CS A253: PRINCIPLES IN SYSTEM DESIGN

Item	Value
Curriculum Committee Approval Date	12/02/2020
Top Code	070600 - Computer Science (Transfer)
Units	4 Total Units
Hours	72 Total Hours (Lecture Hours 72)
Total Outside of Class Hours	0
Course Credit Status	Credit: Degree Applicable (D)
Material Fee	No
Basic Skills	Not Basic Skills (N)
Repeatable	No
Open Entry/Open Exit	No
Grading Policy	Standard Letter (S)

Course Description

This lab is required by four-year institutions, coupled with CS A253, to satisfy lower-division work that prepares students for upper-division work in Computer Science. PREREQUISITE: CS A150 and CS A170. Transfer Credit: CSU; UC.

Course Level Student Learning Outcome(s)

1. Solve problems using the C programming language.
2. Describe the operation of a basic dynamic memory allocator and virtual memory.
3. Use low-level operating system APIs such as POSIX, to interface between user programs and the operating system.

Course Objectives

- 1. Describe the operation of a basic dynamic memory allocator and virtual memory.
- 2. Program using POSIX API functions in the C programming language.
- 3. Navigate in the *nix command-line environment.
- 4. Understand the fundamentals of program execution within an operating system.
- 5. Program in a "low-level" language, such as C.
- 6. Understand the client-server networking model.
- 7. Understand the complexities in concurrent programming.

Lecture Content

Unix/Linux command-line environment Navigating in the file system Directories Current working directory and root Pathnames Absolute/Relative Changing Directories Listing the contents of a directory Files and Directories Creating Files Creating Directories Copying Files and Directories Redirection Redirecting Standard Input, Output and Error Pipes Filters Permissions Read, Write and Execute (chmod) Processes What is a process Viewing current processes Pausing/Interrupting a process Using the vim command-line text editor C Programming Language Variables and Data types Operators and Expressions The main function Command line arguments Compilation from the command-line gcc compiler make utility Control Statements if statement while loop for loop switch statement Data Structures Arrays User defined Data Structures Memory layout Pointers Dynamic Memory Management Allocation using malloc Deallocation using free How dynamic memory is managed through the Operating System Functions Function signature Function pointers The Stack and local variables Linking and Loading Object file structure How the operating system uses and executable file to load a program Debugging using gdb System Stack Interrupts Exceptions Signals POSIX Abstractions Use low-level system APIs to interface with the Operating System File system I/O functions: open, close, read and write Process creation fork exec Memory Management mmap and munmap Memory Hierarchy Cache Direct mapped cache Set associative cache Fully associative cache Virtual Memory Pages and the Page Table Page faults Translation-lookaside buffer (TLB) Concurrent Programming Threads (Pthreads Library) Shared resources Race conditions Mutexes Atomicity Semaphores Network Programming Sockets IP addresses IPv4 and IPv6 Ports Client-Server Model Server Client Application Layer/Communication protocol Writing a simple Server/Client application

Method(s) of Instruction

- Lecture (02)
- DE Live Online Lecture (02S)
- DE Online Lecture (02X)

Instructional Techniques

Lecture, discussion, demonstration, live coding, in-class exercises.

Reading Assignments

Students will spend a minimum of 4 hours per week reading the textbook and/or other reading material assigned. Students will be expected to follow along with the exercises in the reading material

Writing Assignments

Students will spend a minimum of 6 hours per week writting code.

Out-of-class Assignments

Students will spend a minimun of 6 hours per week completing weekly programming assignments.

Demonstration of Critical Thinking

Written examinations and laboratory exercises.

Required Writing, Problem Solving, Skills Demonstration

Successful performance of the laboratory assignments.

Eligible Disciplines

Computer science: Master's degree in computer science or computer engineering OR bachelor's degree in either of the above AND master's degree in mathematics, cybernetics, business administration, accounting or engineering OR bachelor's degree in engineering AND master's degree in cybernetics, engineering mathematics, or business administration OR bachelor's degree in mathematics AND master's degree in cybernetics, engineering mathematics, or business administration OR bachelor's degree in any of the above AND a master's degree in information science, computer information systems, or information systems OR the equivalent. Note: Courses in the use of computer programs for application to a particular discipline may be classified, for the minimum qualification purposes, under the discipline of the application. Master's degree required.

Textbooks Resources

1. Required Bryant, R.E., O'Hallaron, D.R.. Computer Systems: A Programmer's Perspective, 3rd ed. Santa Monica, CA: Pearson, 2016