Class Number	**CSC 615.01 Spring 2013** MWF 1:10-2:00, TH 331 (date of last update: 1-12-13)
Class Title	Unix Programming
Instructor	**Professor Marguerite Murphy** Office Hours: MWF 11:30-12:00 (by appt), MWF 2:10-3:00 Office: TH 968 email: mmurphy@sfsu.edu Office telephone: 415-338-2261 URL: http://dbsystems.sfsu.edu/~mmurphy
Course Description	**Bulletin Copy:** Programming in a UNIX environment. Topics include regular expressions; utilities such as awk, sed, grep, csh, sh, ksh; system calls such as signals, sockets, POSIX IPC, and POSIX threads; kernel internal structures. ***This course is a senior elective in the area of Operating Systems and Distributed Processing.*** **Spring 2013 Revised Course Description**: Introduction to Linux system programming (at the upper division level).Review basic operating systems concepts and their realization in the Linux 2.6 kernel. Lectures and programming exercises will introduce advanced C programming techniques, shells, modules, system calls and an introduction to the data structures and algorithms used to implement major subsystems of the Linux 2.6 kernel. Although this course will be taught in lecture format, questions and (limited) classroom discussion are encouraged!
Prerequisites	CSC 415 with grade of C or better or consent of instructor. You should be familiar with using the Unix programming environment (e.g. TheCity machine at SFSU), and have a good working knowledge of the C programming language as well as the basics of concurrent programming (processes, threads, synchronization)
Text (required)	1. Love, Linux Kernel Development, 3rd Edition, Addison-Wesley, 2010 (available as a SFSU Library eBook, primary reference) 2. Any standard upper division operating systems textbook (to review basic concepts). 3. Other supplementary class materials will be posted to the class web site and/or available over the Internet
Text (optional)	Harbison & Steele, "C: A Reference Manual", Fifth Edition,

	Prentice Hall, 2002. Stevens and Rago, "Advanced Programming in the Unix Environment", Addison-Wesley, 2008. Kerrisk, The Linux Programming Interface, No Starch Press, 2010 (strongly recommended supplementary reference detailing the Linux system call interface)
Course Web Site	http://dbsystems.sfsu.edu/~csc615 (password required)
Reader	There are no published lecture notes for this class. Lecture notes will be available download from the course web site during the semester.
Course Objectives and Role in Program	The objectives of this course include: • Introduce the student to Unix/Linux kernel programming techniques • Teach advanced C systems programming and debugging techniques in a Unix/Linux environment • Review basic concepts covered in the core Operating Systems course prerequisite as they are realized in the Linux platform • Discuss correct synchronization techniques for both application programs and kernel code running on uniprocessor as well as multiprocessor (SMM) platforms Students will implement and evaluate several small application programs utilizing low level Unix system calls, then work through a series of progressively more difficult kernel programming tasks, culminating with the design and implementation of correctly synchronized kernel module code. The knowledge of advanced programming techniques (including correct synchronization) and the Linux platform play an important role in developing our students into skilled professional programmers.
Learning Outcomes	At the end of this course students will be able to: • Write correct and well documented advanced C code using low level Unix/Linux system calls that is demonstrated to execute correctly • Know where to look for platform specific programming information and be familiar with reading and using man page information as well as other standard reference materials • Clearly and accurately explain design decisions in written program documentation • Be familiar with the mechanics of Unix/Linux kernel

	programming: installing/ configuring the Linux kernel from source and building a useful personal programming environment; modifying the kernel code and recompiling/testing/debugging the new kernel version; designing, installing and testing and debugging a new Linux kernel module and possibly a new system call. • Be able to design and implement simple, but efficient, concurrent process and thread based applications.
Lecture Topics	1. Using system calls to observe the Linux kernel 2. Basic Linux kernel installation & configuration 3. Kernel data structures & memory allocation 4. Interrupt Handlers & keeping track of time 5. Kernel Synchronization 6. Module programming & debugging 7. Kernel implementation of filesystems & device I/O 8. Kernel implementation of virtual memory 9. Kernel implementation of scheduling
Assignments	There will be approximately 10 substantial C programming assignments, illustrating and extending the topics covered in lecture. In addition, there will be a short in-class quiz after each major topic has been covered in lecture and a comprehensive final examination at the end of the semester. The final exam will **only** be given during the time printed in the Schedule **(*Friday, May 24, 10:45-1:15*)** in the regularly scheduled classroom.
Grading	Programming/Homework Assignments: 40%, Quizzes: 25%, Final Exam: 35%