CS202 Systems Programming Concepts

April 5, 2009

1 General information

Instructor	Changhua Wu
Office	AB 2-100M
Office Hours	Monday & Thursday 3pm-5pm
Phone	762-9500 x 9706
Email	cwu@kettering.edu
Text	Molay, Understanding Unix/Linux Programming
Reference	Harbison and Steele, C: A Reference Manual, Fifth Edition K. N. King, C Programming: A Modern Approach
Prerequisites	CS-102

2 Catalog Description

Fundamental system programming concepts are examined using the C programming language. Topics include: C programing language, Unix variants and standardizatoin, data representation, interrupt handling, I/O, file management, dynamic structures, parameter passing, memory management, system calls, process creation, process control, interprocess communication, and language interfaces.

3 Topics

- C language programming including problem solving, program design, implementation, and testing.
- C programming language constructs including variables, constants, literals, and comments.
- Programming style, white space for clarity of program.
- Basic data types and structs
- Pointers and reference handling.
- Program statements including assignment, invocation, control flow via decision, case, and loop.
- Operating system standardization and portability.

- Low level and high level file I/O including buffering.
- Terminal I/O.
- Atomic and non-atomic operations.
- File topics including file types, directories, permissions, ownership, access, and file systems.
- System data files.
- Unix process environment including memory layout and memory allocation.
- Process control including process creation, process termination, process execution, and interprocess communication.
- Signals, pipes, coprocesses, and FIFOs.

4 Objectives

Each student who receives credit for CS-202 will have demonstrated the ability to do all of the following tasks.

- Design, implement, compile, test, and run a C computer program which uses system calls.
- Effective use of Unix man pages.
- Write a program using appropriate documentation and style for effective communica- tion with a human reader.
- Write a systems program which is portable to another POSIX-compliant operating system.
- Create, terminate, and execute processes via calls to system routines.
- Implement simple communication between processes.
- Write code to handle system interrupts.

5 Grading

Midterm Exam	20%
Final Exam	20%
Programming Projects	60%

In order to pass this course, your total weighted points must be at least 40% of the total points. Furthermore, a necessary condition to receiving a grade of 78 or higher for the course is a programming score of at least 50% of the total programming points.

6 Policies

- There is no make-up examination for a midterm.
- No late assignments will be accepted. Assignments to be handed in are due at the time and date listed on the assignment.
- Attendance is strongly encouraged, but is not required. Lack of attendance per se will not adversely affect your grade. However, you are responsible for all information, an- nouncements, etc., given in class, whether or not you are in attendance. The instructor will put lecture note online before the class. You should print it before coming to the class.
- Unless otherwise stated, you are expected to do your own work. You are not allowed to work in teams; any outside references you use must be cited. All suspected cases of academic dishonesty will be handled in strict accordance with department and institute policy.
- This syllabus provides a general plan for the course; deviations may be necessary.