COURSE SYLLABUS

****

YEAR COURSE OFFERED: 2015

SEMESTER COURSE OFFERED: Fall

DEPARTMENT: Computer Science

COURSE NUMBER: CS4375

NAME OF COURSE: Theory of Operating Systems

NAME OF INSTRUCTOR: Dr. Eric Freudenthal

****

The information contained in this class syllabus is subject to change without notice. Students are expected to be aware of any additional course policies presented by the instructor during the course.

****

Learning Objectives

On successful completion of this course, students will

1. be able to apply the following in new situations:
    a. operating system objectives and functions
    b. process definition/description and control/management
    c. threads, symmetric multiprocessing, microkernels
    d. mutual exclusion and synchronization (software and hardware approaches)—semaphores, monitors, message passing, readers/writers problem
    e. concurrency: deadlock and starvation—principles of deadlock, deadlock prevention, avoidance, and detection
    f. dining philosophers problem
    g. memory management—paging, segmentation
    h. virtual memory—hardware and control structures
    i. scheduling algorithms

2. be able to apply:
    a. file management (file organization, directories, and sharing), record blocking, secondary storage management
    b. multiprocessor and real-time scheduling
    c. I/O management and disk scheduling

3. have been introduced to:
    a. Current windows operating system
    b. UNIX operating system
    c. distributed processing, client/server, and clusters

d. distributed process management

Course Grading

- Course grade is an aggregation of:
  - Partial-term grades
  - Final exam grade
- Partial term grades
  - Consist of grades collected during and immediately following the portions of the course corresponding to major course themes
    - OS overview & processes
    - Coordination & deadlock
    - Memory Management
    - I/O & filesystems
  - The minimum of scores from
    - Graded instruments (tests, quizzes)
    - Graded assignments
    - Labs
- Graded instruments
  - Tests/quizzes
    - Final exam: date and time are specified by the university.
    - Frequent Quizzes (in lieu of tests)
      - Generally unannounced, at least one every two weeks, generally focus on recently studied topics, but may contain topics studied earlier
      - Short: Generally 10-20 minutes
      - May not be "made up"
    - Graded homeworks (assigned intermittently)
  - Multiple skills may be measured by the same problem within an instrument
  - To the extent that it is practical, **useful competency (generally a binary value)** for distinct skills, called a proficiency, will be assessed independently
    - Thus, if an instrument with four questions measures ten skills, ten measurements will be computed.
    - A single problem may provide opportunity to demonstrate all or most of the proficiencies being measured.
    - A proficiency not demonstrated by any answer provided by a student due to not providing complete and correct answers to all problems will be assessed as not meeting the threshold of **useful competency.**
    - It is possible that all skills measured by a particular instrument will be demonstrated in multiple problems.
    - **In-class Instruments are designed to require substantially less time than allotted**. Therefore not completing a quest within the allotted time may be an indication of weak understanding worthy of discussion with the instructor.
  - Notes on test-taking strategy

        - If short of time - it **is not generally advantageous** to partially answer multiple questions in a manner that repeatedly demonstrates the same skills.. .
    - The grade for an instrument will correspond to the **fraction f of skills in which useful competency is demonstrated.** Generally
        - **100% corresponds to an A+ (4.3 on a 4-pt scale)**
        - **50% corresponds to an F (zero on a 4-pt scale)**
        - Conversion back to 4-pt scale: *Grade = (f-0.5) * 8.6* where *f* is the fraction described above
    - As in life, all instruments are cumulative.
        - If test (midterm/quiz/final) T1 occurs before test T2, a skill measured in T1 may also be assessed in T2.
- Graded assignments (including labs)
  All labs involve low-level programming in the C programming language and must be developed using the "official" linux command-line tools (including gcc, make, svn, and emacs) using the "arch1 virtual machine."
    - Intention:
        - Assignments and labs provide an opportunity for students to practice and explore concepts presented in class.
        - Students are expected to act professionally
            - By helping each other select and design problem-solving approaches
            - 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.
    - Submission and due dates
        - Submission is via SVN commit. Solutions will not be accepted via email.
        - Due dates are posted on course web site
        - Graded twice (averaged)
            - As submitted at due date
            - As updated one week after grade is distributed, **only upon email request from the student.**
    - Rules
        - Students must only submit solutions that fairly reflect their own understandings.
        - Solutions must clearly and fairly attribute credit people and resources that contributed to their design or preparation.
        - Descriptive text included with solutions must be composed by the student submitting it.
    - Implication
        - It is academic dishonesty for a student to submit a solution they cannot replicate individually or to not fairly credit their sources.

Required Reading

COURSE SYLLABUS

- **Text:** Tannenbaum's "Modern Operating Systems", 2nd , 3rd , or 4th Edition.

Recommended Reading

- Kerningham, Brian W & Ritchie, Dennis M. "The C Programming Language, Second edition," Prentice Hall, ISBN: 0-13-115817-1.
- Recommended by students (and by no way required): Android app "Programmer Mental Math" by Joel Jurix.

Accommodations for people with disabilities

If you have a disability and need classroom accommodations, please contact the *Center for Accommodations and Support Services* (CASS) at 747-5148, or by email to cass@utep.edu, or visit their office located in UTEP Union East, Room 106. For additional information, please visit the CASS website at www.sa.utep.edu/cass.

List of discussion/lecture topics

Note: detailed lecture notes are available on the course web site.

- 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 race condition and 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 cannot deadlock (prevention)
- Memory Management
  - Fragmentation: internal and external

        - Scope: User (e.g. malloc) v. system (sbrk or mmap) allocation
        - Algorithms for user-program allocation
        - OS memory management
            - Segmented, paged, relevance of locality
            - Translation mechanisms and data structures
            - Resident or demand-loaded
            - Placement, replacement (eviction) strategies
- Input/Output
    - characteristics of i/o devices including keyboards, mice, graphics adapters, storage, network interfaces
    - layered software architecture used for device drivers and network stacks including event management (both “hard” and “soft” interrupts)
    - Security challenges related to i/o (e.g. buffer overruns)
- File Systems
    - files
    - directories
    - implementation
    - special files (/dev, links, /proc)
    - example filesystems
    - security policies
        - implementation of discretionary and mandatory policies
        - TOCTTOU vulnerabilities
- Virtualization
    - Concept, Motivations
    - Type I and II
    - Compare with emulation/translation (e.g. JVM, CLR)