Introduction and Background

Tom Kelliher, CS 411

1 Administrivia

Announcements

GitHub usernames?

Assignment

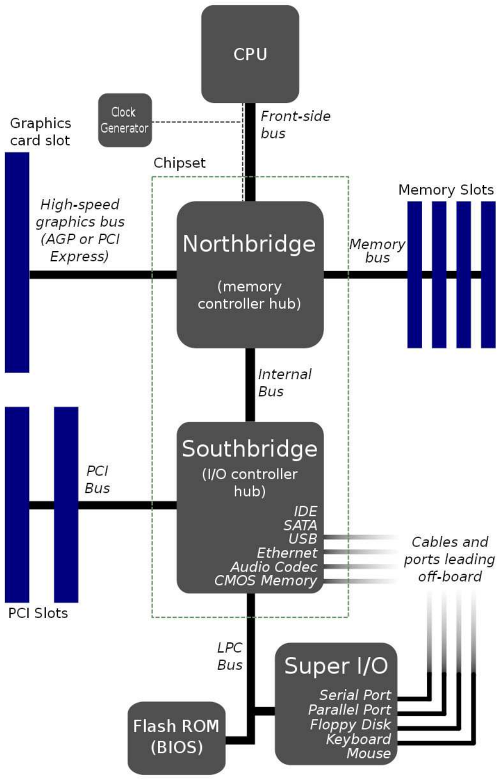
Outline

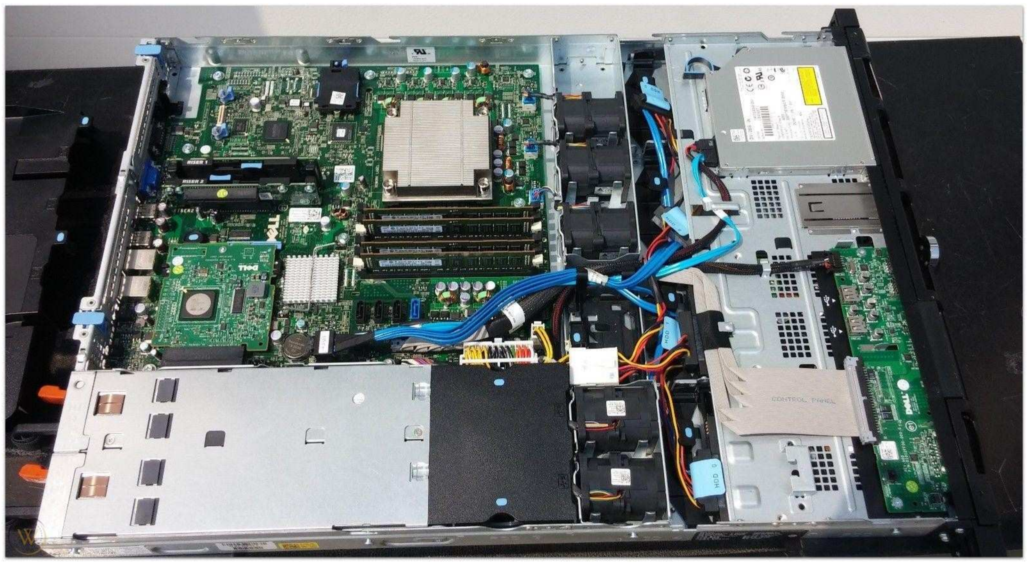
1. What's Inside a Computer?
2. From C Source Code to Running Program
3. Linux Command Line
4. Linux Man Pages
5. Structure of a C Program
6. Compiling and Running a C Program

