**CS 483 – Operating Systems**

**Fall 2012**

**PEX 4: Simulating Page Replacement - 100 Points**

Due: 2300 hrs on Lesson T37, 5 December 2012

# Help Policy

|  |
| --- |
| **AUTHORIZED RESOURCES:** Any, except another cadet’s assignment or published solutions to the assigned problem.  **NOTE:**   * Never copy another person’s work and submit it as your own. Here are a few blatant examples of copying:   + Making an electronic copy of another cadet’s solution and then modifying it slightly to make it appear as your own work.   + Reading a printout or other source of another cadet’s work as you implement your solution.   + Completing your entire solution by following explicit instructions from another cadet, while he/she refers to his/her own solution * Do not jointly implement a solution. * Helping your classmates learn and understand the homework concepts is encouraged, but extensive assistance should generally be provided by DFCS instructors. Only provide assistance up to your depth of understanding, beyond which assistance by more qualified individuals is more appropriate and will result in greater learning. If you have to look at your solution while giving help, you are most likely beyond your depth of understanding. * Help your classmates maintain their integrity by never placing them in a compromising position. Do not give your solution to another cadet in any form (hard copy, soft copy, or verbal). * **DFCS will recommend a grade of F for any cadet who egregiously violates this Help Policy or contributes to a violation by others.** **Allowing another cadet to see your assignment to help them will result in a zero on this assignment.** |

# Documentation Policy

|  |
| --- |
| 1. You must document all help received from sources other than your instructor or instructor-provided course materials (including your textbook). 2. The documentation statement must explicitly describe WHAT assistance was provided, WHERE on the assignment the assistance was provided, and WHO provided the assistance. 3. If no help was received on this assignment, the documentation statement must state “NONE.” 4. Vague documentation statements must be corrected before the assignment will be graded and will result in a grade deduction equal to 5% (ceiling) of the total possible points. |

**OBJECTIVES**

* Understand the basic concepts of demand paging.
* Be able to implement page replacement algorithms.
* Understand the relationship between frame allocation and page-fault rate.

**OVERVIEW**

In this project you will build a simulator to test different frame allocations using the Least Recently Used (LRU) page replacement algorithm. You will be using a reference stream captured on an Intel Pentium system to find the page fault rate as the number of frames of physical memory varies, testing sizes of 1 to 1000 frames. You will then present your results and make a recommendation on how many frames should be allotted to your reference process.

**SUBMISSION INSTRUCTIONS**

Submit your PEX via the “Submit Assignments” link to Moodle on the Course SharePoint site. Submit all your c source (.c) and header (.h) files, your makefile, your Excel file, and your Word file. Do NOT structure your project into subfolders. Submit each of your files separately. Do NOT zip or otherwise archive your files. Your instructor will download your submission into one folder and should be able to type make and then run your program.

Provide your documentation as a separate documentation.txt file. You may include in-line documentation, but please consolidate your documentation into one file.

NOTE:

* Moodle documentation has been disabled for this assignment. All your documentation must be in your documentation file. As you will not have Moodle to enforce the quality of documentation, be sure you are thorough in your documentation of WHAT assistance was provided, WHERE on the assignment the assistance was provided, and WHO provided the assistance.
* There is a 2MB file size limit on Moodle. If you exceed this limit for this assignment there is probably something drastically wrong with your code, but you may submit your files via email if that becomes necessary.
* You are also limited to uploading a maximum of 20 files.

**GENERAL REQUIREMENTS**

* You must implement this program in C on your Ubuntu virtual machine.
* You must make a single makefile for your project.
  + Include a clean: target that removes all files created as a result of a call to make
  + Your project should compile cleanly (i.e. free of any warnings or errors)
* As this will be a single-threaded program, you should not have any need for global variables.
* Properly allocate and free memory as necessary.

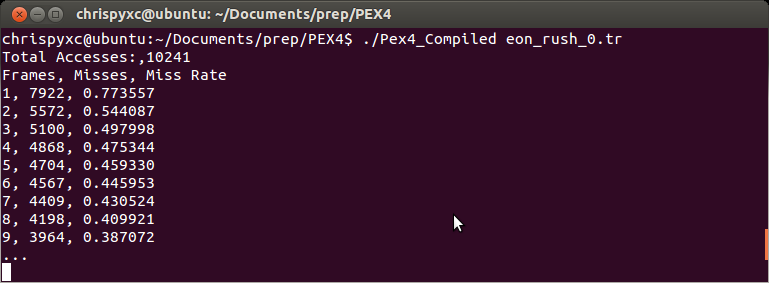
**SIMULATION REQUIREMENTS**

Download the [PEX4 Shell Archive](https://eis.usafa.edu/academics/compsci/CS483/Course%20Documents/Assignments/PEX%204%20Shell.zip) from the [Assignments folder](https://eis.usafa.edu/academics/compsci/CS483/Course%20Documents/Forms/AllItems.aspx?RootFolder=%2Facademics%2Fcompsci%2FCS483%2FCourse%20Documents%2FAssignments&FolderCTID=0x01200013B8C45B79BF6C46A730C987F90C9D26&View=%7bE1F298D3-DE3C-4E98-B4CC-F70F28EA94D7%7d) on the course SharePoint site. It contains three files. eon\_rush\_0.tr is the trace file. It contains, among other items, the physical addresses generated by the reference process. The other files, byutr.h and PEX4Shell.c include code that reads the trace file, as it is stored in binary format with several fields that are not needed for our purposes. (byutr = Brigham Young University TRace. The trace was produced at BYU.)

