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

Use this section to put additional information that's relevant to whom this course is targeting

Pre-Requisites and/or Co-Requisites

CSCI 1301 (Pre-requisite): Introduction to Computing and Programming
CSCI 1302 (Co-requisite): Software Development in Java

Pre-Requisites and/or Co-Requisites

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

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

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

This four-hour 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 programmers" will familiarize students with the use of C and C++ in systems programming. At the end of the semester, all students will be able to do the following:

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 program 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 1	•	•	•						•	•	•
2	•	•	•						•	•	
3	•	•	•						•	•	
4	•	•	•						•	•	•
5	•	•							•	•	
6	•	•	•						•	•	•
7									•		
8									•		
9									•		
10									•		•
11	•	•	•						•	•	•
12	•	•	•						•	•	•
13	•	•	•						•	•	•
14	•	•	•						•	•	•

Student Outcomes

a. An ability to apply knowledge of computing and mathematics appropriate to the discipline.
b. An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution.
c. An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs.
d. An ability to function effectively on teams to accomplish a common goal.
e. An understanding of professional, ethical, legal, security and social issues and responsibilities.
f. An ability to communicate effectively with a range of audiences.
g. An ability to analyze the local and global impact of computing on individuals, organizations, and society.
h. Recognition of the need for and an ability to engage in continuing professional development.
i. An ability to use current techniques, skills, and tools necessary for computing practice.
j. An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices.
k. An ability to apply design and development principles in the construction of software systems of varying complexity.

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 hours) 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) Prgramming with Pthreads (2 h) Signals (2 h) Sockets (1.5 h)
Assessment Plan for this Course	Each time this course is offered, the class is initially informed of the Course Outcomes listed in this document, and they are included in the syllabus. At the end of the semester, an anonymous survey is administered to the class and each student is asked to rate how well the outcome was achieved. The choices provided use a 5-point Likert scale containing the following options: Strongly agree, Agree, Neither agree or disagree, disagree, and strongly disagree. The results of the anonymous survey are tabulated and results returned to the instructor of the course. The course instructor takes the results of the survey, combined with sample student responses to homework and final exam questions corresponding to course outcomes, and reports these results to the ABET committee. If necessary, the instructor also writes a recommendation to the ABET committee for better achieving the course outcomes the next time the course is offered.
How Data is Used to Assess Program Outcomes	Each course Learning Outcome, listed above, directly supports one or more of the Student Outcomes, as is listed in "Relationships between Learning Outcomes and Student Outcomes". For CSCI 1730, Student Outcomes (a)(b)(c)(i)(j) and (k) are supported.
Course Master	Dr. Eileen Kraemer
Course History	