

# CS 385 - Operating Systems Concepts and Design

## Course Policies – Fall 2011

**Course Objectives:** At the end of this course, students will be able to:

- Explain the basic tasks and structures of an OS, and how they relate to one another.
- Implement algorithms for process management, including CPU scheduling, process synchronization, and deadlock avoidance, and to analyze the trade-offs of different algorithms and parameter adjustments.
- Compare different methods for memory management, including timing analyses and the pros and cons of different approaches to memory management.
- Analyze storage management techniques, including file systems, mass storage, and I/O systems.
- ( Time permitting ) Evaluate the protection and security of a system, and explain the difference between protection and security.

**Instructor:**

John Bell, jbell@cs.uic.edu  
http://www.cs.uic.edu/~jbell  
1035 SEO, 413-9054  
Office Hours: May be MWF 12:30 – 2:00  
See web for details  
Open Door policy during other times.

**Teaching Assistant:**

Weixiang Shao, wshao4@uic.edu, 1309 SEO, 413-2391  
Office Hours: See web site for details.

**Pre/Co-requisites:**

CS 201 and either CS 366 or ECE 267 **If you are an undergraduate student who does not have the necessary pre-requisites, DROP THE CLASS NOW.** Otherwise you will be automatically dropped later, when it will be too late to sign up for anything else instead

**Credits:**

3

**Course Web Page:**

http://www.cs.uic.edu/~i385

**Textbooks:**

**Required:**

- Abraham Silberschatz, Peter Baer Galvin, and Greg Gagne, “Operating System Concepts, Eighth Edition”, Wiley, 2008, ISBN 978-0-470-12872-5.

**Officially Recommended:**

- Marc J. Rochkind, “Advanced UNIX Programming”, Second Edition, Addison Wesley, 2004, ISBN 0-13-141154-3.
- Mark Allen Weiss, “C++ for Java Programmers”, Pearson Prentice Hall, 2004.

**Other Recommendations:**

- Tanenbaum, Andrew, “Modern Operating Systems”, Third Edition”.
- Tanenbaum and Woodhul, “Operating System Design and Implementation”, 3<sup>rd</sup> Edition.
- Loukides, Mike and Andy Oram, "Programming with Gnu Software".
- Oram, Andrew and Steve Talbott, "Managing Projects with make".
- Stevens and Rago, “Advanced Programming in the UNIX Environment”, 2<sup>nd</sup> Edition.

### **Planned Schedule:**

The general schedule planned as of this writing is to cover the first 15 chapters of the required textbook in 15 weeks, at a steady rate of approximately one chapter per week. If this planned schedule turns out to be overly ambitious, then chapters 15 and/or 14 may be omitted.

Weeks	Dates	Part - Topic	Chapters	Notes
1 - 2	23 Aug - 2 Sep	I - Overview	1 - 2	
3 - 7	6 Sep - 7 Oct	II - Process Management	3 - 7	
8 - 9	11 Oct - 21 Oct	III – Memory Management	8 - 9	<b>MIDTERM I WEDNESDAY 12 OCT, 6-8 P.M.</b>
10 - 13	25 Oct - 18 Nov	IV – Storage Management	10 - 13	
14 - 15	22 Nov - 2 Dec	V – Protection and Security	14 - 15	<b>NO CLASS NOV 24 – 25 THANKSGIVING</b>
<b>Finals Week</b>	<b>9 Dec</b>		<b>All of the Above</b>	<b>Final Exam Friday 1:00 – 3:00</b>

**MIDTERM EXAM I: WEDNESDAY 12 OCTOBER, 6:00 – 8:00 P.M.**

**NO CLASS NOVEMBER 24 – 25, THANKSGIVING**

**FINAL EXAM: FRIDAY 9 DECEMBER, 1:00 – 3:00 P.M.**

( Course listed **FIRST** in Timetable has precedence. )

### **Exam Policy:**

- All mid-term exams will be given at night, so that students will have ample time to complete the exam.
- Exams will be written so that the average student will be expected to finish in about an hour, so time constraints will not be a factor.
- Any exam conflict needs to be brought to the instructor's attention for resolution **before** the regularly scheduled exam. Requests for make-up exams after the regularly scheduled exam will not normally be granted.
- Exams will be closed-book. One crib sheet will be allowed, no larger than 8.5x11 inches, double sided, **handwritten**.
- All exams are cumulative, with emphasis on material not previously covered on exams.
- All material covered in class or in assigned reading or which should have been learned in the course of completing homework is fair game on exams. No more specific information will be provided as to exam content.
- Anyone who fails to stop working on the exam when time is called will receive a minimum of a 5 point late penalty.

