CSE 303: Concepts and Tools for Software Development
Course Information and Syllabus
Winter 2009

Logistics and Contact Information: Instructor: Hal Perkins, CSE 548, perkins(at)cs. See `www.cs.washington.edu/303/` for information about teaching assistants, office hours, etc.

Communications A discussion list is linked to the course home page so we can keep in touch outside of class meetings. Please participate. You will also automatically be subscribed to a class mailing list at your uwnetid address. This will primarily be used for announcements, clarifications, hints, and other notices from the course staff, and you are responsible for everything that is mailed there.

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%	Wednesday, February 11, in class (tentative)
Final	25%	Monday, March 16, 8:30-10:20
Homeworks	45%	approximately weekly (6 or 7 total)
Short "Issue" Paper	10%	to be described later

In general, all homeworks contribute equally to the 45%, but larger programming projects may be weighted slightly more that other assignments. Percentages are tentative and may be adjusted slightly.

Late Policy: Deadlines will be given in each assignment. These deadlines are 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 four "late days". You are strongly advised to save them for emergencies. You may not use more than two for the same assignment, and on group projects you may only use late days if all members of the group have them available, and all members of the group will be charged for each late day used. They must be used in 24-hour (integer) chunks. *This policy may not be the same as in other classes. You are responsible for understanding it if you choose to submit late work.*

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 two books listed for the course. *The course webpage discusses the texts' "requiredness."*

- Linux Pocket Guide by Daniel Barrett, O'Reilly, 2004.
- C: A Reference Manual (5th ed.) by Samuel Harbison and Guy 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 your proficiency and more easily learn what you need in the future.
- If you are spending an enormous amount of time on an assignment, 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 following, 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, gender and diversity, 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