Guest lecture CS3300- SE Software Design – Pro Tips

Version 1.1.0 - BETA

So You Think You Can Code

- Programming is supposed to make your life easier. Make sure you're doing things that actually make it easier for you, and for others.

The Basics
1. Learn An Editor

- Coding in Notepad is hard
- Really, learn an IDE
- Emacs vs. vi(m)
- Code vs. Notebook?

The Basics
2. Use Version Control

- Don’t just overwrite your old stuff with new stuff—you might want it back
- Don’t save a bazillion copies of the old stuff—you might not want it back

myfile.c
myfile2.c
myfile.c.bak
myfile_20190709.c
myfile_old.c
myfile_yougettheidea.c.bak

The Basics
3. Learn shell scripting

- Test your code: `for x in 1..100 do; foo $x; done`
- Pretty up your prompt
- Hack your history
- Simplify updates

The Basics
4. Organize Your Work

- If your project/office/company has a system, use that. If not, create one.
- It doesn't matter how (probably)
- Everything in one directory is a bad idea
- Everything in your home directory is a **REALLY** bad idea

How Do We Write Software?

1. A little at a time—use modular code
2. Take time to design first
   - What does this do?
   - How does it do it?
   - What does it rely on?
3. **Code Review & Test**

Code Review
The Big Idea

1. It’s harder to read code than write it.
2. Most code you write will be read by someone else.
3. You’re (probably) not that clever.

Code Review
Some sample code

What does this code do?

```
char *p;
p = malloc(MAX_LEN);
while(*p++ = *q++);
```

Code Review
Questions you might ask

- What does this program do?
- Was that easy to figure out? Does the name of the file help? The names of the functions? The variables? The comments?
- There are comments, right?
- Can you figure out how to run it?
- Does it work?
- Does it solve the problem it's supposed to solve?
- Can it solve similar problems without major re-writes?
- Are there any obvious issues you can see with the code?

How To Write Easy-to-read Code
1. Write Idiomatic Code

- myints is an array (or maybe a list) of 100 integers
- write code to run a function (foo) on each member of myints, and store those results

```
int newarray[100];
int i;
for(i=0; i<100; i++) {
    newarray[i] = foo(myints[i]);
}
```

How To Write Easy-to-read Code
2. Don’t Repeat Yourself

- AKA write modular code
- Set yourself up to solve similar problems

How To Write Easy-to-read Code
3. Don’t Repeat Others

Libraries Exist

- If it’s a basic function, someone probably already wrote it, and better than you will.
- BONUS: maintaining the code is not your problem!
- **Corollary:** If you’ve written something useful, consider making it a library, so others don't have to repeat you.

How To Write Easy-to-read Code
4. Document it (Externally)

Documentation lets users look up what the code does without having to look at the actual code.

- What arguments does this function take?
- What does it return?
- Does it throw exceptions?
- What else should I know? (e.g., input restrictions)

How To Write Easy-to-read Code
5. Document it (Internally)

"Comments are always failures… when you find yourself in a position where you need to write a comment, think it through and see if there isn't some way to turn the tables and express yourself in code." - *Clean Code* by Robert Martin

- Code should be able to describe **WHAT** it does without comments
  - Variable names should describe what they **are**
  - Function names should describe what they **do**
  - Easterly's Rule: the length of an object's name should be proportional to its scope
- Comments should explain **WHY** the code is doing what it's doing

How To Write Easy-to-read Code
6. Don’t Be Clever

“Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it.” – *The Elements of Programming Style* by Kernigan and Plaugher

- Don’t rely on implementation, or undefined behavior
- Don’t reinvent the wheel – Libraries Exist
- StackOverflow exists, too
- Be paranoid

No one said there'd be a test!

Does your code work with:

- Different inputs
- Nonsense input
- No input

Assume the user is incompetent or malicious. Ask yourself how they can break your program.