CS 521: Systems Programming

Beginning C & VM Setup

Lecture 2

Announcements / Reminders

- Password changes this semester

Today's Schedule

- Differences: C vs. Other Languages
- VM Setup

Today's Schedule

- **Differences: C vs. Other Languages**
- VM Setup

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

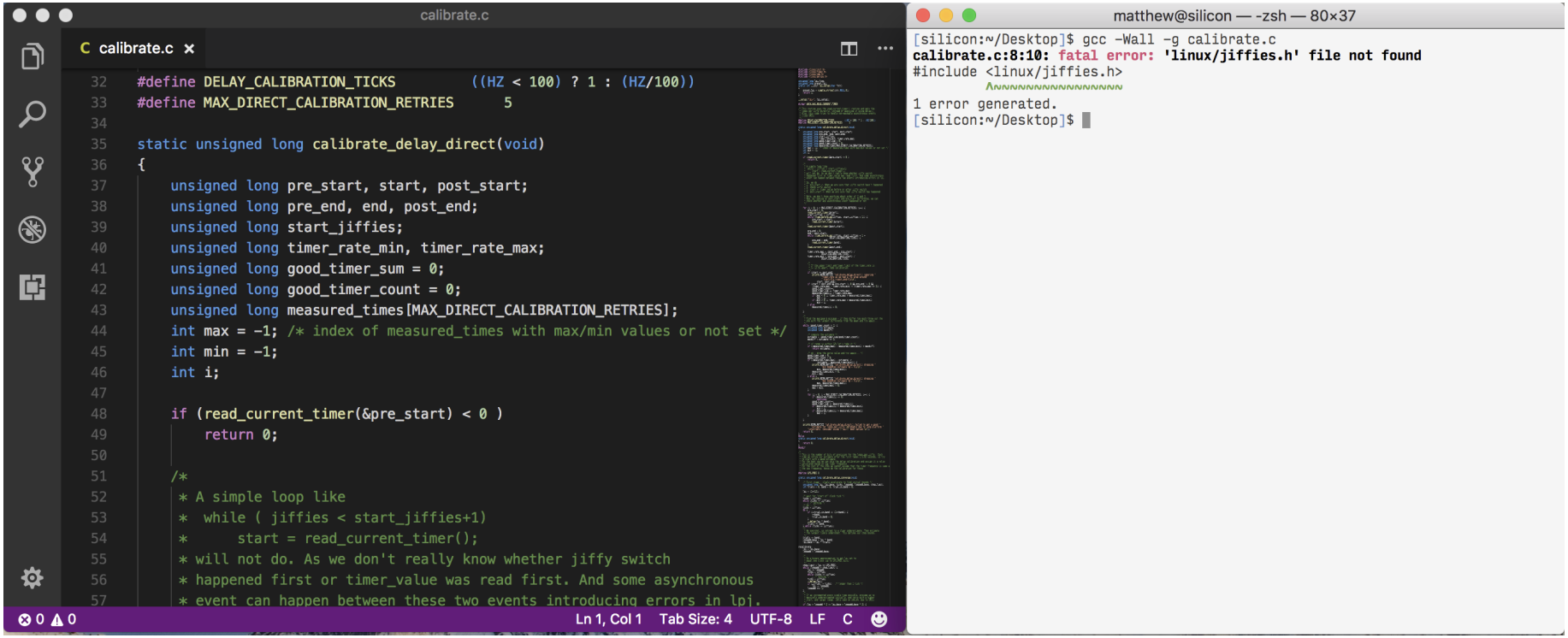
Standardization

- 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 😉

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

Writing C Programs



Revisiting Hello World

```c
#include <stdio.h>

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

To run:

```
gcc 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

- Whitespace is mostly ignored
- Semicolons are required
- Comments: /* */ and //
- Including libraries looks a bit different
- No public/private etc. access modifiers
- Forward declarations (prototypes)
- But, there are a lot of similarities…

Similarities to Java/Python

- Arithmetic is mostly the same
- We use &&, ll, 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*
- Ok, with that out of the way, let's set up our development environment!

