CS 521: Systems Programming

Beginning C

Lecture 3

Today's Schedule

- Differences: C vs. Other Languages
- Editors
- Our First Program

Today’s Schedule

- **Differences: C vs. Other Languages**
- Editors
- Our First Program

Architectural Differences

- C is compiled to ***machine code***, unlike Python or Java
  - The compiled ***binary executable*** contains instructions that your CPU understands
  - There are several compilers on the market today (gcc, clang, msvc) that transform your code into machine code
- Java runs on a virtual machine (JVM)
- Python is interpreted (translated to machine code on the fly)
- We can achieve better performance with C, but are also given more responsibility
  - Memory management is up to us (no automatic garbage collection)

Main Advantages

- C is fairly simple: the language does not have a multitude of features
- But coming from Java, the syntax is still familiar
- It's the *lingua franca* of systems programming
  - When we operate close to the hardware, it can be much easier to implement than the equivalent Java/Python/etc.
  - Want to contribute to the Linux kernel? It's written in C (including the drivers)
- Performance

Main Disadvantages

- Much less functionality is available in the standard library than other languages
  - For example: no built in list, hashmap, tree, etc.
- Memory leaks
- Segmentation faults (invalid memory access)
- No objects – if you're used to object-oriented programming, C will make you rethink your program

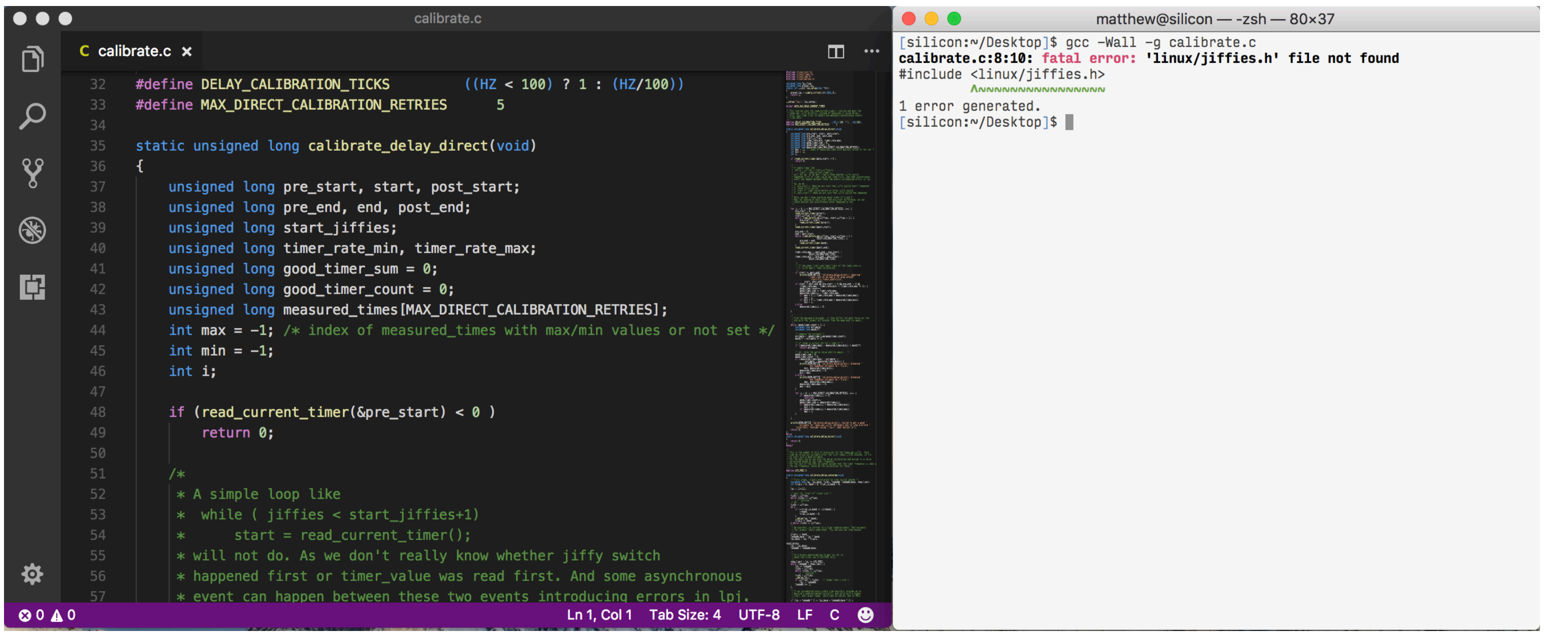
Standardization

- C is not controlled by a single entity; it is a standard
- The standard itself is fairly loose, and allows **undefined behavior** (UB)
  - Basically, the language standard doesn't specify how *everything* should work
    - Compilers can do whatever they want with UB
    - This is why we're making sure we all have the same platform (our VMs) in class

