

COURSE DESCRIPTION

Department and Course Number	CMPS 455	Course Coordinator	Andrew C. Lee
Course Title	Operating Systems	Total Credits	3
URL		Semester	Fall/Spring Semester
		hours	lectures 3 hours per week for 14 week

Current Bulletin Description

Textbook

Silberschatz et.al , Operating System Concepts, John Wiley, 7th edition.

References

See the links as posted on the Moodle site.

Course Goals

1. Study the issues, designs, and tradeoffs that face the designer of an operating systems.
2. Study the underlying concepts and algorithms used in modern operating systems.

Course Objectives

Upon successful completion of this course, students will

1. have a better understanding of the basic concepts used in the design of an operating system for a computing device.
2. have an experience in kernel development via an simulated operating system (Nachos).
3. be able to explain the choice of data structures and algorithms used in operating systems development.

Prerequisites by Topic

1. Experience in project planning, development and management.
2. Experience in handling large pieces of code.
3. Knowledge of C/C++ and Unix.

Major Topics Covered in the Course

1. Introduction to Operating Systems: history, basic ideas and terminology (1 week)
2. Computer Science Overview (1 week)
3. Process Management I: Description and Control (1 week)
4. Process Management II: Threads and SMP (1 week)
5. Synchronization I: Concurrency (2 week)
6. Synchronization II: Deadlocks (1.5 week)
7. Memory Management I: Basic Ideas (1.5 week)
8. Memory Management II: Virtual Memory (1.5 week)
9. Uni-Processor Scheduling (1 week)
10. File Systems Management (2.5 week)

11. Advanced Topics: examples include Operating Systems Security and other contemporary topics (1 week)

Laboratory projects (specify number of weeks on each)

After discussions in our faculty meetings, it was decided that we will continue to use the Nachos instructional operating system as the basis of our projects. This project changes each semester. It is usually divided into three phases.

Phase 1: It is an individual project. The objective is to help the students to get familiar with the use of Nachos. The subject usually includes the use of threads and solutions to classical synchronization problems such as the dining philosophers problem and the producer consumer problem.

Phase 2: It is usually designed as a team project. It involves more in-depth work using Nachos. For example, they may be asked to implement some typical system calls. Other programming tasks usually involve non-trivial modification of the existing codes and a careful design before implementation is necessary. Each group will need to submit a short design document as well.

Phase 3: More in-depth implementation work is required. For example, it may include the implementation of a simple file system or the implementation of several page replacement algorithms to support virtual memory. Each group will need to provide a demo and explain their implementation or problem they encounter. Each group are required to submit a final report.

Oral and Written Communications

Written: Quizzes (usually 5); Midterm and Final. In some semesters, the final is a take home final. The student is required to summarize and critique an article. It is usually related to contemporary issues in operating systems design.

Oral: Each group is usually required to present their term project work in a demos setting.

Social and Ethical Issues

N/A

Theoretical Content

To meet the course goals, we emphasize the design of algorithms and the choice of data structures for the implementations. Students will be able to compare and contrast the choice of data structures and algorithms that will lead to differences in performances.

The approach to this course is mainly theoretical. The Project helps the students to understand the implementation side via the simulated operating systems. Contemporary topics, such as file access control and security are usually included.

Problem Analysis and Solution Design

The quizzes, written midterms and finals include questions that require good problem solving skills. For instance, they may need to construct good examples to show that two algorithms behave differently.

In the project, there are questions that ask them to solve a problem empirically. For example, they may be asked to vary parameters inside a standard algorithm to observe differences in operating systems behavior.