

## Synopsis:

Nowadays it is well recognized that the master control program for a computer system should be designed from the outset to provide for *extensibility*: that is, in order to avoid early obsolescence, the system software must be able to 'evolve' during its lifetime, not merely to allow for the correction of 'bugs', but also to accommodate the use of new hardware devices as they become available, and to support performance requirements that may not have been envisioned at the time the operating system software was initially being designed.

Two mechanisms for 'extensibility' are provided by the Linux operating system. First, its source code is made freely and publicly available, so that this code can be altered by a knowledgeable systems programmer, then recompiled and reinstalled, and the resulting system rebooted, to provide any desired upgrades or bug fixes that an existing system might need. Second, it is possible for programmers to create entirely new pieces of code (called 'installable kernel modules') that can be independently compiled and then inserted into a system while it is running, without modifying any of the original code or shutting down the machine for a system reboot. Moreover it is just as easy to remove these new 'modules' as desired, thereby allowing the system revert to its former behavior.

Both of these Linux extensibility mechanisms offer invaluable opportunities for computer science students to study in detail the inner workings of a 'live' system. Accordingly this course is all about Linux kernel modules! It will:

- focus on the Linux operating system for Intel Pentium processors
- assume familiarity with the C/C++ programming language, with 80x86 machine architecture, Unix operating system commands, and standard data-structures and algorithms (e.g., CS 112, 210, 245, 326)
- be open to USF graduate students in computer science (and to qualified undergraduates or non-degree students with instructor permission)

Planned course topics include: loadable kernel modules; shared libraries; thread and memory management; asynchronous input/output; The '/proc' and 'ext2' file systems; character-mode device-drivers; system data-structures and algorithms; kernel threads, timers, and wait-queues; interrupts, exceptions, and system-calls; and IA32 Symmetric Multiprocessing architecture.

The course consists of lectures, readings, discussions, demonstrations, and programming exercises. Some are conducted as in-class experiments, while others are completed as independent projects outside of class.

## Learning Outcomes:

- You will be able to read and write code-modules for an operating system
- You will be able to implement customized extensions to the Linux kernel
- You will be able to craft your own tools that let users control their PCs
- You will be able to identify system features which impact performance

## Instructor:

Dr. Allan B. Cruse, Professor of Computer Science and Mathematics  
Harney Science Center - Room H-212 Telephone: (415) 422-6562  
Office Hours: Mon-Wed-Fri 12:45-1:30pm, Tues-Thurs 6:30-7:15pm  
Email: [cruse@usfca.edu](mailto:cruse@usfca.edu) Website: <http://cs.usfca.edu/~cruse/cs635/>

## Textbooks:

Bovet and Cesati, *Understanding the Linux Kernel (Third Edition)*,  
O'Reilly Media, Incorporated (2006), ISBN 0-596-00565-2

Corbet, Rubini, and Kroah-Hartman, *Linux Device Drivers (Third Edition)*,  
O'Reilly Media, Incorporated (2005), ISBN 0-596-00590-3

## Classroom Facility:

The course is scheduled to meet Tuesdays and Thursdays, 7:30-9:15pm,  
in the Michael D. Kudlick Interactive Computer Classroom (HRN-235).  
Students will need to have individual computer accounts set up for access  
during classes.

## Exam Dates:

Exam I will be on Tuesday, October 2.  
Exam II will be on Thursday, November 15.  
Final Exam will be Tuesday, December 11 (7:30pm)

## Grading scheme:

Class Participation	25%
Programming Projects	25 %
Midterm and Final Exams	50%

*NOTE: Unprofessional conduct, such as an abuse of USF computer privileges (unauthorized access), or a violation of academic integrity (plagiarism or fraud), will result in the student receiving a failing grade.*