CS8803-O02: Graduate Introduction to Operating Systems

Instructor

Ada Gavrilovska, School of Computer Science, College of Computing

Goal

This course teaches operating system abstractions, mechanisms, and their implementations, including for concurrency (threads) and synchronization, resource management (CPU, memory, I/O), and distributed services.

Course Description

This is a graduate-level introductory course in operating systems. The course teaches the basic operating system abstractions, mechanisms and their implementations. The core of the course contains concurrent programming (threads and synchronization), inter process communication, resource management (CPU, memory and I/O) and an introduction to distributed operating systems.

Learning Objectives

As part of this course students will:

- become familiar with a broad range of operating systems principles and implementation;
- gain knowledge via implementation and evaluation of various OS aspects in a practical manner, through a sequence of programming assignments; and
- understand the rationale behind current design and implementation decisions in modern OS's (like Linux) by considering the historic evolution of various OS constructs.

Course Educational Outcomes

After completing this course, students will:

- understand the internal design, tradeoffs, and implementation issues concerning key operating system components, such as memory management, scheduling, synchronization mechanisms, file systems, virtualization, and systems software support for distributed services;
- understand the evolution of the systems software stack in response to changes in hardware capabilities and workload demands;
- discuss and critique research papers on these topics;
- design, implement and evaluate systems software components dealing with multithreading, synchronization and other aspects of concurrency, in a low-level programming language such as C;
- design, implement and evaluate systems software mechanisms for enabling inter-process interactions, via shared memory and remote procedure call interfaces; and
- design, implement and evaluate distributed services and distributed state management algorithms.

Recommended Textbooks

No textbook is strictly required, however, it is recommended that students have access to an operating systems textbook. Select examples include:

- *Operating Systems Concepts*, or (the Essentials version), Silberschatz and Galvin.
- *Modern Operating Systems*, A. Tanenbaum.
- *Operating Systems: Three Easy Pieces*, Arpaci-Dusseau
  (a free version available at: http://pages.cs.wisc.edu/~remzi/OSTEP/)

Some other useful texts include:

- *Distributed Systems*, Tanenbaum and VanSteen. Prentice Hall.
- *Multithreaded Programming with Pthreads*, Lewis and Berg. Prentice Hall.
- *Pthreads Programming*, Nichols, Buttlar, Farrell. O'Reilly.

A number of papers will be made available online.

Grading

- 55% Exams: Midterm 25%, Final 30%. All exams are closed-books, closed-notes.
- 45% Programming Assignments. Programming assignments are due by midnight (11:59pm) of the due date.
- P/F students must have passing grade in both components to pass.

Academic Integrity

Academic dishonesty will not be tolerated. This includes cheating, lying about course matters, plagiarism, or helping others commit a violation of the Honor Code. Plagiarism includes reproducing the words of others without both the use of quotation marks and citation. Students are reminded of the obligations and expectations associated with the Georgia Tech Academic Honor Code and Student Code of Conduct, available online at www.honor.gatech.edu.

Learning Accommodations

If needed, we will make classroom accommodations for students with documented disabilities. These accommodations must be arranged in advance and in accordance with the Office of Disability Services (http://disabilityservices.gatech.edu).

Excused Absence Policy

http://www.catalog.gatech.edu/rules/4/

Outline of Topics

1. Operating Systems Overview
2. Process and Process Management Overview
3. Multithreading

    - Multithreading models and patterns; programming with threads; threads management and implementation; multithreading vs. event-based systems
4. Scheduling
    - Algoritms and frameworks; case-studies; multicores
5. Memory Management
    - Virtual and physical memory management, shared memory and IPC
6. Synchronization
    - OS support for synchronization, comparison of spinlock implementation algorithms
7. I/O
    - Device management, synchronous and asynchronous I/O interfaces, file systems
8. Virtualization Technology
9. OS Support for Remote Services
    - Remote procedure calls, Sun RPC, RMI
10. Distributed Services
    - Distributed file systems, NFS, caching
11. Distributed State Management
    - Distributed shared memory, consistency models
12. Cloud Computing Technologies Overview