CMSI 387

Operating Systems

http://myweb.lmu.edu/dondi/spring2007/cmsi387

Spring 2007 — Doolan 222
TR 10:50am–12:05pm, 3 semester hours
Office Hours: TR 3–6pm or by appointment

John David N. Dionisio, PhD
e-mail: *dondi@lmu.edu,* AIM: *dondi2LMU*
Doolan 106; (310) 338-5782

Course Objectives

The primary objective of this course is to master the fundamental concepts behind modern operating systems through a comparative study of real-world systems. Understanding conceptual issues and mechanisms on their own, without confusing them with a particular operating system's specific policy, implementation, or interface, is crucial to being able to learn, use, and control any system effectively and quickly.

Course Requirements

Programming proficiency in a systems-level language, typically C; a prior course in Computer System Organization (LMU CMSI 284 or equivalent). Familiarity with Java, shell scripting, and system administration is also beneficial.

Materials and Texts

- Abraham Silberschatz, Peter Baer Galvin, and Greg Gagne. *Operating System Concepts*, 7th edition. John Wiley & Sons, 2005. — *a.k.a. "SGG"*
- Assorted handouts, articles, and sample code to be distributed throughout the semester.

The following text is recommended and not required — but it *will* fill in a lot of specific details:

- Daniel Pierre Bovet and Marco Cesati, *Understanding the Linux Kernel*, Third Edition, O'Reilly Media, 2005.

Additional information is also available on the Web; do not hesitate to look for further sources of information regarding the concepts, techniques, tools, and paradigms that we will discuss.

Course Work and Grading

Graded coursework consists of homework (25%), 1 midterm (25%), 1 kernel project (25%) and 1 final exam (25%). Letter grades are determined as follows: ≥ 90% gets an A– or better; ≥ 80% gets a B– or better; ≥ 70% gets a C– or better. The instructor may curve grades upward based on qualitative considerations such as degree of difficulty, effort, class participation, time constraints, and overall attitude throughout the course. Grades are never curved downward.

Homework

Homework consists of questions, exercises, and programming assignments to be given throughout the semester. Homework is where you can learn from your mistakes without grading penalty: if you do the work and submit it on time, you will get full credit, regardless of correctness. What goes around comes around: the effort you put into your homework pays off in the tests and the kernel project. The homework submission deadline is always the beginning of class on the designated due date; the due date is encoded in the homework number. Submissions after the deadline receive half credit, period. Extra credit homework may be assigned; fulfilling this is counted on top of the 25% allocation of homework to your final grade.

Tests

The midterm is initially scheduled for February 15. The final exam is scheduled for May 1. All tests are open-paper-everything; no sharing. "Open computer" might also be allowed depending on the scope, subject matter, or circumstances. You may neither solicit nor give help while the exam is in progress. Late and/or missed tests are handled on a case-to-case basis; in all instances, talk to me about them.

Kernel Project

The general and theoretical principles behind the material will find specific application in a Linux kernel project. The project may either be a demonstrable nontrivial modification or enhancement to the Linux kernel, or a system program that interacts with the kernel in an interesting and significant manner.

The kernel project will be graded according to the following criteria:

1. *Design (30%):* How good is the overall structure of the code? Is it clear, flexible, and easy to maintain? Is it elegant or innovative? How well does it apply the principles of "separation of concerns" and "one change, one place?"
2. *Functionality (30%):* How well does the code work? Does it fulfill requirements? Are its results accurate or correct? Does it perform its tasks in a reasonable amount of time? How well do unit tests validate the code?
3. *Naming (20%):* Are program entities — classes, subroutines, variables, etc. — clearly and consistently named? Do their names correspond to their functions and roles?
4. *Comments (15%):* Are comments provided where appropriate? Are they clear and well-written? Does the code take advantage of any special support for comments provided by the project language or platform (e.g., JavaDoc)?
5. *Version control (5%):* Is the code committed at reasonable intervals? Are milestones appropriately tagged? Are adequate descriptions of provided in the commit logs?

The kernel project is due on May 1. Late projects will not be accepted.

Combination Project Option

If you are enrolled in both CMSI 387 and CMSI 371 this semester, you have the option of using the same code base for the projects in both courses. In order to do this, your project must fulfill the requirements of *both* projects. That is, your project must involve interactive graphics using OpenGL *in addition to* being a demonstrable and nontrivial modification or enhancement to the Linux kernel or a system program that interacts with the kernel in an interesting and significant manner. Combination projects will follow the deadlines specified for the CMSI 371 graphics project.

Attendance

I am not a stickler for attendance, but I do like having a full class. Remember that the university add/drop with 100% refund deadline is January 12. The deadline for withdrawal or credit/no-credit status is March 16.

University Policy on Academic Honesty

Loyola Marymount University expects high standards of honesty and integrity from all members of its community. Applied to the arena of academic performance, these standards preclude all acts of cheating on assignments or examinations, plagiarism, forgery of signatures or falsification of data, unauthorized access to University computer accounts or files, and removal, mutilation, or deliberate concealment of materials belonging to the University Library.

Course Schedule

This schedule may change based on the actual ebb and flow of the class; deadlines, exams, and university dates (italicized) are less likely to change than lecture topics.

January	Operating systems overview; process management
January 12	*Add/drop deadline for full refund*
February	Scheduling and synchronization; memory management: main memory, virtual memory
February 15	Midterm
March	Storage management: file systems, mass-storage organization
March 5–9	*Spring break; no class*
March 16	*Withdraw/credit/no-credit deadline*
April	I/O systems; security; additional topics (time permitting)
May 1	*Final exam, 11am; kernel projects due*

You can view the class calendar on the Web at *http://ical.mac.com/dondi/LMU*. If you have an iCalendar-savvy client (i.e., Mozilla Calendar, Ximian Evolution, KOrganizer, Apple iCal, etc.), you can subscribe to the class calendar at *webcal://ical.mac.com/dondi/LMU.ics*. On-the-fly updates and adjustments to the class schedule will be reflected in this calendar.