine how the number of page faults changes with the number of physical pages allocated to the process. Repeat the experiment for every page replacement algorithm. Discuss the results.
9. Submit two files:

a. **Paging.java** which contains your implementation of LRU and CLOCK page replacement algorithms.

b. **Discussion.xxx** which answers to questions in 7 and 8.

Tips:

- the operating system will not call the replacePage methods unless all frames are full (ie. Do NOT handle the case where a page is requested and there is a free frame)
- pageBuffer should be used in all algorithms – the circular buffer in clock / the list of loaded pages for LRU
- The system all ready stamps pages as used and set a clock value for each page.
- FIFO is a good template to work from
- Focus on which page is the victim and updating the page buffer and page table bits