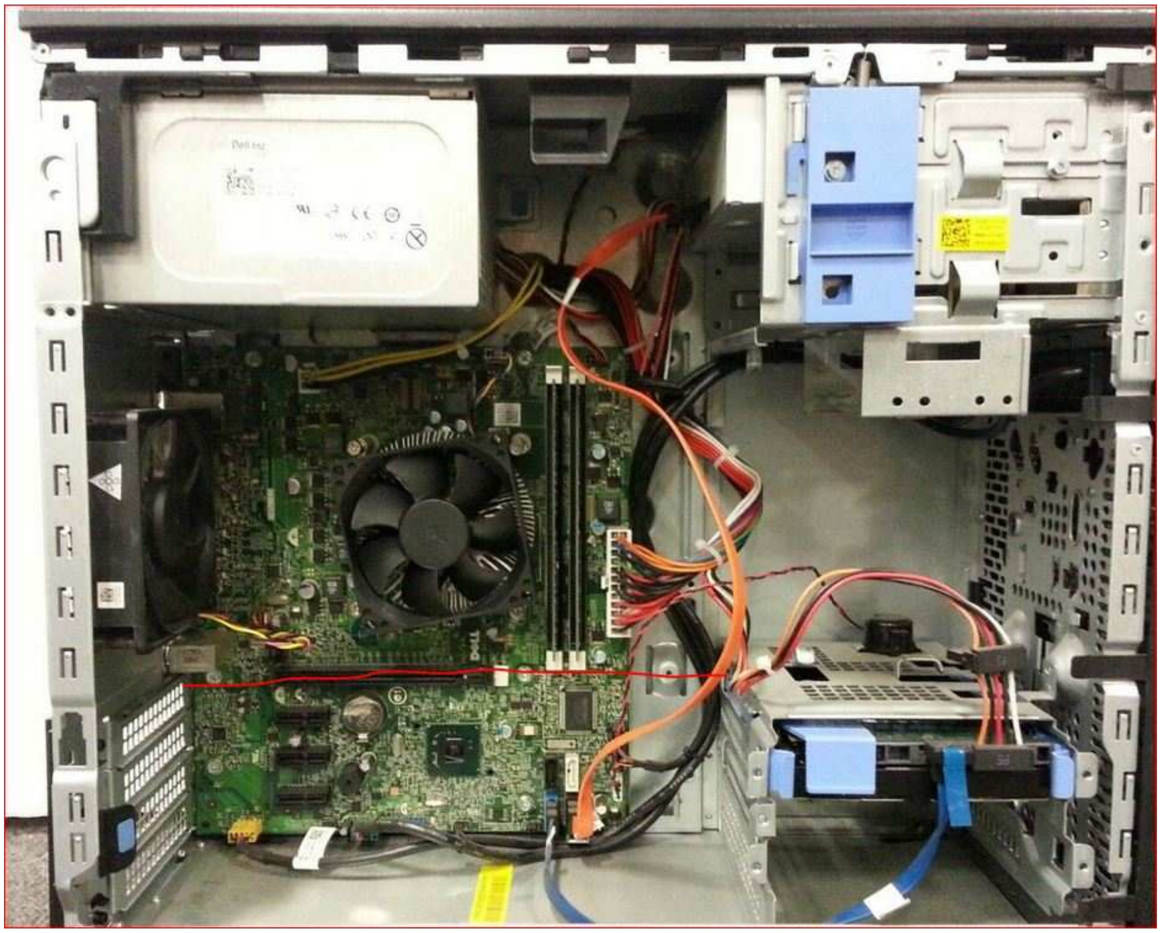
Coming Up

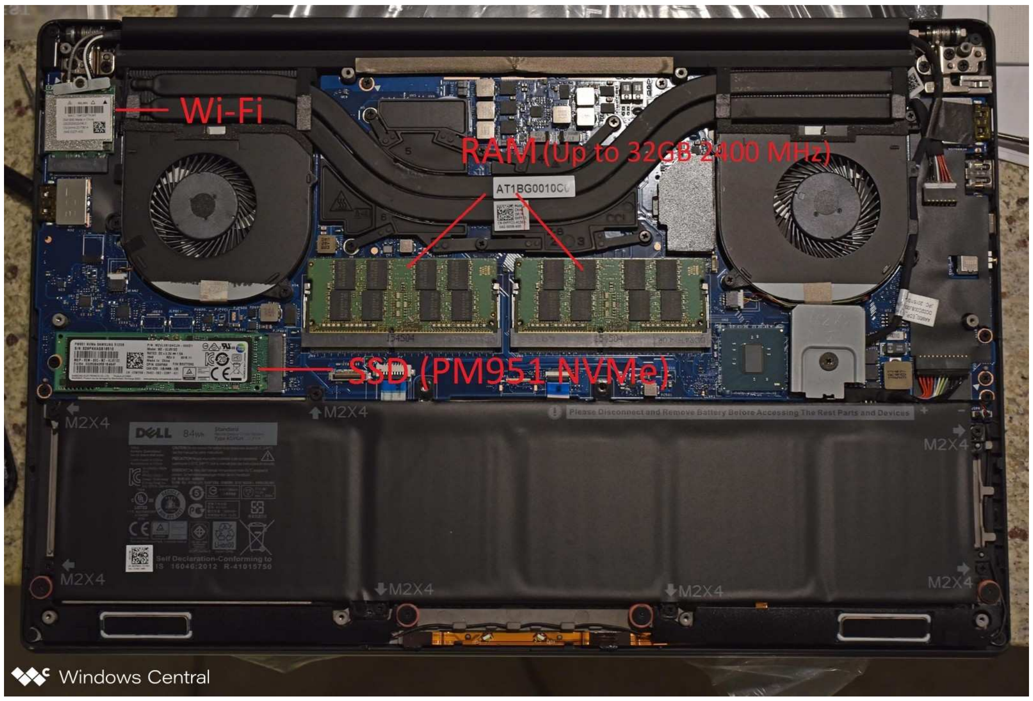