Today's Schedule

- Differences: C vs. Other Languages
- **VM Setup**

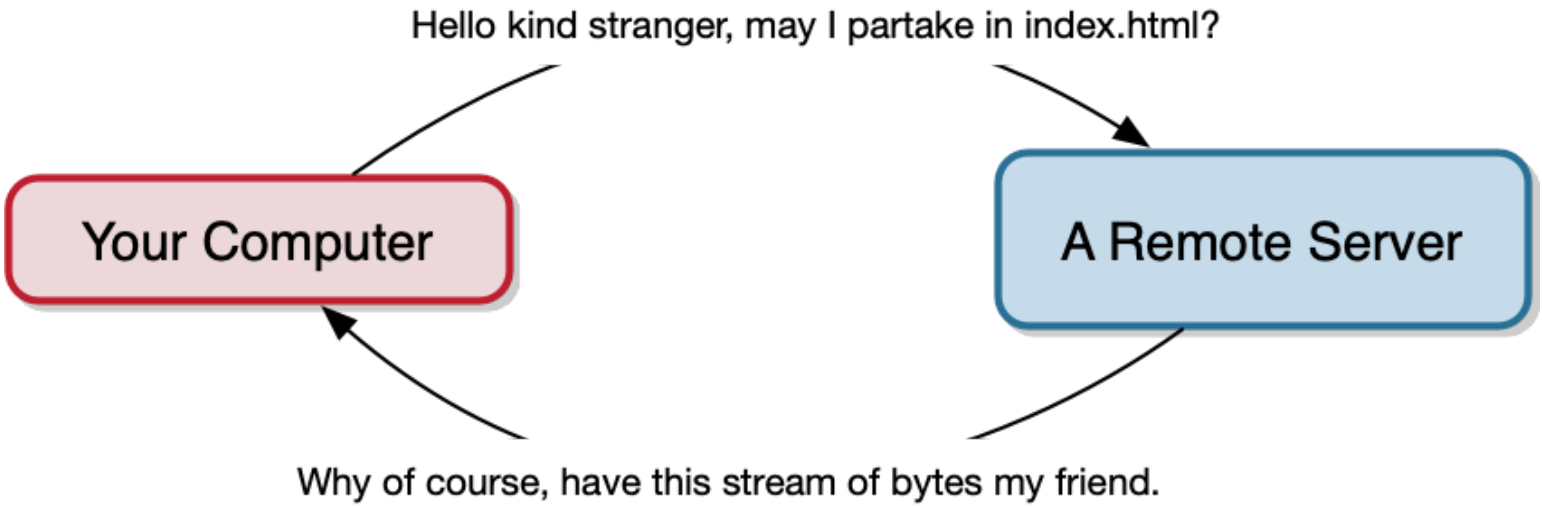
Setting up a VM

- Today we'll create your very own VM!
  - So unique and amazing… and configured just like everybody else's
- You need some information for this:
  - **Most important**: a name for your VM
  - Your CS username
  - Your CS password
  - Your CS 521 Classroom ID
    - Determines the MAC address of your network interface

