The University of Georgia

Department of Computer Science

Course Information Sheet

CSCI 1730

Systems Programming

Brief Course Description
(50-words or less)

Programs and programming techniques used in systems programming. Assembler, linker, loader, pipes, sockets, and system analysis methods used in systems environment.

Extended Course Description / Comments

This course covers the basics of UNIX systems programming, including file and directory structures, basic and advanced file i/o, process creation, and interprocess communication. An initial unit on "C++ for Java rogrammers" will familiarize students with the use of C/C++ in systems programming.

Pre-Requisites and/or Co-Requisites

Prerequisite: CSCI 1301: Introduction to Computing and Programming
Co-requisite: CSCI 1302: Software Development

Required, Elective or Selected Elective

Required Course

Approved Textbooks
(if more than one listed, the textbook used is up to the instructor's discretion)

Author(s): Deitel and Deitel
C++*: How to Program*
Publisher: Prentice Hall
Edition: Eighth
ISBN-13: 9780132662369

Author(s): Adam Hoover Title:
Title: *System Programming with C and UNIX*
Publisher: Addison Wesley
Edition: First
ISBN-13: 978-0136067122

Specific Learning Outcomes (Performance Indicators)

1. Design and implement a C++ project of moderate size, consisting of a main driver class and multiple class files and employing composition, inheritance and polymorphism.
2. Design and implement programs that use both static objects and dynamic memory management and demonstrate knowledge of state and behavior by constructing memory maps and predicting program output.
3. Demonstrate knowledge of the differences between pass-by-value and pass-by-reference by predicting program output.
4. Demonstrate knowledge of variable scope rules by predicting output.
5. Design and implement programs that make appropriate use of pointers, references, function pointers, operator overloading, and exception handling.
6. Construct memory maps of the state of the stack, heap, and global and static memory during the execution of a C++ program.
7. Use the "make" utility, a software engineering tool for managing and maintaining computer programs.
8. Use "gdb" to debug programs with a variety of errors.
9. Use the UNIX command line interface to create, delete, move, copy and copy files and directories.
10. Use the UNIX command line interface to spawn processes that redirect input or output or communicate through a pipe.
11. Design and implement C programs that employ the UNIX file access primitives (open, close, read, write, lseek, fcntl).
12. Demonstrate knowledge of UNIX kernel and process data

structures by sketching file descriptor table, file table, and inode table structures and the updates that correspond with the execution of sample code.
13. Design and implement C programs that employ UNIX process system calls and signal handling (fork, exec, wait, join, etc.)
14. Design and implement C programs that implement a client and server that communicate via the UNIX socket system call interface.

Relationship Between Student Outcomes and Learning Outcomes

		Student Outcomes										
		a	b	c	d	e	f	g	h	i	j	k
Learning Outcomes		●	●	●						●	●	●
		●	●	●						●	●	
		●	●	●						●	●	
		●	●	●						●	●	●
		●	●							●	●	
		●	●	●						●	●	●
	7									●		
	8									●		
	9									●		
	10									●		●
	11	●	●	●						●	●	●
	12	●	●	●						●	●	●
	13	●	●	●						●	●	●
	14	●	●	●						●	●	●

Major Topics Covered
(Approximate Course Hours)

3 credit hours = 37.5 contact hours
4 credit hours = 50 contact hours

Note: Exams count as a major topic covered

C++ development environment, style guidelines, Makefiles (3.5 h)
How C+ differs from Java (1 h)
UML class diagrams and modeling (1 h)
Unix command line (1.5 h)
Editors (vi and emacs) (1 h)
C++ Classes (3 h) & Control Structures (1.5 h)
Scope, storage class, parameter passing (1.5 h)
Pass-by-value, Pass-by-reference (1.5 h)
Debugging with gdb (1.0 h)
Function templates, arrays, vectors (1.5 h)
Pointers, array names, function pointers (3 h)
Pointer and array notation (2 h)
Constructors, destructors, member-wise assignment (2 h)
Composition and Inheritance (3 h) & Operator Overloading (2 h)
Inheritance and Polymorphism (4 h)
Unix system architecture (1 h) & Files and Directories (2.5 h)
Processes (fork, exec, etc.) (3 h)
Signals (2 h) & Concurrency (3 h) &Programming with Pthreads (2 h)
Sockets (1.5 h)

Course Master

Dr. Michael Cotterell