Continuing...

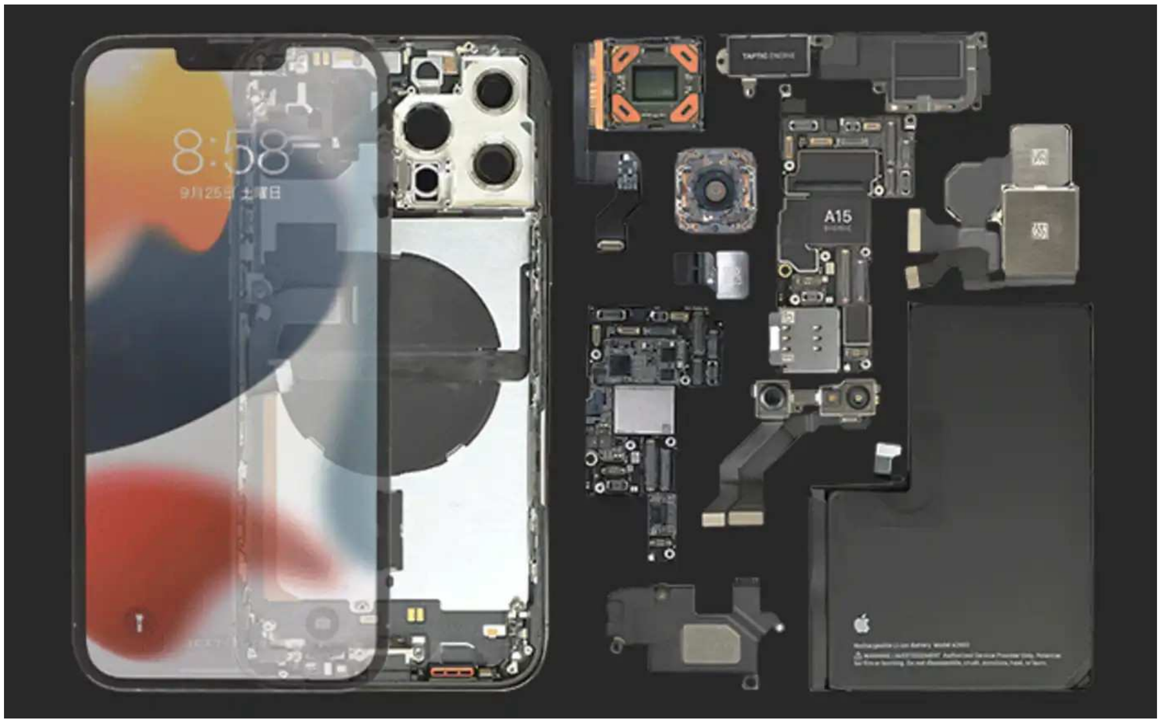
2 What's Inside a Computer?