Your simulation will read the reference stream and track page faults. For each reference in the stream, you will check whether the referenced page is already in memory, if it is not in memory, you will insert it into memory using LRU to determine what page to replace if memory is full.

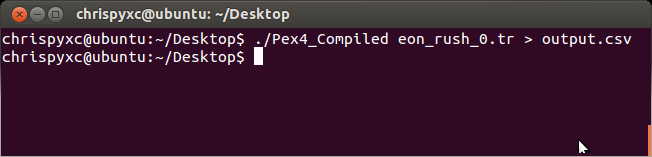
In order to calculate the page-fault rate for each memory size, you will track how many times the system faults and divide by the total number of references. You will repeat this calculation for each of the 1000 memory sizes. Ideally you should discover that page-fault rate is a function of the memory size (more memory implies fewer page-faults).

Output to the console the memory size, number of page faults, and the page-fault rate for each memory size. Each memory size should be on its own line and all values within a line should be separated by a comma. Your output should resemble the output below.



**REPORT REQUIREMENTS**

Run your file, redirecting your output to a file. This is done with the redirect operator “>” as shown below. Copy this file to your host (Windows) machine and open it using Microsoft Excel, being sure to include commas as a delimiter. Create two graphs in Excel: 1)the number of frames vs the number of page faults and 2) the number of frames vs page-fault rate.



Based on your results, write a short (about 3 paragraphs) recommendation for the “optimum” number of frames for this process. Be sure to back up your recommendation. Assume there is a requirement to keep the page-fault rate below 10%. Be sure to address any interesting features on your graphs.

**HINTS/PLAN OF ATTACK**

At first glance one would think that this would require a separate simulation “run” for each size of memory but, there is actually a nice way to do this in a single run using a LRU Stack. Review the book’s description of a stack implementation of LRU in section 9.4.4.

Begin this PEX by building a stack ADT. You will need to be able to find a value in the stack, report its depth, and move it to the top of the stack. If the value is not in the stack, you will pop a new node onto the top of the stack and report back an invalid depth (e.g. -1). Print out the referenced page and the depth returned by your stack function. Desk-check your program by tracking the stack by hand and ensuring correct functionality for the first several references.

The trick here will be to not limit our stack to the size of memory, thus never remove a page from the stack as we “swap out” pages. Instead, you will just let the LRU stack overflow your physical memory size. This will allow you to consider the page to have been in memory if its depth in the stack was less than or equal to the size of memory. Otherwise (if it was not in the stack or it was deeper than the size of memory), it wasn’t in memory and you count this as a page fault. In this way, you will be able to simulate all memory sizes simultaneously and only require a single pass through the trace file.

Use an array to track the number of page faults for each memory size. If your stack function reports a -1, this is a page fault for all memory sizes and every element in the array needs to be incremented. If your stack function reports back a valid depth *d*, then this is a hit for all memory sizes greater than or equal to *d*, but a miss for all frame sizes less than *d*. Therefore, increment the array elements for sizes 1 to *d*-1.

**EXPECTATIONS**

|  |  |
| --- | --- |
| **Requirements**   1. Coding Standards | **+10/10** |
| * 1. Proper decomposition, commenting, naming conventions, indentation, no global variables, etc. |  |
| * 1. Memory management      1. Proper allocation of memory      2. Free of memory leaks |  |
| * 1. makefile      1. clean: target      2. properly compiles program without warnings or errors |  |
|  |  |
| 1. Program Functionality | **+60/60** |
| * 1. Correctly parses the trace file   2. Correct LRU functionality   3. Correct accounting of page faults & total accesses   4. Correctly calculates page-fault rate   5. Correctly outputs required data formatted for Microsoft Excel |  |
| 1. Analysis    1. Correctly builds graphs    2. Graphs are professionally formatted    3. Makes a clear recommendation    4. Recommendation backed up by discussion of simulation results    5. Demonstrates knowledge of domain    6. Professionally written       1. Free of spelling and grammar errors       2. No first person       3. Active voice       4. Clear and concise language | **+30/30** |
|  |  |
| **Penalties** |  |
| 1. Vague/Missing Documentation (5%) | **-0/5** |
| 1. Late Submission (25% cap/day) |  |
|  |  |
| **Total** | **100** |