CSC230
Getting Starting in C

Tyler Bletsch

NC STATE UNIVERSITY

What is C?

- The language of UNIX
- Procedural language (no classes)
- Low-level access to memory
- Easy to map to machine language
- Not much run-time stuff needed
- Surprisingly cross-platform

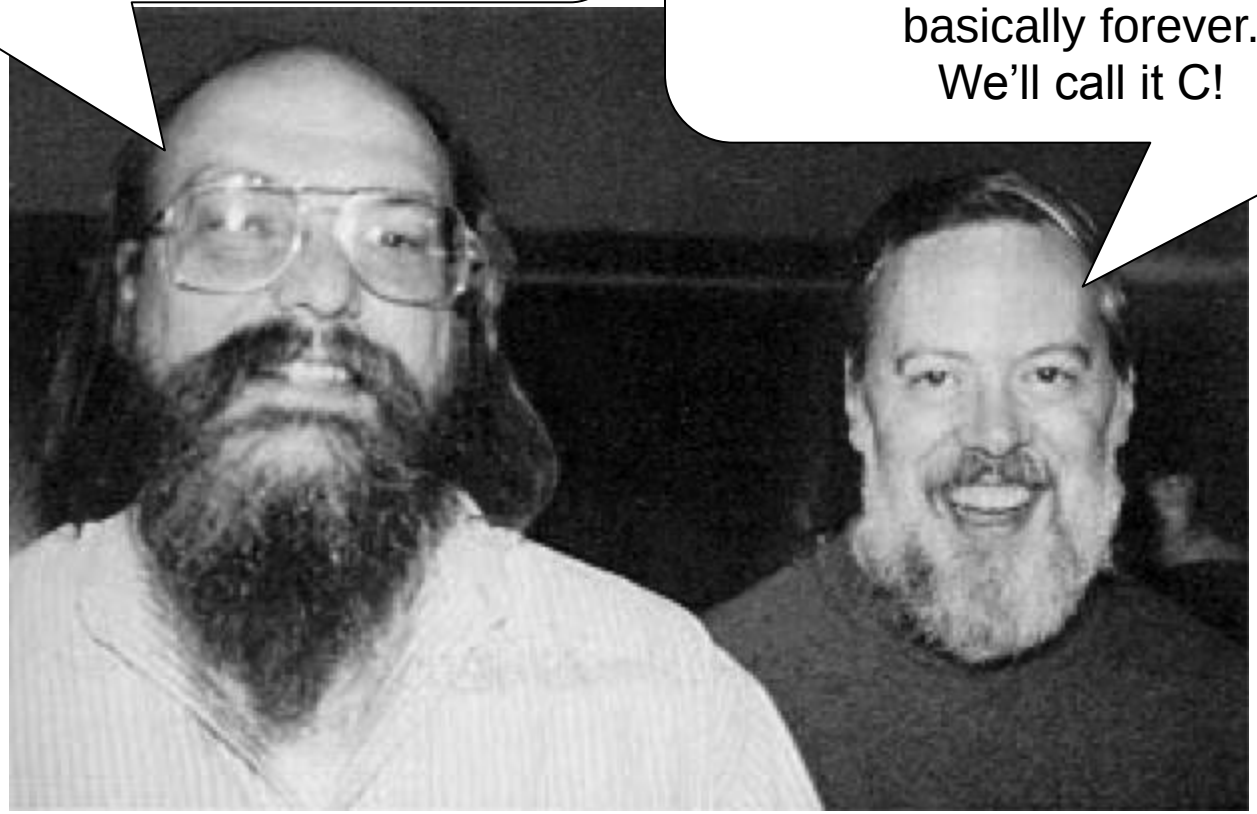
Why teach it now?

To transition from basic programming to Operating Systems (CSC246), Software Engineering (CSC326), etc.

The Origin of C

Hey, do you want to build a system that will become the gold standard of OS design for this century? We can call it UNIX.

Okay, but only if we also invent a language to write it in, and only if that language becomes the default for all systems programming basically forever. We'll call it C!



Ken Thompson

Dennis Ritchie

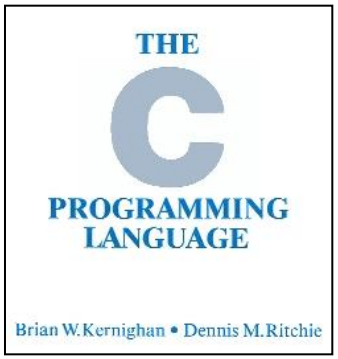
AT&T Bell Labs, 1969-1972


Cool, it worked!
Told ya.

What were they thinking?

- Main design considerations:
  - Compiler size: needed to run on PDP-11 with 24KB of memory (Algol60 was too big to fit)
  - Code size: needed to implement the whole OS and applications with little memory
  - Performance
  - Portability
- Little (if any consideration):
  - Security, robustness, maintainability
  - Legacy Code

C vs. other languages



Most modern languages	C
Develop applications	Develop system code (and applications) (the two used to be the same thing)
Computer is an abstract logic engine	Near-direct control of the hardware
Prevent unintended behavior, reduce impact of simple mistakes	Never doubts the programmer, subtle bugs can have crazy effects
Runs on magic! (e.g. garbage collection)	Nothing happens without developer intent
May run via VM or interpreter	Compiles to native machine code
Smart, integrated toolchain (press button, receive EXE)	Discrete, UNIX-style toolchain make → gcc (compilation) → gcc (linking) (even more discrete steps behind this)


```
$ make
gcc -o thing.o thing.c
gcc -o thing thing.o
```

Why C?