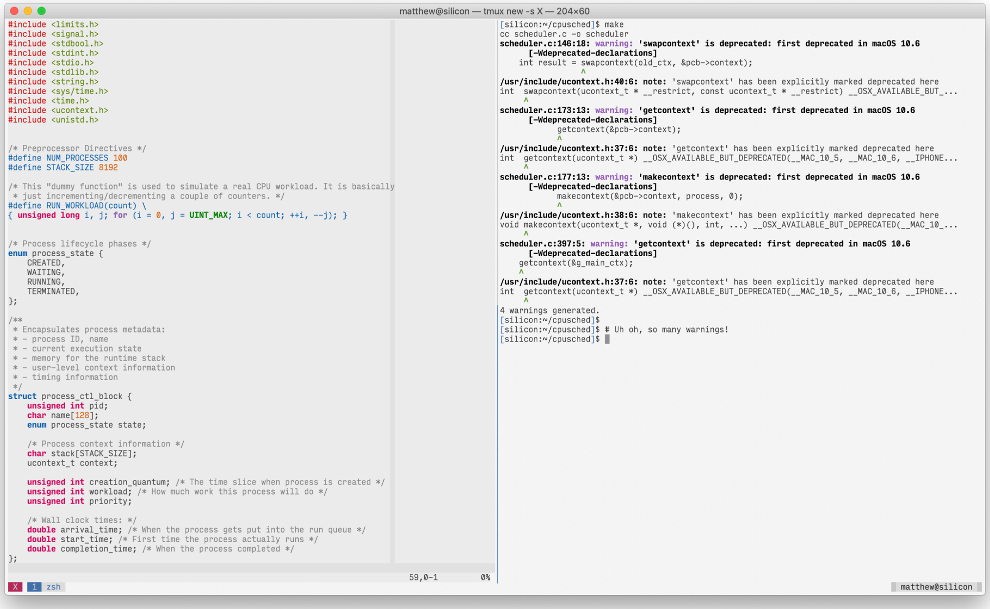
Today's Schedule

- Differences: C vs. Other Languages
- **Editors**
- Our First Program

Writing C Programs



Writing C Programs



Systems Culture

- There is a somewhat different culture in the systems world
- Using an IDE (like Eclipse, IntelliJ, etc) is less common
  - The Unix command line provides many of the usual IDE features
- Many developers prefer to use a text editor and a terminal to write their programs
  - Text editor: edit, save
  - Terminal: compile, run

Recommendation

- Use whatever is comfortable for you
- If you get a chance, try to learn the basics of a terminal editor (even `nano` counts!)
  - Or `vim`, `emacs`, `micro`
- (maybe at least know how to quit vim and emacs…!)
  - By the way, what's the universal "quit" key combination in the terminal?

Today's Schedule

- Differences: C vs. Other Languages
- Editors
- **Our First Program**

Hello World

```
#include <stdio.h>

int main(void)
{
    printf("Hello world!\n");
    return 0;
}
```

...and to run it:

```
cc hello.c -o hello
./hello
```

Slightly More Advanced

```c
#include <stdio.h>

void say_hello(int times);

int main(void) {
    say_hello(6);
    return 0;
}

void say_hello(int times) {
    int i;
    for (i = 1; i <= times; ++i) {
        printf("Hello world! (#%d)\n", i);
    }
}
```

Differences from Java/Python

- Including libraries looks a bit different
- No public/private etc. access modifiers
- Forward declarations (prototypes)
- No objects
- No exceptions
- A **huge** difference: what return types are used for
  - Often error checking!
- But, there are a lot of similarities…

Similarities to Java/Python

- Arithmetic is mostly the same
- We use &&, ||, and != instead of and, or and not
- if, then, else
- Loops
- Switches

Some Advice

- The similarity between C and Java can be deceiving
- In these small programs, there's hardly a difference!
- However, you will soon see that the structure of larger programs ends up being quite different
  - Since there are no classes, the focus shifts to writing functions
  - Organization might seem a bit less natural, but you can still break your functions up into ***modules***