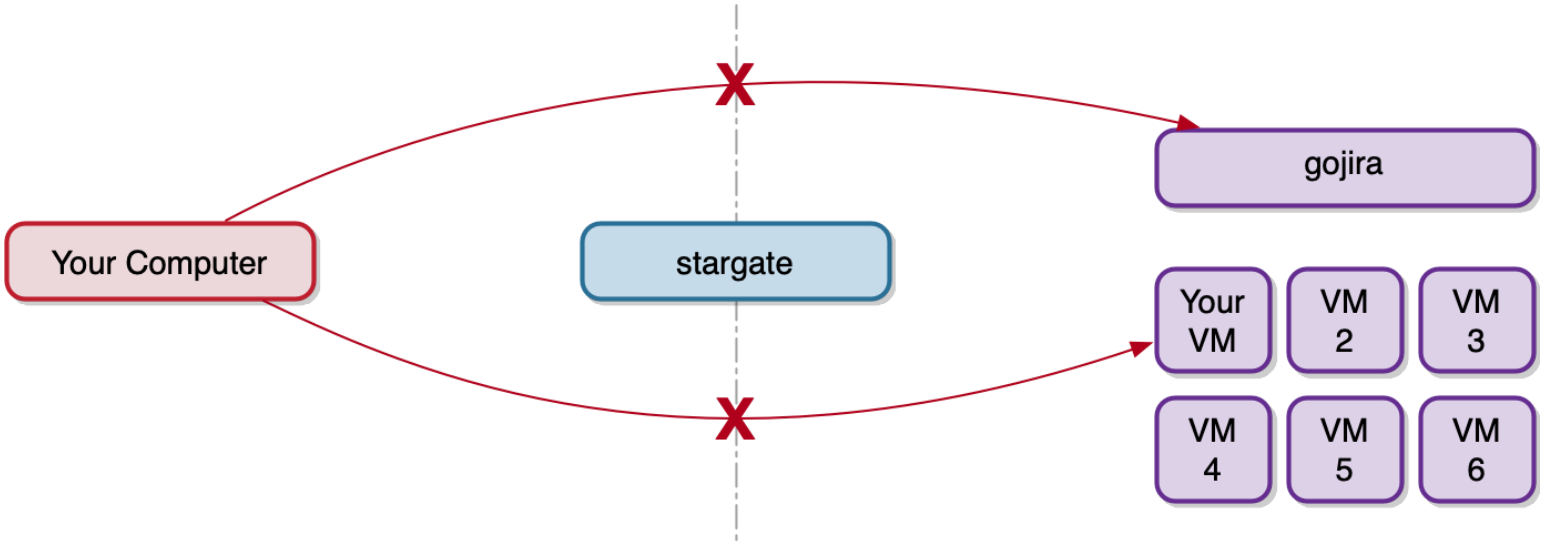
Important: Networking

- The VMs we create will be more or less the same as any other computer
- The only catch: they're locked up behind a firewall or two
  - We don't want them to escape and start posting on Twitter
- Let's look at a very simplistic model for network communication…