- It's a "portable assembly language"
- Useful in…
  - Systems development: OS & Embedded
  - Optimized routines for use with other languages
  - Need for speed, size, or predictability
- Notable pure C software:
  - UNIX and Linux – kernel and most utilities
  - NetApp Data ONTAP (most common storage OS)
  - Python, Perl, PHP, Java*, Ruby*
  - A bajillion applications:
    - http://en.wikipedia.org/wiki/Category:Free_software_programmed_in_C

* With some C++ as well

Example C superpowers

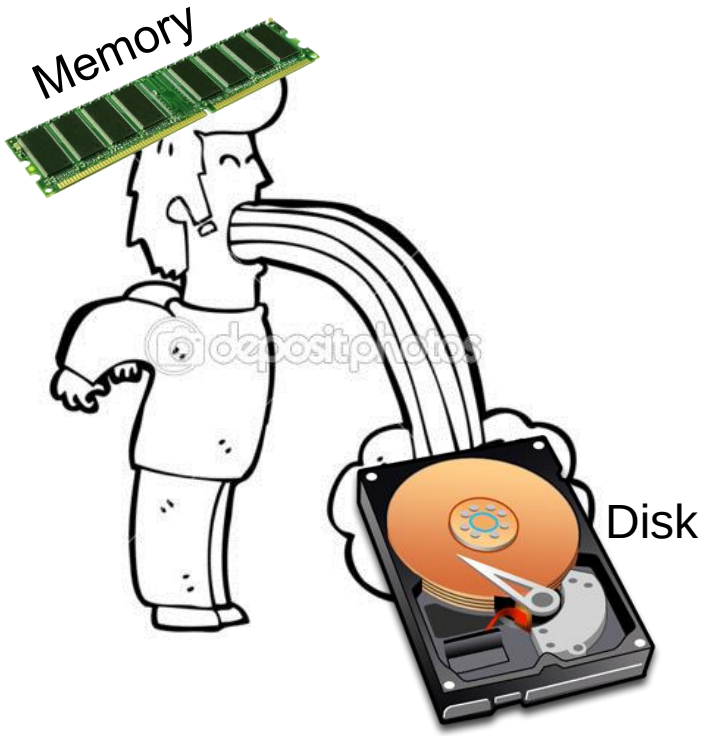
Task: Export a list of coordinates in memory to disk

Most languages

- Develop file format
- Build routine to serialize data out to disk
- Build routine to read & parse data in
- Benchmark if performance is a concern

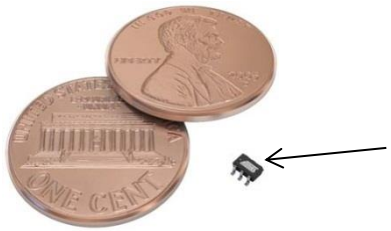
C

- Read/write memory to disk directly



Example C superpowers

Task: Blink an LED



Atmel ATTINY4 microcontroller :
Entire computer (CPU, RAM, & storage)!
1024 bytes storage, 32 bytes RAM.

```
led = 0
while (true):
   led = NOT led
   set_led(led)
   delay for 1 sec
```

Language	Size of executable	Size of runtime (ignoring libraries)	Total size	RAM used
Java				
Python				
Desktop C				
Embedded C (Arduino)				

Max: 1024 B

Max: 32 B

What about C++?

- Originally called “C with Classes” (because that’s all it is)
- All C programs are C++ programs, as C++ is an extension to C
- Adds stuff you might recognize from Java (only uglier):
  - Classes (incl. abstract classes & virtual functions)
  - Operator overloading
  - Inheritance (incl. multiple inheritance)
  - Exceptions

Bjarne Stroustrup developed C++ in 1979 at Bell Labs

OUT OF SCOPE

C and Java: A comparison

C

```c
#include <stdio.h>
#include <stdlib.h>

int main(int argc, const char* argv[]) {
    int i;

    printf("Hello, world.\n");

    for (i=0; i<3; i++) {
        printf("%d\n", i);
    }

    return EXIT_SUCCESS;
}
```

```
$ gcc -o thing thing.c && ./thing
Hello, world.
0
1
2
```

Java

```java
class Thing {
  static public void main (String[] args) {
    int i;

    System.out.printf("Hello, world.\n");

    for (i=0; i<3; i++) {
      System.out.printf("%d\n", i);
    }

  }
}
```

```
$ javac Thing.java && java Thing
Hello, world.
0
1
2
```

Common Platform for This Course

- Different platforms have different conventions for end of line, end of file, tabs, compiler output, …
- Solution (for this class): compile and run all programs consistently on one platform
- Our common platform:

NCSU Linux Machines!

Don't you gimme no "it worked on my box" nonsense!

Your Choices

Option	Use GUI-based Editor?	Access to your unity Filespace?	Matches grading environment?
Use Unity Lab Computer	Y	Y	Y
ssh to VCL (linux)	N**	Y	Y
ssh to remote-linux.eos.ncsu.edu	N**	Y	Y
Use Mac OS X (+developer tools)	Y	sftp*	N
Use MS Windows + cygwin	Y	sftp*	N
Use Linux on your PC (dual boot or virtualized)	Y	sftp*	N

* direct if you install realm kit

** Yes if you run X windows server on your computer

Hello world

C

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, const char* argv[]) {
    int i;

    printf("Hello, world.\n");

    return EXIT_SUCCESS;
}
```

File with library function declarations

Entry point of the program, with command line arguments

Standard library function, with message argument

Exit program and indicate successful completion

```
$ gcc -Wall -std=c99 -o hello hello.c
$ ./hello
Hello, world.
```