

CS440/CS540 - Operating Systems

Text: Operating Systems Internals and Design Principles, 5th Edition
William Stallings

Class meets Monday-Wednesday, 5:30-6:45 CB15 room 101


Important notes: (1) Class attendance is your responsibility. If you miss a class meeting, I will not give 'replays' under any circumstances. If you miss a class, it is your responsibility to discover what was covered and what you missed. (2) Missing an exam will result in a grade of zero unless you make arrangements prior to the time the exam is given. (3) Early exams are possible for reasonable circumstances. (4) any form of cheating, plagiarism, or other academic misconduct will result in a grade of F for the course. This includes copying any assignments, reports, programs or exams that are given as work to be done and turned in for a grade in this course. Helping others cheat is also a form of academic misconduct. For more information see the UAB web site.

This is a cross-listed course with both graduate and undergraduate students enrolled. Graduate students must meet higher expectations both on the exams and on the final project. Each of the three exams will include a set of questions for graduate students only, undergraduate students may answer them but they will not be graded. To avoid the occasional "bad question" each exam will be scaled, but the scaling will be done separately for the graduate and undergraduate exams, so that a "bad question" will not reduce the exam average grade. The final project is described in detail at the bottom of this document, and the graduate/undergraduate expectations are listed. Note that the final project is normally handed out when the memory management part of the course is covered, but accreditation standards require that this be a part of the course syllabus. No questions about the final project will be answered until the memory management section of the course is reached.

The grading scale for undergraduate students is 85-100=A, 70-84=B, 60-69=C, 50-59=D, <50=F. For graduate students, the scale is 90-100=A, 80-89=B, 70-79=C, <70=F.

There are two additional requirements for this course, the completion of the pre-class exam and the CIS exit exam. All students must take the pre-class exam when , and all students must take the exit exam unless you have taken it previously (if you have already taken the exit exam you will be given credit as though you take it this semester.) Students meeting both of the above requirements will receive an additional 40 points added to their total points for the semester. Students missing either or both of these exams will receive zero extra points. The net result of this is that failing to complete both exams will lower your semester grade by one letter.

Office Hours: Tuesday-Thursday 3:00pm-5:30pm



General Outline:

Chapter 1 Computer System Overview

Chapter 2 Operating System Overview

Chapter 3 Process Description and

Chapter 9 Uniprocessor Scheduling

Chapter 10 Multiprocessor and Real-Time Scheduling

Chapter 11 I/O management and disk scheduling

Exam (February 6)

Chapter 7 Memory Management

Chapter 8 Virtual Memory

Discussion of Final Project

Exam (March 15)

Chapter 12 File Management

Exam May 3, 4:15pm

Final Project

The course includes a final project that covers topics presented in the memory management segment of the course. The project has two components, (explanations will be given in class when the particular topic is covered):

(a) You must produce a reference string for a program that is well-behaved, typically-behaved, and poorly-behaved. These reference strings should span at least 100 different page numbers (0-99) and should be at least 5,000 elements long. You have some flexibility in defining the above terms (well-behaved, typically-behaved and poorly-behaved) but you are expected to explain your definitions in the write-up described below.

(b) You must write code to simulate at least three page replacement strategies including optimal, random, and some form of LRU/LRUA. You will use these programs on the reference strings produced in part (a) above to verify that the data from part (a) is reasonable.

Undergraduate students can either do part (a) and write one LRU/LRUA algorithm to process the different reference strings, or do part (b) and write a program to produce only one reference string that will be used on the three replacement strategies. Graduate students must do both part (a) and part (b).

The write-up must not exceed two pages of single-spaced typed text. An extra page or two of figures is acceptable, and you may also include raw data if you think it is appropriate. You must include all source code as an appendix. The write-up must cover your reference string generation, making sure that you explain your assumptions for each string you produce. It must also include a brief description of your page replacement programs, although you do not need to define terms already covered in class lectures. Finally, and this is the most important part of the project, you must take the results produced by running your replacement programs on your reference string data, and report your observations and conclusions. Pay special attention to anything that looks unusual or unexpected, and carefully explain what caused this behavior. This is the key part of the assignment. Writing the code to produce the reference strings and simulate the replacement strategies is not what the project is about. The project is about using that data to develop a better understanding about program behavior and how it affects various page replacement strategies. Do not turn in programs to do the above and think that is all that is required. That is just the starting point. For a hint, perhaps your "well-behaved reference string" produces far more page faults than you expected. Perhaps your definition of "well-behaved" is not what you thought it was. If you find one of those "things that make you go hmmm..." while doing this, you just encountered the point of the project. Now it becomes your task to make sure that you include this in your write-up.