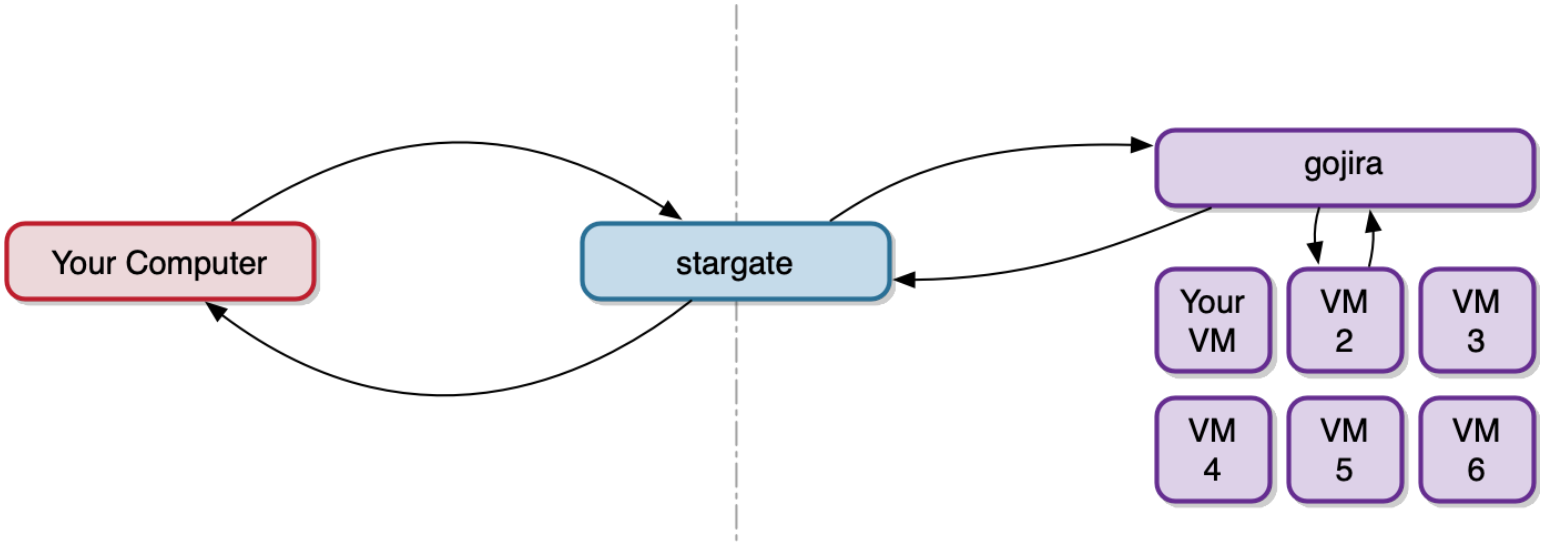
Normal Communication



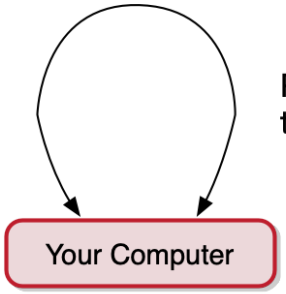
Our Situation



SSH Tunneling

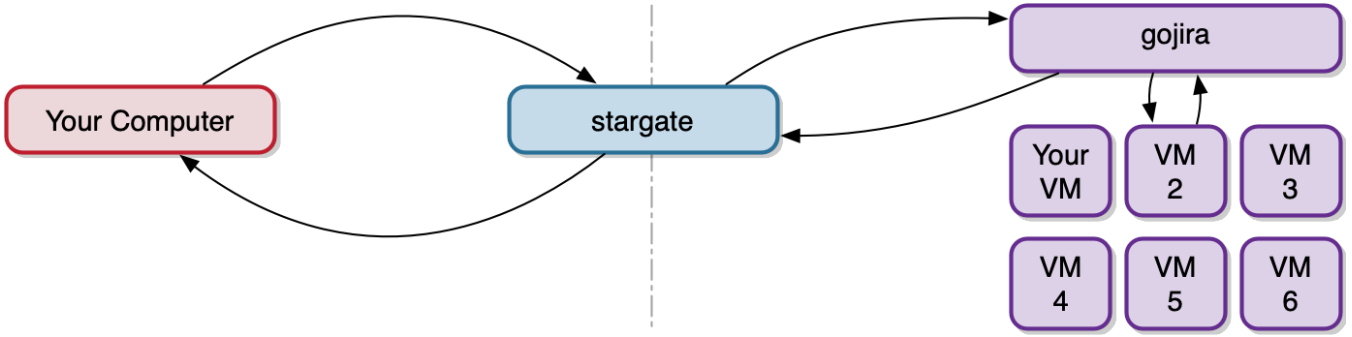


SSH Tunneling: The Result



Programs think they are talking
to your own computer (localhost)…

…but **this** is actually happening any time there is communication:



Important

- You'll need passwordless SSH set up
- At the end of the day, you should be able to type `ssh myvmname` and be logged in without typing a password or doing additional steps
  - Using this, you can configure remote editing (e.g., with VSCode)
  - You will also be able to connect directly to your VM with SFTP via a tunnel

Demo: ssh

Let's check out passwordless SSH and one option for transferring files…

Lab 1: Grading

- You can work in teams of up to four students if you'd like
- As you set up your VM, take note of any new (or old!) commands/concepts
  - Add these to your lab notebook (`username-concepts.txt`)
- Show that **all** the teammates VMs are up and running and that you can connect to them with a single command
  - `ssh VMNAME`
    - No password should be required to log in!