# CSE 303: Concepts and Tools for Software Development
# Course Information and Syllabus
## Winter 2006

**Logistics and Contact Information:** The instructor is Dan Grossman. See the course homepage (`www.cs.washington.edu/education/courses/cse303/06wi`) 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 necessary to automate common computing tasks such as file-manipulation and string-processing
- Internalize C-level programming and obtain beginning proficiency in C programming
- Appreciate common programming tools including debuggers, profilers, linkers, recompilation managers, and version-control systems
- Practice valuable software-engineering practices regarding specification and testing
- Critically evaluate several societal/ethical implications of computing and develop the intellectual maturity to evaluate new issues as they arise

**Grading and Exams:**

| | | |
|---|---|---|
| Midterm | 20% | February 8, in class |
| Final | 20% | March 16, 8:30–10:20 |
| 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 two books listed for the course.

- Linux Pocket Guide by Daniel J. Barrett, O'Reilly, 2004.
- C: A Reference Manual (5th Edition) by Samuel P. Harbison, Guy L. Steele. Prentice Hall, 2002.

See the course webpage for a discussion of the texts' requiredness.

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

- The unease felt when using new tools will be a constant energy drain 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 find yourself spending an enormous amount of time on a homework, it's likely because you are missing a key concept. If so, spending more time "fighting through it" is not effective; use yourself and course staff to determine what you're missing.

**Approximate Topic Schedule:** This schedule is *subject to change* and probably has more than we have time for. We will do most of it.

1. Societal/ethical implications of computing
   (a) 4–5 *discussions* on topics to-be-announced
   (b) *interspersed in the course*
   (c) 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 (2.5 weeks)
   (a) Command-line utilities
   (b) File editing
   (c) Shell scripting
   (d) String processing; regular expressions
   (e) Web scripting (cgi, http, html)

   *Note: For the sake of consistency, we will all use Linux and bash. The concepts are similar for other operating systems and shells.*
3. C Programming (2.5 weeks)
   (a) Memory model
   (b) Pointers
   (c) Arrays
   (d) Manual resource management
   (e) Idioms for safe programming
   (f) Hint of C++
4. Programming tools (2 weeks)
   (a) Debuggers
   (b) Profilers
   (c) Linkers
   (d) Libraries
   (e) Compilation managers
   (f) Version-control systems
5. Software-engineering issues (1.5 weeks)
   (a) Multiperson programming
   (b) Specification
   (c) Testing
   (d) Code-reuse patterns
6. Threads (.3 weeks)