## Grading Policy:

Numerical scores will be based upon the following contributions:

( 2 ) Exams ( 1 midterm, 1 final )	20 points each
Programming Assignments	10 points each
Quizzes	as needed
Total:	Normalized to 100 point scale

Unless otherwise specified, all programming & homework assignments will carry equal weight. The exact number of such assignments will be determined as the course progresses.

Conversion of numerical scores to letter grades is a serious business, requiring careful consideration of every student's **complete** semester performance, **and will not be considered until all scores are compiled at the end of the semester.**

**There are no predetermined grade guarantees.** However it is expected that grades will follow the general pattern given below. **Regardless of the numerical score, it will not be possible to pass this course without passing the exams, particularly the final exam, and completing most of the homework assignments.**

<u>The grade break for:</u>	<u>will probably be somewhere around:</u>
A / B	90
B / C	80
C / ?	70

Note that the final grade breaks may be either slightly below **or slightly above** the numbers given here.

## Homework Grading Policies

Specific homework grading guidelines will be determined on a case-by-case basis. For programming assignments, it is expected that the points will break down roughly as follows:

Program compiles and runs ( on CS Linux computers )	25%
Program handles simple, straightforward situations:	25%
Program handles more advanced and/or tricky situations:	25%
Program is efficiently written using good programming style:	25%

### Notes:

1. A program that does not compile and run cannot, by definition, solve any problems. However it may still be efficiently written using good programming style. Conversely, programs that happen to solve all problems may still be poorly written.
2. Graders may, at their discretion, give partial credit for any of the above categories. They are not, however, obligated to do so.

## Special Considerations

- All programs must be turned in using Blackboard.
- Each program must be accompanied by a readme file and a makefile.
- Programs must compile and run properly on the CS Linux computers to receive full credit. **Programs which run on other systems** ( e.g. Microsoft's Visual Studio ) **but which do not run on the CS system may be given a zero, at the grader's discretion.**
- **Graders are under no obligation to grade any program that does not compile.** However, they may choose to do so at their own discretion.
- The purpose of the readme file is to explain what the program does and how it does it, so that any reasonably competent programmer can easily understand it. It is in your best interest to make the grader's job as easy as possible by submitting a readme file that is both easy to read and which clarifies your code.
- Assignments submitted after the due date and time, but within 24 hours, will be assessed a 20% penalty. No assignments will be accepted more than 24 hours after the due date and time.
- All appeals for grading errors, no matter how justified, must be submitted within two weeks after the graded assignments are returned. No appeals for regrades will be heard after that time.

## Academic Integrity:

Students are encouraged to study together and to help each other learn. When one student teaches another, both benefit from the experience. However, it is a strict violation of class and university policy for any student to hand in any work that is not 100% their own creation. Therefore:

- All work on all exams and all homework must be individually performed by the student whose name appears on the paper.
- No student may give any other student any portion of their code, either written down, electronically, or through any other means.
- Students are responsible for safeguarding the integrity of their work. This includes but is not limited to changing their passwords and keeping their computer accounts secure.
- Direct copying of code from any textbook or other source is strictly forbidden.
- Students may discuss homework problems, including background concepts and general solution strategies, but they are forbidden from discussing or sharing specific solutions. In particular, **it is forbidden for any student to show any other student any portion of their computer programs or homework solutions for any reason, including debugging assistance.** This means you must hand in your own homework. **You are not allowed to see anyone else's work, or show your work to anyone.** Failure to protect the privacy of your work may be a violation.
- All submitted programs will be analyzed using MOSS, to identify any unacceptable similarity to other students' code or to previous or published solutions if applicable.
- In the case of extreme discrepancy between homework performance and exam performance ( e.g. very high homework scores and very low exam scores ), the instructor shall determine which scores more accurately reflect the students' true work.
- All violations will be reported directly to the Office of Student Judicial Affairs, <http://www.uic.edu/depts/sja>. First violations will be penalized with zero on the relevant assignment(s) **and** a penalty equal to the value of the assignment(s), **for all parties involved in the transgression.** Second or more serious violations may result in a failing grade, probation, suspension, or expulsion from the university. The violation will also be recorded on the permanent records of all students involved.