Course Home Page and Syllabus

UTEP CS 4375: Theory of Operating Systems
(and sometimes EE 4374: Operating Systems Design)

Quick links: Course description, Course objectives, Text & Email , Labs, Exams, TA, Instructor, Course outline and lecture notes, policy.

Contents

Course Description

Most students enjoy this course because it exposes so much of the "magic" of how programs and resources are managed on modern computer systems.

More precisely the learning outcomes of this course enable students describe and analyze

- mechanisms and policies that enable the efficient illusion of concurrent execution of multiple programs upon one (or more) CPUs
- mechanisms and policies that enable a "virtual" address spaces that may be larger than allocated memory
- security mechanisms and policies that protect processes from unauthorized access to their memory or files
- data structures and algorithms used to organize persistant storage
- how resources such as cpu time, memory, and persistent storage are managed and allocated
- challenges in concurrency and approaches to avoiding race conditions and deadlock
- the role of device drivers in providing generic programming interfaces to i/o devices
- the intertwined role of interrupts, traps, paged and segmented memory translation, and supervisor mode in enabling multi-tasking, isolation, system calls, access to i/o devices, and timers
- coarse estimations of access time to persistent storage devices

These functions of a modern operating system are all based on the application of a few fundamental concepts. This course teaches these key concepts and "makes them real" by engaging students in the creation of small programs that implement their key functionalities.

The course includes frequent quizzes, a super-quiz (that serves as a midterm) and a final exam. All grading is explicitly mapped to course learning outcomes, which are generally assigned boolean scores indicating whether the students' work indicates an appropriate level of mastery. In order to appropriately assign credit for skill mastery in the context of clerical errors, "clerical accuracy" is also included as an additional skill that is graded on all test instruments. In order to enable the computation of a final grade largely dependent upon a comprehensive final to not be overly sensitive to errors caused by student stress, test and quiz grades are aggregated on a skill basis - which permits skills demonstrated in multiple (prior) quizzes to over-ride corresponding skills that may not be demonstrated on the final exam.

Operating systems design is an exciting area of computer systems that ties together architecture, language, and real-world constraints. Many solutions to the problem of managing a systems explored in this course involve interesting trade-offs between flexibility, performance, and reliability. In order that students gain a mature appreciation of these trade-offs, this course is conducted in a very interactive manner. Students are encouraged to dialog with each other and the professor to discuss the implications of design options. An email list facilitates student/instructor interaction outside of class.

Since the design of operating systems requires a very practical understanding of computer architecture, algorithms, and software design, most students are not prepared for this course before their senior year.

Few programming languages are as well suited to the implementation of operating systems as "C". For this reason, most labs for this course must be written in C. **Students with superficial understandings of the C language will struggle in this course and I strongly encourage all students attending CS4375 to carefully review "The C Programming Language" by Brian Kernigham and Dennis Ritchie.**

Course Objectives

Students enrolled in this course will:

- Learn the role of operating systems.
- Learn the theory underlying how operating systems are implementer and the implications of resulting design choices.
- Develop practical skills needed to understand and modify operating system code, feel competent to do so, and understand why it matters
- Acquire sufficient knowledge to be able to solve problems & know how to learn additional relevant info when needed.

Text & Email list

- **Text:**
  - Required
    - Operating Systems: Three Easy Pieces, by Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-

Dusseau.
    - Available free online at http://pages.cs.wisc.edu/~remzi/OSTEP/.
    - Hardcover ($36), softcover ($24), and electronic ($10) copies are also available from the same link.
  - Peter Dordal's Introduction to Computer Networks (free & web viewable)
- Recommended
  - Prior years: Tannenbaum's "Modern Operating Systems."
    - The 4th edition is current.
    - The 2nd or 3rd editions are very similar (and adequate for this course). In June 2015, Amazon was selling the 2nd edition at $0.88.
  - Kernigham & Ritchie, *The C Programming Language,* any edition.
  - Michael Kerrisk's The Linux Programming Interface (No Starch Press)
  - Nick Parlante's, *Pointers and Memory (pdf download)*
- Much communication is via email: **JOIN THE CLASS MAILING LIST:**
  - 2018 Fall OS google group: https://groups.google.com/forum/#!forum/f18-os
  - 2018 Fall OS teaching team: f18-os-teaching-team@googlegroups.com