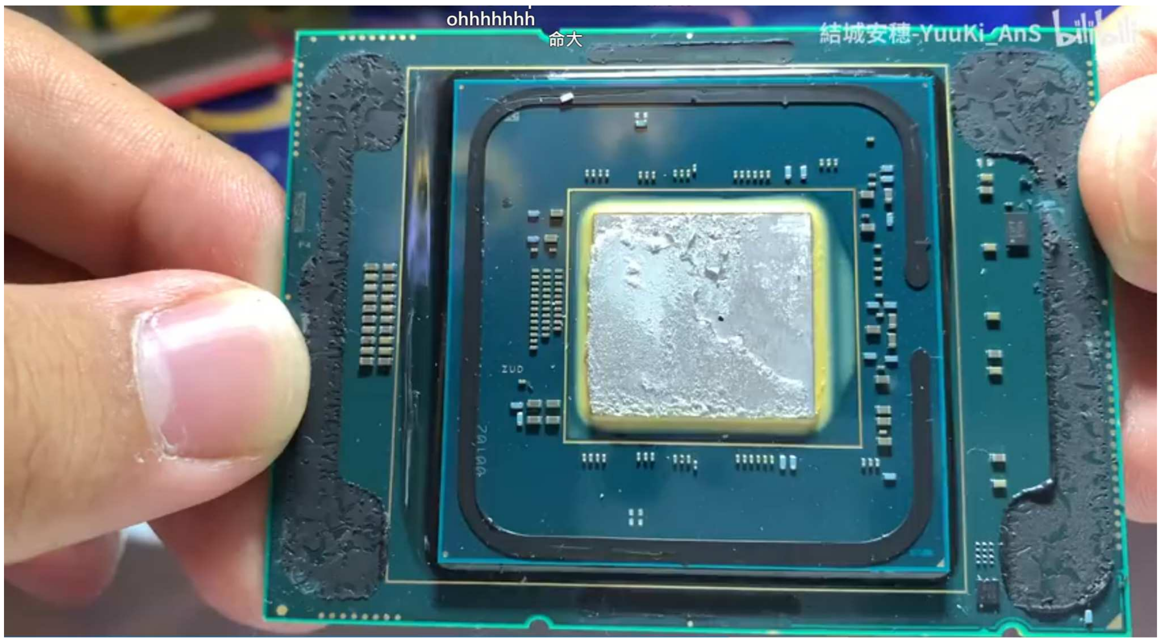


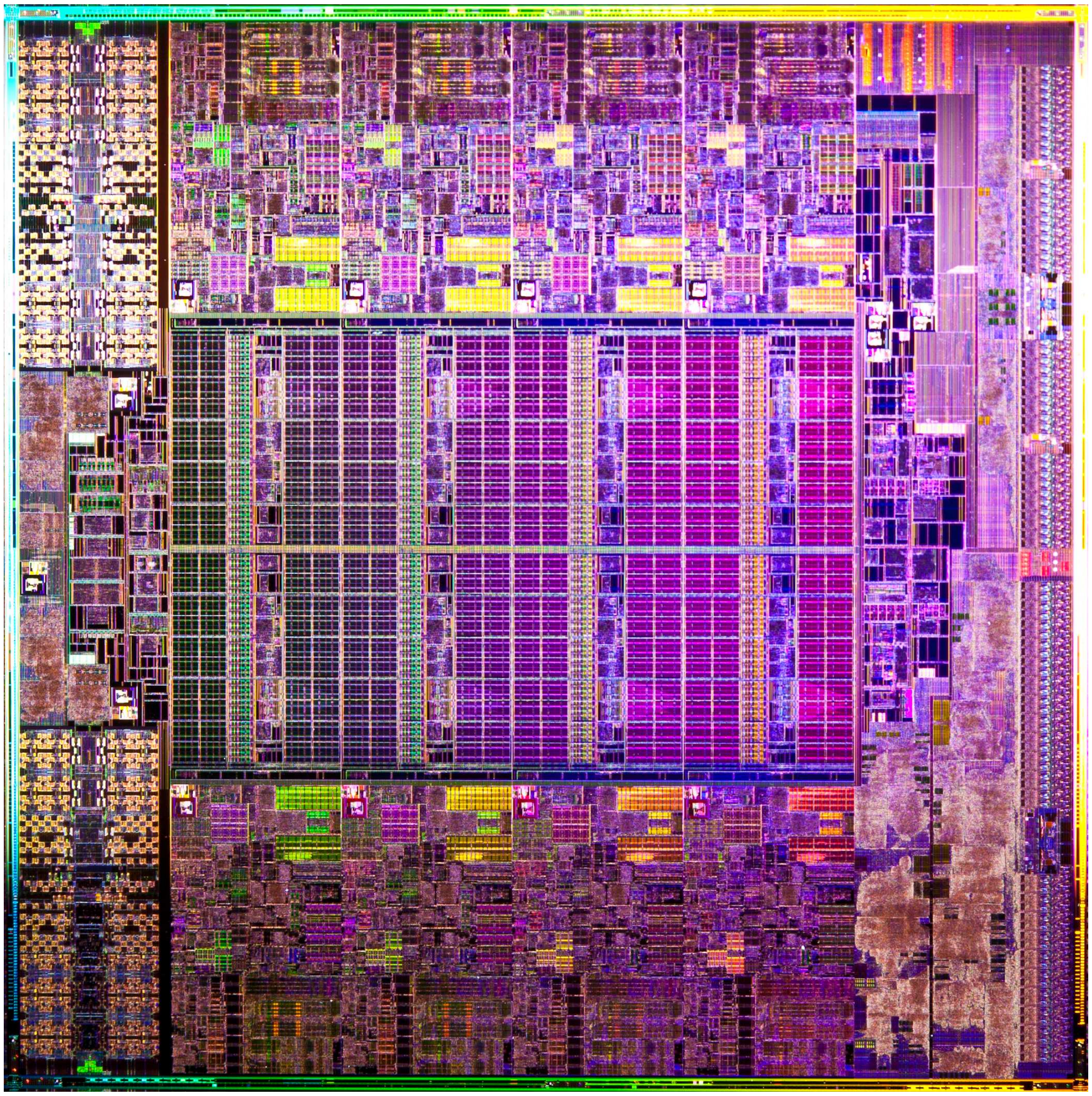



Wi-Fi
RAM (Up to 32GB 2400 MHz)
SSD (PM951 NVMe)
Windows Central

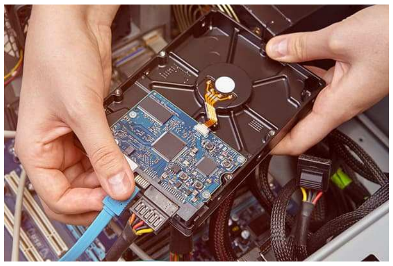

TAPTIC ENGINE
A15
8:58

TAPTIC ENGINE
Li-ion


ohhhhhhh
命大
結城安穗-YuuKi_AnS



Proudly American
LIFETIME
MEMORY PRODUCTS INC.
16GB DDR3 1866 ECC REG DIMM (PC3 – 14900)
WARRANTY VOID IF REMOVED (23488) x4
www.lifetimememory.com



