CS2014
Systems Programming

Lectures:
Stephen Farrell
stephen.farrell@cs.tcd.ie, x2354
Room: WR3.4 (impossible to find!)

Teaching Assistant:
Christian Cabrera
cabrerac@scss.tcd.ie
See Christian @ labs
C Programming

• Learn by doing!
• If you do the assignments for the course and revise before the exam, you’ll be just fine
• But, the real goal is that you end up being able to tackle programming tasks as you hit them later in your studies and later in life
• C programming history mirrors that of Unix
• C is a fine language to use in many cases, especially if you have some h/w interfacing to do or need maximal portability, or both.
• Learn by doing! (Did I say that already?)
View #1: steps/waterfall

1. Define the program objectives
2. Design the program
3. Write the code
4. Compile
5. Run the program
6. Test and debug the program
7. Maintain and modify the program

https://www.9wy.net/onlinebook/CPrimerPlus5/ch01lev1sec6.html
A circa 2004 view
View #2: tools/objects

Edit Program

Source Code

Compile

Object Code

Library Files

Link Object Code

Executable

https://www.scss.tcd.ie/David.Gregg/cs2014/notes/ (oops, that URL gets a 403;-)
View #3: version control

- All source objects (.c, .h, Makefiles) are versioned and maybe also object code files, libraries, binaries, and test data/test cases.

- You probably have to deal with lots and lots of versioned things as part of a maybe-large team.

- You almost certainly have to deal with code for some duration (a few years) but that’s only a small part of the overall lifetime of the code (e.g. 20-30 years).
View #4: Test Driven Development

Diagram above: https://upload.wikimedia.org/wikipedia/commons/0/0b/TDD_Global_Lifecycle.png
Also: an “amusing” pressie: https://www.slideshare.net/amritayan/test-driven-development-in-c
CS2014: Not Software engineering, just Programming

• We could continue with software engineering paradigms and methodologies forever

• Some wise people care a lot about this stuff
  – I’m not very wise, so only care a bit:-)

• Meanwhile, let’s learn the rudiments of programming needed in all cases

• We still need a few more concepts first though...
  – Sorry
A few more concepts

- Better drawn on the board than in a slide so I’ll do that as we go...
- Stack, heap, memory management
- Debugging, gdb, “gcc -g”
- Build, Makefile, make
- Inputs, outputs: stdin, stdout, stderr
  - File handling
- Command line arguments, argc/argv
- Expressions and statements: if, while, for, conditionals
- Let’s learn as we go...
C Programming Examples

• Ok, let’s go play with some examples
• So with that, we’ll head over to:
  

And