CSE 303: Concepts and Tools for Software Development
Course Information and Syllabus
Spring 2007

Logistics and Contact Information: The instructor is Dan Grossman. See the course homepage (`www.cs.washington.edu/education/courses/cse303/07sp`) for information about teaching assistants, office hours, etc. *You must join the class email list and check email at least once every 24 hours.*

Goals: Successful course participants will:

- Develop the skills to automate common computing tasks such as file-manipulation and string-processing

- Internalize C-level programming and obtain beginning proficiency in C programming

- Appreciate programming tools such as debuggers, profilers, compilation managers, and version control

- Learn valuable software-engineering practices regarding specification and testing

- Understand the basic issues and pitfalls of shared-memory concurrency

- Develop the intellectual maturity to evaluate the societal/ethical implications of computing

Grading and Exams:

Midterm	20%	Monday, April 30, in class
Final	20%	Tuesday, June 5, 2:30-4:20
Homeworks	50%	approximately weekly (probably 7 total)
Short "Issue" Paper	10%	to be described later

Unless announced otherwise, all homeworks contribute equally to the 50%.

Late Policy: Homework will always be due at 9:00AM on the due date. This deadline is strict. Therefore, it is exceedingly unlikely that skipping class or being late to class because of homework is in your interest. For the entire quarter, you may have three "late days". You are strongly advised to save them for emergencies. You may not use more than two for the same assignment. They must be used in 24-hour chunks.

Academic Integrity: Any attempt to misrepresent the work you did will be dealt with via the appropriate University mechanisms, and your instructor will make every attempt to ensure the harshest allowable penalty. The guidelines for this course and more information about academic integrity are in a separate document. *You are responsible for knowing the information in that document.*

Text: There are three books listed for the course. *The course webpage discusses the texts' requiredness.*

- Linux Pocket Guide by Daniel J. Barrett, O'Reilly, 2004.

- Programming in C (3rd Ed.) by Stephen G. Kochan, Sams Publishing, 2005.

- C: A Reference Manual (5th Ed.) by Samuel P. Harbison, Guy L. Steele. Prentice Hall, 2002.

Advice: This course will expose you to a tremendous number of tools, concepts, and issues. Be warned:

- The unease from using new tools may drain your energy because you will constantly learn new tools.

- The lectures will *not* teach the tools with enough detail to do the homework. Rather, they will give you the concepts behind the tools and enough to point you in the right direction.

- You will not master everything in 10 weeks, but you will learn enough to continue increasing proficiency and more easily learn computer science.

- If you are spending an enormous amount of time on a homework, you are likely missing a key concept. Spending more time "fighting through it" is not effective; use yourself and the course staff to determine what you are missing.

Approximate Topics (subject to change): We will do almost all of the follwing, but not in this order.

1. Societal/ethical implications of computing (4 classes)

   (a) *discussions* on topics to-be-announced

   (b) example topics: software patents, digital privacy, digital rights management, software licensing, software-engineer certification, the digital divide, accessibility, software security, electronic voting

2. Files, processes, and shells (6 classes)

   (a) Command-line utilities

   (b) File editing

   (c) Shell scripting

   (d) String processing; regular expressions

   (e) Web basics (http, html)

   Note: For consistency, we will all use Linux and bash. The concepts are similar for other operating systems and shells.

3. C Programming (6 classes)

   (a) Memory model

   (b) Pointers

   (c) Arrays

   (d) Manual resource management

   (e) The preprocessor

   (f) Idioms for safe programming

4. Programming tools (6 classes)

   (a) Debuggers

   (b) Profilers

   (c) Linkers

   (d) Compilation managers

   (e) Version-control systems

5. Concurrency (2–3 classes)

   (a) Threads

   (b) Races and deadlocks

   (c) Locks and Transactions

   (d) Message-passing

6. C++ (2 classes)

   (a) C with objects

   (b) Templates

   (c) Other differences from C

7. Software-engineering issues (2 classes)

   (a) Multiperson programming

   (b) Specification

   (c) Testing

   (d) Code-reuse patterns