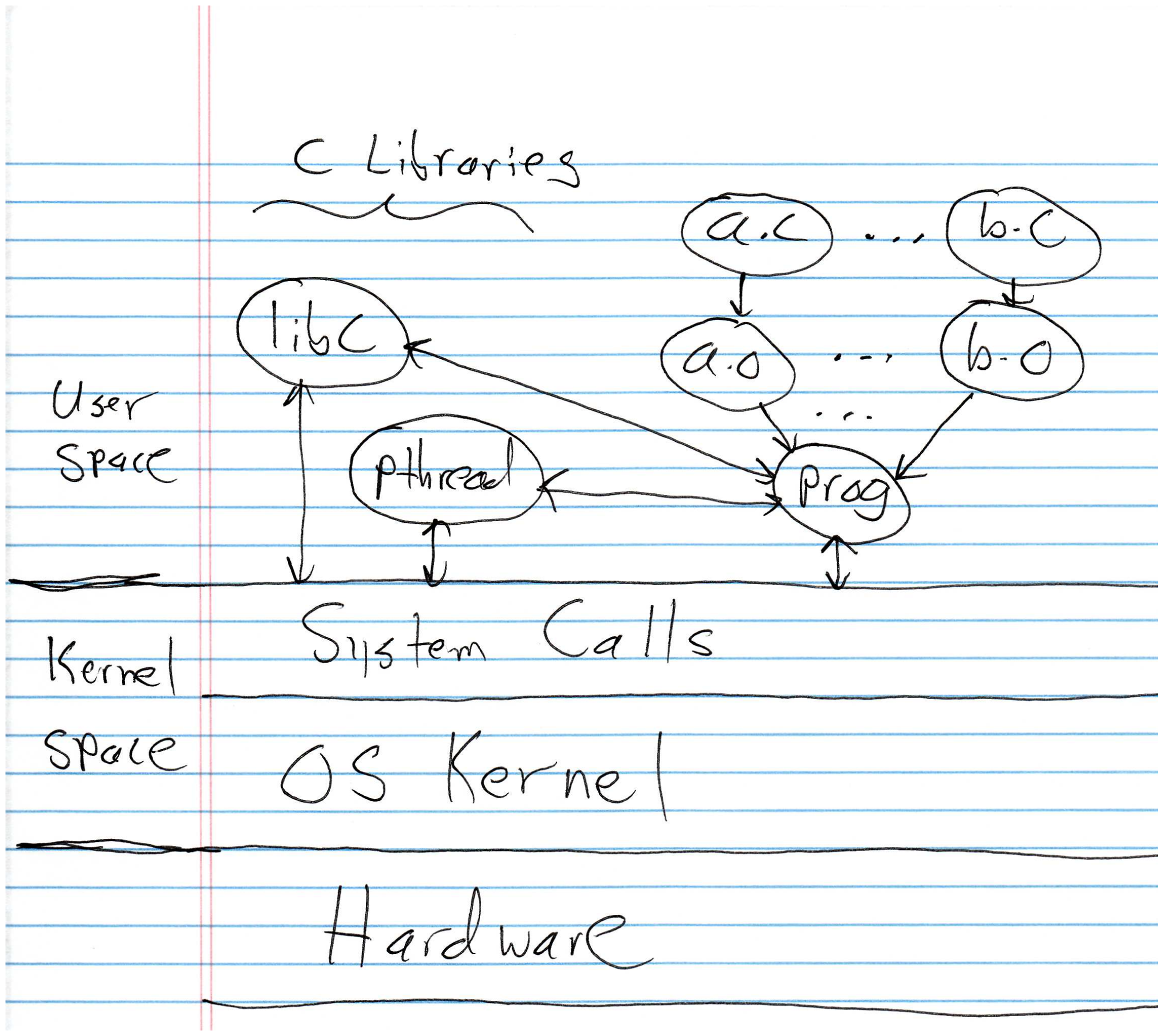
3 From C Source Code to Running Program

Components:

