CS3210 OS Design

Fall 2013, TuTh 12:05-1:25

Van Leer C241

Instructor: Matthew Wolf <mwolf@cc.gatech.edu>

Office: KACB 3323

Office hours: TBD

TA: Yanwei Zhang

Office hours: TBD

Prerequisites & Preparation: CS2200 is required. Some level of proficiency with the C language is also required, as the Linux kernel is completely written in C and most projects will require C programming. Familiarity with the command line environment in GNU/Linux (emacs, vi, gcc, make, etc.) will be an asset, although you can pick up much of it as you go along. A general exposure to OS concepts will also be useful, as will any previous experience with Linux and/or system administration.

Objectives: This course should be viewed as the "lab introduction" in the OS part of the platforms thread. The issues involved in the design (both internal and external) of operating systems will be covered in a formal manner, but this will be supplemented with hands-on practice with the internals of Linux as an example of a modern OS kernel.

Specifically, at the end of the course, you are expected to have developed the skills to design, implement, evaluate, and analyze operating system extensions in kernel and user space to help address parallelism, memory management, and performance. You should also have developed knowledge and understanding of many of the key control systems and concepts used by modern operating systems, including but not limited to their application to the Linux kernel.

Text: *Operating System Concepts Essentials*, Silberschatz, Galvin and Gagne. 2011 ISBN 978-0-470-88920-6

Accounts: You will be using both "normal" Linux resources and some specialized environments where you will be able to modify the kernel directly. Access to the resources will be set up during the 2nd week of class. Additionally, students will be required to bring their laptops/tablets/netbooks/etc. to class periodically for the in-class activities.

Grading:

30%	Tests	2 @ 15% each
40%	Projects	10 @ 8% & 2 @ 15%
20%	Final	20%
10%	Participation	10%

Project Expectations:

Projects will be done in a team of 2 or 3 students. Students teams will need to register with the instructor and in the wiki by the Monday of the 3rd week of class. Any students not paired will be assigned to groups at that time. Note that teams

should be between students with equivalent expectations; do not mix letter grade and pass/fail, etc.

All students must do the first two projects. For student groups that already have experience in system development that are looking for more of a challenge/opportunity, it is possible to replace the"advanced" projects with a single longer-term project. There are a limited number of such projects available each semester, and students must contact the instructor to discuss the possibility no later than the start of the first assignment.

Projects will be graded based on an evaluation of the code produced, the write-up detailing the design decisions and choices, experimental performance numbers of the code, and a team demonstration of the code. Note that these components are not in order of importance. In particular, there is significant weight on the write-up; this class emphasizes the design of operating systems, and each team needs to work through and explain their design process.

Communications Goals: Despite the highly technical focus of much of the content we will be covering in this class, it is critically important that students make substantial efforts towards clear and concise writing and oral presentations. Technical brilliance is not sufficient in the real world; you still need to be able to explain your decisions and their implications in a clear fashion. The same applies in 3210. Points may be deducted from project reports which are not up to the standards, and the final project grade will have a specific portion of the grade awarded based on the oral presentation of the results.

Team Etiquette: Working in a team is an important part of every day life and developing these skills is expected of all class members. We expect that most team issues will be able to be resolved without any intervention by the instructor or TA. However, we recognize that some issues may best be addressed by re-arranging teams and/or assigning different grades to different members of the team. Bring any such problems to the attention of the instructor **as far before the due date** of any relevant project as possible – there are limits to what can be done to improve a bad grade after the fact.

Participation: In addition to expected elements such as attending class or participating in discussions, the participation grade also includes regular short writing assignments. These quick feedback exercises are to better focus students on key concepts in the lecture and to provide the instructor with targetted feedback. Some of these will be in-class, but many of them will be administered through the t-square system. It is the student's responsibility to log into t-square within the allocated time for the quiz.

Honor Code: This course is conducted under the Georgia Tech Honor Code. In particular, although projects will be conducted in teams, members from different teams should not collaborate or share code with one another. Tests and in-class quizzes are individual efforts and are covered under the normal expectations.

Public Repository Contro:. Keeping your projects in a version control system, such as svn, hg, or git, is highly recommended for all class projects. Georgia Tech and the College of Computing make available a number of resources for privately hosting code repositories, and many external/cloud services have private hosting capabilities as well. However, externally hosting those repositories in places that are world readable, or readable by anyone other than class teammates and instructors, is considered to be a violation of the GT Code of Conduct, as it constitutes undue sharing of work. If you have any questions about acceptable solutions, please ask your instructor.

Topics: This course will cover the following material, some in lecture format and some through directed labs.

- Operating Systems definitions
- Booting
- User and Application interfaces
  - Syscalls, interrupts
- Memory Control
  - Hardware/software coordination (32 vs 64 bit, PAE, caching)
  - Swapping, paging, segmentation (multicore)
- Process Control
  - Threads (multicore)
  - cpu scheduling (multicore)
  - process scheduling
- Storage Control
  - File systems
  - Storage device drivers
- Virtualization/Protection