Introduction to Operating Systems and Concurrent Programming

EECS 338, Spring 2016

Eamon Johnson

About This Course

- A *core* course
  - Is this a hands-on "OS building" course that involves kernel-level programming?
    No: That requires extensive C and X-86 programming knowledge.

- Course objectives
  1. To provide a grand tour of major operating system concepts:
     OS structures, CPU scheduling, deadlocks, memory management, virtual memory, thread management, Virtual Machines **(~10 weeks)**
  2. To teach concurrent process/concurrent programming basics **(~4 weeks)**
  3. To provide familiarity with Linux, and Linux Systems Programming (via recitations and assignments)

- Class web page here.

What Do We Learn in This Course?

- **Many informally** (yet reasonably precisely (≠ formally)) **expressed OS concepts/terminology**.
  - Please read the book before coming to class: too many new concepts.
  - Knowledge tested via quizzes. **Basic concepts!**

- **Concurrent algorithms: Significant algorithmic component.**
  - Essential for computer scientists. See Jeff Dean's slides on the blackboard.
  - Knowledge tested via assignments and the midterm exam.

- **(Ubuntu) Linux (and C)-based programming via assignments.**
  - This is the only course in the CS curriculum that has Linux/Unix coverage.

Textbooks

- **Required Textbooks:** See the syllabus.

- **Recommended C book:** The C Programming Language, B.W. Kernighan and D. M. Ritchie, Prentice Hall, 1988

- See the "Resources" section of the course webpage for additional materials. For instance, **Interprocess Communication in Linux** provides important background knowledge for Linux system programming.

Lectures and Recitations-Attendance

- Lectures
  - You should attend them. No attendance taken.
  - Missed lectures? Go over the posted slides.

- Recitations
  - Once each week,
    - On an “as-needed” basis, there may be other lectures.
  - You should attend recitations.
    - help on assignments
    - Supplemental materials
  - Missed recitations? Talk to the recitation leader.

Topics Covered in Midterm Exam

- Concurrent Programming Basics (Ch 5 & Notes)
- Process Synchronization Algorithms (Ch 5 & Notes)
- Language Constructs for Concurrent Programming (Ch 5 & Notes)
- Concurrent programming Languages (Ch 5 & Notes)

Pace: Slow

- Algorithmic in nature; material is **not** in the textbook.
- Assignments related to this coverage:
  - Two algorithmic (paper-and-pencil) assignments;
  - Two concurrent programming assignments.
    - One concurrent process-based programming assignment
    - One multithread-based concurrent programming assignment.

Topics Covered in Quizzes (Tentative)

Some chapters in the textbook are FYI-only, and not covered in quizzes.

Quiz topics—This may change, depending on coverage:

- Intro, OS Structures, processes (Chs 1-3; Quiz 1)
- CPU Scheduling (Ch 6)
- Deadlocks (Ch 7)

  } Quiz 2-tentative

- Threads+ (Ch 4)
- Memory Management (Ch 8, Ch 9)

  } Quiz 3-tentative

- File systems and disk scheduling (Ch 11, Ch 12, and parts of CH 10)
- Virtual Machines (Ch 16)

  } Quiz 4-tentative

Pace: Fast

- Basic knowledge; simple algorithms (if any); material is in the textbook.
- No assignments related to this part.

Assignments

Assignments

- **Linux programming assignments**--C language. Yyour code needs to run on eecslinab machines. 4-5 assignments, one of which is about concurrent programming.

- **Concurrent programming algorithm assignments**—Paper-and-pencil (two) assignments.

- **You must work on the assignments to pass this course**:
  Coding is important: you must collect at least 20% out of 50% from assignments.

- **Missed assignment deadlines?**
  Only one assignment extension (Two 1-day extensions or one 2-day extension) in the semester.

- **No make-up quizzes**.

- **Teamwork?** Not allowed. Moss cheating detection software will be used.

Midterm and Assignments

- **Study old tests; solve them; then, study their solutions**.
  They will give you an idea what to expect in your midterm exam.

- **Study old assignments** and their solutions here; both concurrent process assignments and programming assignments.
  - Exams and algorithmic assignments evaluate your knowledge on class material.
  - Unix/Linux programming assignments improve your "Unix systems programming" skills, and evaluate your knowledge on recitation material.