Instructors:

- Dr. Eric Freudenthal
  - email: efreudenthal at utep.edu
  - phone: 915 747 6954
  - Office Hours: See my homepage
- Mr. Adrian Veliz
  - email: aeveliz@...
  - Office Hour: TR 10:30-11:20 @ TA Room

Teaching assistant

- TAs will be present for the first 10 minutes of their office hours and depart if nobody is present or expected.
- If you need to meet a TA at a particular time, please send them email.
  - If nobody is present or expected, they may leave.
  - If you need to meet a TA at another time, send them email.

TA Schedule	David Pruitt	Manali Arun Mahalungkar	Adrian Veliz	Edward Seymour
email	ddpruitt@miners	mamahalungka@miners.utep.edu	aeveliz@miners	esseymour@miners
Arch lab	tbd	tbd	tbd	tbd
Office hours	tbd	tbd	tbd	tbd

Course Content

- Lecture notes
- Coarse-grained topic outline
  - Introduction
    - What is an operating system?
    - History
    - Operating systems for various devices
    - Review of computer architecture
    - Summary of OS concepts
    - System Calls
    - Operating System Structure
  - Processes and Threads
    - Processes
    - Threads
    - Inter-process communication
    - Classical problems
    - Scheduling
  - Synchronization (mostly deadlock prevention)
    - Resources
    - Deadlock (another reason why ostriches do not dominate the planet)
    - Detection of deadlock
    - Stopping systems from entering deadlock (avoidance)
    - Designing systems that can not deadlock (prevention)
  - Memory Management
    - Single address space (both mono- and multiprogramming)
    - better nomenclature than book
      - virtual memory
      - demand v. voluntary systems
      - paged v. segmented systems
    - Managing paged systems
      - exposes a eviction (replacement) problem
      - program behavior (locality)
      - algorithms
      - implementation
      - simulation
    - Managing segmented systems
      - exposes both a placement & eviction problem
  - Input/Output
    - characteristics of i/o devices
    - how to structure drivers
    - disks
    - terminals, graphical devices
    - networked terminals
    - power mgmt
  - File Systems

        - files
        - directories
        - implementation
        - special files (/dev, links, /proc)
        - example filesystems
    - Multimedia
        - File encoding
        - processor scheduling issues
        - disk scheduling issues
        - storage, caching, and transmission issues
        - Multiprocessor Systems
        - Security

Exams, labs, and grading

- Course schedule
    - The course outline and schedule indicates the approximate sequence of assigned readings, labs, and homework exercises. Students are expected to
        - Skim readings prior to the class where they are discussed including attempting listed exercises.
        - Notify the instructor prior to or during class about topics and exercises they found confusing.
        - Review readings & homeworks after the class they are discussed.
- Exams/quizzes:
    - Quizzes
        - Quizzes assess individual students' abilities to demonstrate knowledge, to design solutions to realistic problems, and to present these solutions in a clear and professional fashion. Quizzes will be graded "by skill" (see below), can cover any concept or skill previously taught in the course, are generally offered at the rate of once per week, and unannounced (so students must be continuously prepared).
    - Final Exam
        - The final exam date is scheduled by the university based upon lecture time. Like quizzes, the final exam will be graded "by skill" (see below) and can cover any concept or skill previously taught in the course.
    - Rules
        - Solutions must be prepared individually without communication with or assistance from anybody except the instructor or proctor.
- Labs
    - Frequency & Deadlines
        - Labs will be assigned during the lab course
        - Due 1 week after assignment unless otherwise indicated.
    - Intention:
        - Labs are intended to provide an opportunity for students to practice and explore the application of concepts presented in class within programming assignments.
    - Students are expected to act professionally
        - By reading whatever resources they find relevant

