CS212

Hard Skills

Spring 2006

Announcements

- BOOM:
  - Wed March 8, 4-6, Duffield
  - http://www.cis.cornell.edu/boom/2006sp/
- P2a meetings! Mandatory!
- P2b due after Prelim 1

Overview

- Software tools
- Programming
- Software engineering

Software Tool: Unix

- Unix:
  - what is it?
  - why did it start?
  - connection to programming?
- Philosophy:
  - everything is a ____ ?
  - little or big tools?
  - connecting tools?
  - shells and customization?

Obtain Unix

- Cygwin: www.cygwin.com
- Linux: www.linux.org
- FreeBSD: www.freebsd.org
- Cornell:
  - CSUG: www.csuglab.cornell.edu
  - ACCEL: www.accel.cornell.edu

Programming Languages

- Fortran
- C/C++/C#...
- Philosophy:
  - choose language based on...?
  - can all languages do everything?
- Unix for programming:
  - **`cc`**, **`CC`**, **`f95`**, ...
  - compilers: **`yacc`**, **`lex`**

Scripting Languages

- ***Script***: sequence of common commands made into a single program
  - Unix uses shell scripts
  - The shell is the interactive interface to Unix
  - You can combine commands from the Unix shell to create programs
- A scripting language is
  - Usually easy to learn
  - Interpreted instead of compiled
- Examples:
  - Perl, Python, **`sh`**, **`csh`**, **`bash`**, ...

Example

```
class Stack (object):
  def __init__ (self):
     self.stack = [ ]
  def put (self, item):
     self.stack.append(item)
  def get (self):
     return self.stack.pop()
  def isEmpty (self):
     return len(self.stack) == 0
```

Makefiles

- Used when compiling/recompiling a large system (several interdependent files)
  - Checks which files have changed and only recompiles those that are necessary
  - Because of dependencies, more than just the changed files can need to be recompiled
  - Of course, can always recompile everything, but this can be too expensive
- Once you have a makefile
  - You recompile whatever is necessary with **`make`**
  - see **`man make`** on Unix
- How to write?
  - Find something to copy

Makefile Example

```
/* Inside file called main.c: */
int print1(void);
#include <stdio.h>
int main() {
if(!print1())
printf("Error");
return 0;
}

/* Inside file called sub1.c: */
#include <stdio.h>
int print1(void) {
printf("Hello, world!\n");
return 1;
}
```

Inside file called **`Makefile`**:

```
main:   main.o sub1.o
 cc main.o sub1.o -o main
main.o: main.c
sub1.o: sub1.c
```

At UNIX prompt:

```
% make
```

Output from UNIX:

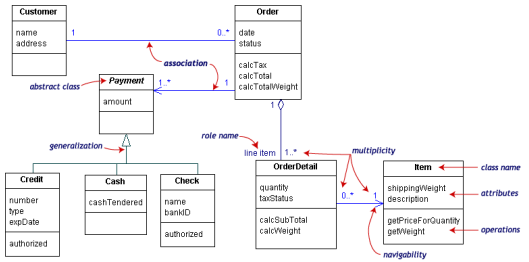
```
cc   -c  main.c
cc   -c  sub1.c
cc main.o sub1.o -o main

% main
Hello, world!
```


Version Control

- Allows you to keep track of changes for a large project
  - Can back up to old version if changes create problems
  - Multiple contributors can work on the system
- ***CVS*** (Concurrent Version System)
  - Open source
  - Widely used tool for version control
  - Maintains a history of all changes made
  - Supports branching, allowing several lines of development
  - Provides mechanisms for merging branches back together when desired

UML

- ***Unified Modeling Language***
  - Design tool for object oriented programming
  - System for showing the interaction of objects



Some Software Engineering

- ***Engineering***
  ABET: "the profession in which a knowledge of the mathematical and natural sciences gained by study, experience, and practice is applied with judgment to develop ways to utilize, economically, the materials and forces of nature for the benefit of mankind."
- ***Software Engineering***
  IEEE: "The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software."
- Is Software Engineering a type of Engineering?

13

General Development Principles

- Decomposition
  - "divide and conquer"
  - stepwise refinement
- Abstraction
  - work on details and then hide them
  - kinds: procedural, data, type, others... ("Generic Programming" in CS211)
- Specification
  - think interfaces
- Documentation

14

Validation

- Validation
  - how do you know your program actually works?
- Formal techniques (in Engineering, the theory!)
  - program correctness and proofs
  - need more CS
- Testing (in Engineering, the experiments!)
  - ***glassbox***:
    examine implementation and attempt to test all possible "paths" through the program
  - ***blackbox***:
    test cases are generated based on the specification without regard to implementation

15

Programming "in the large"

- Models for software development
  - waterfall and others
- Requirements analysis
  - functional, performance requirements
  - delivery schedule
- Data models
  - kinds and relations of data
  - UML very useful
- Program Design
  - top-down
  - bottom-up
  - stepwise refinement
  - coupling/cohesion
- Design patterns

16

Coding Quality

- Pareto's Law:
  - 80/20 rule
  - variant: 80% of everything is...
- Software?
  - ___% of defects caused by ___% of code
- NSA study [Drake, IEEE Computer, 1996] on 25 million lines of code
  - 70-80% of problems were due to 10-15% of modules
  - 90% of all defects were in modules containing 13% of the code
  - 95% of serious defects were from just 2.5% of the code

Profiling

- The goal is to make a program run faster
  - Pareto: 80% of the time is spent in 20% of the code
  - No use improving the code that isn't executed often
  - How do you determine where your program is spending its time?
- People are notoriously bad at predicting the most computationally expensive parts of a program
- Example: part of data produced by a profiler (Python)

```
2649853 function calls (2319029 primitive calls) in 53.502 CPU seconds
 Ordered by: standard name
 ncalls  tottime  percall  cumtime  percall filename:lineno(function)
   2521    0.227    0.000    1.734    0.001 Drawing.py:102(update)
   7333    0.355    0.000    0.983    0.000 Drawing.py:244(transform)
   4347    0.324    0.000    4.176    0.001 Drawing.py:64(draw)
   3649    0.212    0.000    1.570    0.000 Geometry.py:106(angles)
     56    0.001    0.000    0.001    0.000 Geometry.py:16(__init__)
 343160    9.818    0.000   12.759    0.000 Geometry.py:162(_determinant)
   8579    0.816    0.000   13.928    0.002 Geometry.py:171(cross)
   4279    0.132    0.000    0.447    0.000 Geometry.py:184(transpose)
```


Political/Legal Issues

- Intectual Property (IP) and your job
- Copyright and plagiarism
- Open source and "free" software (and peer review)

Some Programming Quotes

- Weeks of programming can save you hours of planning.
- Bad code isn't bad, its just misunderstood.
- Debugging is anticipated with distaste, performed with reluctance, and bragged about forever.
- If I had eight hours to chop down a tree, I would spend 6 hours sharpening an axe.
- Real programmers don't comment their code. If it was hard to write, it should be hard to understand.
- ...and then it occurred to me that a computer is a stupid machine with the ability to do incredibly smart things, while computer programmers are smart people with the ability to do incredibly stupid things. They are, in short, a perfect match.
- Computer Science is no more about computers than astronomy is about telescopes.