

Course Information and Standards

INTRODUCTION TO OPERATING SYSTEMS

91.308

University of Massachusetts Lowell
Department of Computer Science
Spring Semester 2012

Time and location: MWF 11:00 – 11:50 AM, Olsen 402
Instructor: Prof. W. Moloney
Office and telephone: Olsen 222, 978-934-3640 (x3640)
Office hours: MWF 10:00 - 10:50, W 1:00 – 2:00, and by appt.
Email: bill@cs.uml.edu
Web site: www.cs.uml.edu/~bill/cs308

1. Course Description:

This course introduces and develops the major components of operating systems, including the process and thread abstractions, concurrency and synchronization mechanisms, deadlock management strategies, processor allocation, memory management, I/O device and file management, and distributed processing. The course also presents techniques for operating system design, implementation, and evaluation.

2. Prerequisites:

91.204, 91.305

3. Required Text:

Modern Operating Systems, 3rd Ed., by A. Tanenbaum, Prentice-Hall, 2008
ISBN 0-13-601919-6

4. Grading:

Final grades will be based as follows:

Type	Number	Weight
Hour Exams	2	40 %
Final Exam	1	20 %
Programming Assignments	approx. 6	40 %

The programming assignments are to be coded using the 'C' or 'C++' programming language.

5. Lateness:

For each **class day** an assignment or project is late, **10 % of the total points** will be deducted from the points received. This will continue for **five class days**, after which the assignment or project may be submitted at any time up to and including the last day of scheduled class and will receive a maximum of half the total credit.

5. Academic Dishonesty:

In this course, **all** work is to be each student's own. Students should therefore be familiar with the University's rules on academic dishonesty, which can be found in the *Bulletin of Undergraduate Studies* and in the *Schedule of Classes*. In particular, **plagiarism** will not be tolerated! Any student caught plagiarizing another's work will automatically receive a grade of **F** for the course. If you are unsure as to what constitutes plagiarism, it is your responsibility to check with the instructor. Other forms of dishonesty will result in similar actions. You may collaborate with your classmates on the design and results of the programs you will write in this course, but each student must implement these programs alone. Submission of shared student code is not permissible, and will result in a grade of **F** for the course. Help files are typically provided for each programming assignment, and students are encouraged to cut and paste useful code from these help files into their assignment submissions, but all other code must be the specific work of each student.

6. Topical Outline:

The topics covered in this course include the following:

- What is an operating system
- The evolution of operating systems
- The process model
- Systems and system states and transitions
- Processes and address space layouts
- Computational progress and process/thread organization
- Process and operating system interfaces
- System call and exception based kernel access
- The need for process/thread synchronization
- Synchronization techniques and mechanisms
- Process scheduling and dispatching
- General address space issues and memory management introduction
- Process/thread sharing of physical memory
- Virtual memory paradigm
- Fetch, placement and replacement requirements
- Page replacement algorithms
- Page fault performance issues and modeling techniques
- General design issues for paging systems
- UMA, NUMA, and NORMA system architectures
- Introduction to file systems
- Name space issues
- File system security and protection mechanisms
- General device level IO control
- Device driver models
- Synchronous vs. asynchronous driver interfaces
- Resource management and deadlock
- Requirements for deadlock
- Prevention, avoidance, and detection and recovery deadlock strategies
- Introduction to distributed systems
- General distributed system design issues
- Communication in distributed systems
- The general client-server model
- The remote procedure call paradigm
- The need for data consistency between heterogeneous systems