Course Objectives
The primary objective of this course is to master the fundamental concepts behind modern operating systems, with some 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
• Assorted handouts, articles, and sample code to be distributed throughout the semester.

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.

Tests
The midterm is initially scheduled for February 26. The final exam is scheduled for May 8. 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 provided in the commit logs?

Kernel project deliverables are due on **May 8**. Late projects will not be accepted.

**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 18**. The deadline for withdrawal or credit/no-credit status is **March 14**.

**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.

<table>
<thead>
<tr>
<th>Month</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>January</td>
<td>Operating systems overview; process management</td>
</tr>
<tr>
<td>January 18</td>
<td>Add/drop deadline for full refund</td>
</tr>
<tr>
<td>February</td>
<td>Scheduling and synchronization; memory management: main memory, virtual memory</td>
</tr>
<tr>
<td>February 26</td>
<td>Midterm</td>
</tr>
<tr>
<td>March</td>
<td>Storage management: file systems, mass-storage organization</td>
</tr>
<tr>
<td>March 3–7</td>
<td>Spring break; no class</td>
</tr>
<tr>
<td>March 14</td>
<td>Withdraw/credit/no-credit deadline</td>
</tr>
<tr>
<td>April</td>
<td>I/O systems; security; additional topics (time permitting)</td>
</tr>
<tr>
<td>May 8</td>
<td>Final exam, 11 am; kernel projects due</td>
</tr>
</tbody>
</table>

You can view the class calendar on the Web at [http://ical.mac.com/dondi/LMU](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](webcal://ical.mac.com/dondi/LMU.ics). On-the-fly updates and adjustments to the class schedule will be reflected in this calendar.