        - By attributing credit to any person or reference materials that substantively contributed to their solutions
        - By only submitting solutions they fully understand.
        - Professionalism includes honesty, clarity, and accuracy.
        - Students are encouraged to help each other select and design problem-solving approaches, but they may not present solutions to each other for duplication or paraphrasing.
    - Requirements
        - **Functional:** Assignments will either require students to create complete programs or modify programs provided by the instructor.
        - **Documentation:** Submissions should include documentation that facilitates the grader's determination of
            - How to compile, use, and test
            - Principles of operation (e.g. comments & other descriptive prose)
            - Elements of the submission that were developed by others. This includes both algorithms and code. Vague attributions of credit (e.g. "Assistance was received from Jim Smith.") are unacceptable.
        - **Originality:** Any code not provided by the instructor must be "original work" by the student. As in written "essay" assignments,
            - Each instance of a specific algorithm, code fragment, comment, or explanation composed by or with another person must be individually documented.
            - The majority of "original work" must not be composed of verbatim or paraphrased copies of code or comments composed by somebody else.
        - **Completeness:** Students who labs that do not substantially satisfy functional and documentation requirements will receive no credit.
    - Homework:
        - Most class sessions will conclude with an assignment due at the beginning of the next class session (unless otherwise indicated).
        - While most assignments are neither collected nor graded, student mastery of relevant skills will be tested within quizzes and tests.
    - **More notes on grading policy**

- Submission policy
    - Labs will only be accepted on the published due date unless other arrangements are made.
    - All lab assignments must be completed to earn a passing grade.
    - All labs must be submitted using the version control system. Labs will not be accepted via email.

Accommodations for Students with Disabilities and Exceptional Circumstances

Individuals with disabilities have the right to equal access and opportunity. Please contact Dr. Freudenthal or the UTEP Office of Disabled Student Services (DSSO) if you have a special circumstance such that an accommodation would be

helpful in permitting you to excel or demonstrate mastery of the material covered in this course.

Standards of Conduct and Academic Honesty

- **Standards of Conduct:** Students are expected to conduct themselves in a professional and courteous manner, as prescribed by the Standards of Conduct: http://hoop.utep.edu/Student_Affairs_Chapter_One-HOP.htm Graded work should be unmistakably your own. You may not transcribe or copy a solution taken from another person, book, or other source (*e.g.,* a web page). Copying other's work will not be tolerated. Professors are required to report academic dishonesty and any other violation of the Standards of Conduct to the Dean of Students.
- **Permitted collaboration:** Students may discuss requirements, background information, test sets, solution strategies, and the output of their programs. However, implementations and documentation must be prepared individually. Students are strongly encouraged to document advice received from others and all other resources utilized in the preparation of their assignments.
- **If academic dishonesty is suspected:** You will receive an incomplete for the lab, and your case will be referred to the Dean of Students for adjudication. The Dean of Students has published a website with complete details concerning the UTEP Academic Honesty policy at the following arcane URL: http://studentaffairs.utep.edu /Default.aspx?tabid=4386.

Course Grading Policy

- **Overall course grading:** It's posted on this web page: Exams and Grading

Expectations of UG/Grad Students

Both graduate and undergraduate students will may attend this course. Graduate students are expected to demonstrate a higher level of technical competency, analytical maturity, and communication skills than undergraduates as demonstrated by (1) class participation, (2) exams and (3) lab assignments. Some laboratory assignments will have advanced sections that only graduate students will be required to submit. Finally, the official course outcomes specify a range of mastery levels of the topics covered in this course that must be demonstrated by students in order to earn high marks. In order to be assigned similarly high marks, MS candidates are expected to demonstrate higher levels of mastery.

Tools

Getting Started in Linux/UNIX and SVN

Linux under virtualization

This is another way to run linux - just pickup a virtualization system such as vmware, configure a virtual machine, and instal linux onto it.