CSCI 265 - Course Introduction

Motivation

The problem

- Inefficient programming habits
- Poor understanding of the computing environment

The CSCI 161 environment

- Small programs: a few hundred lines at most
- One or a few files
- Solo programming:
  single author, no users, no maintenance, reliability not critical

The typical industrial environment

- Large programs: thousands of lines each
- Many files: hundreds or thousands
- Team programming:
  - Many authors, many users, maintained for years
  - Multiple releases on multiple platforms
  - Reliability critical; sometimes lives depend on it
- Some of these problems occur in 3rd/4th year courses as well as in industry

Course theme

> "Work smarter not harder"
> or
> "How to spend your evenings in the pub not the lab"

Four ways to work smarter not harder

This is a sketch of what you will see in the next 3 months. Try and get the basic idea; don't try to understand the details today.

1. Software tools

Problem

A 1,000 line program that takes an hour to run

Thoughtless solution

Study the code line-by-line looking for inefficient code
Remove every such occurrence

Smart solution

Run a profiler to find out which statements consume most of the time
Modify only these statements

Discussion

Usually 5% of the code consumes 95% of the time, so most of the work in the thoughtless solution is wasted

2. Module decomposition

Problem

A 5,000 line C program where everything seems to be connected to everything else
Only the original developer can work on it
Every bug fix generates new bugs

Thoughtless solution

Convert the program to C++
Allocate extra hours for maintenance

Smart solution

Break the program down into manageable pieces: "modules"
Document the interconnections between the modules

Discussion

The thoughtless solution is expensive and may make things worse
By itself, a change in programming language rarely helps

3. Systematic verification

Problem

The "killer app" with reliability problems
Bugs are rampant even during demos
Problems are especially bad after every new release

Thoughtless solution

Add new programmers to the development staff and fix the bugs

Smart solution

- Software inspection: systematic peer review
  Standard practice in industry
  Very effective in reducing errors
- Automated testing: programs written to test other programs
  Once written, the test programs run automatically

Discussion

The thoughtless solution is expensive and may actually make the problem worse
It does not solve the problem: lack of systematic verification

4. Code libraries

Problem

You need a program to read two dates and times and determine the difference between them

Thoughtless solution

Devise a way to represent dates and times
Design an algorithm for date arithmetic
Take into account leap year, time zones, daylight savings time, etc.

Smart solution

Use the date and time functions from the standard C library

Discussion

The thoughtless solution costs more for less
The result will be less reliable and it must be maintained as well.