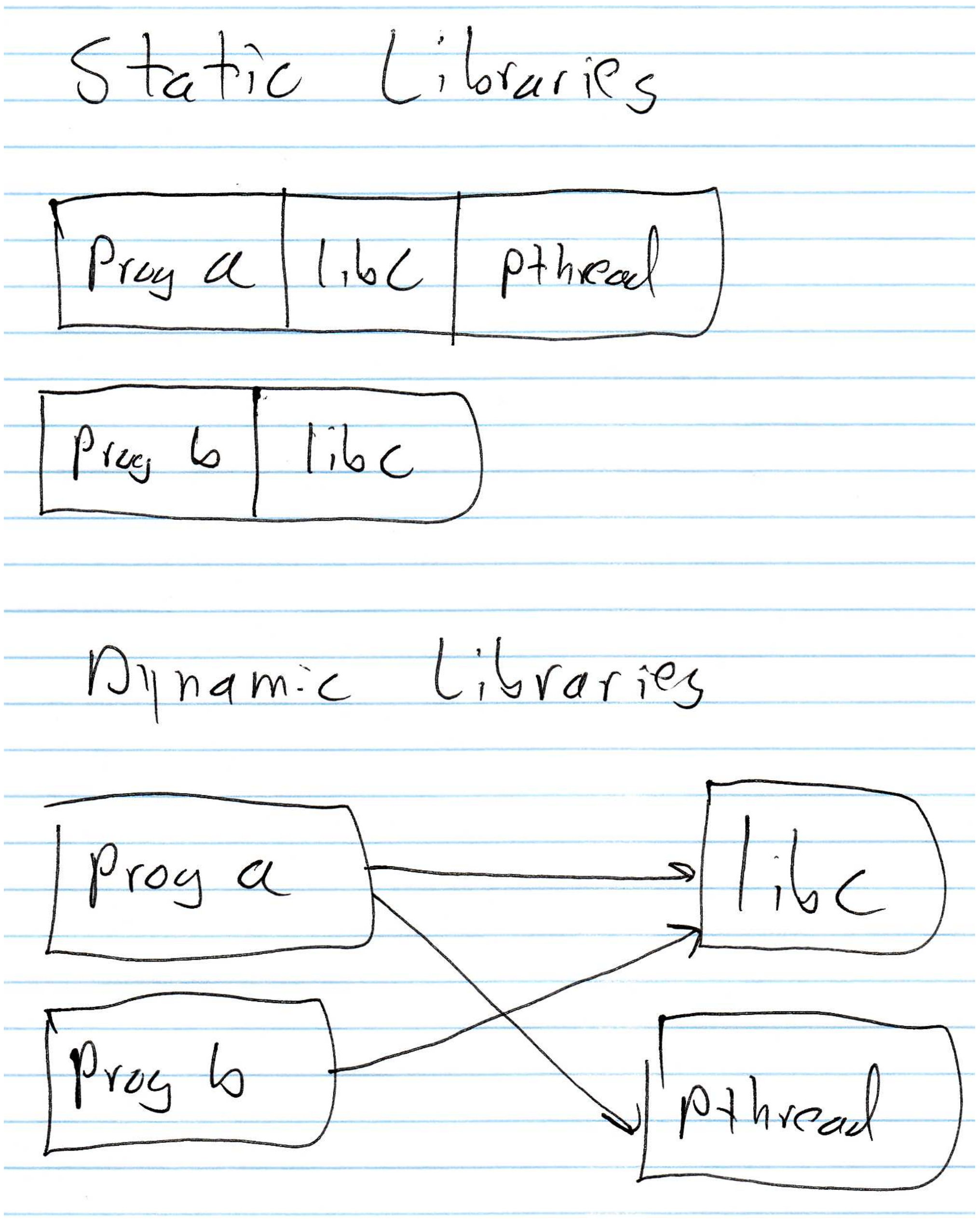
- One or more C source files (.c)
- C libraries — libc (printf, exit, etc.) and possibly others (pthread)
- Linux system calls — fork, open, creat
- Device drivers — NVIDIA, for example

Process:

- Compile
- Assemble
- Link
- Load



4 Static vs. Dynamic Libraries



5 Linux Command Line Interface

- Paths
  - Absolute:
    `/`
    `/usr/local/bin`

```
    /home/kelliher
    /home/jillz
```

  – Relative:

```
    .
    ..
    ~/ ~<username> local          # Assume we're at /usr
    local/bin     # ditto
    ../kelliher   # Assume we're at /home/jillz
```

- ls
  ```
  ls <path>
  ```

- cd
  ```
  cd <path>
  mkdir <path>
  pwd
  cp <source> <dest>
  mv <source> <dest>
  rm <file>...    # There's no turning back...
  rmdir <directory>   # <directory must be empty; see next command
  rm -rf <directory>  # Powerful AND dangerous
  ./hello
  ```

6 Linux Man Pages

Sections:

- Section 1: Shell programs
- Section 2: System calls
- Section 3: Library calls

From `man man`

Examples: `printf`, `pthread_create`

7 Structure of a C Program

hello.c

```
/*
    This is how constants are defined in C.  The convention is to use only
    uppercase letters, numerals, and the underscore character in names.
*/
#define FAVE 42

// <...> tells include to look in system include directories
#include <stdio.h>
#include <stdlib.h>
// "..." tells include to look in the current working directory
//#include "more.h"

// Function prototype
//int hello(char *, int);

int main(int argc, char *argv[]) {
    long fave = FAVE;
    printf("Favorite number is %d\n", fave);
//    func(fave);
    return hello(argv[1], atoi(argv[2]));
}

int hello(char *name, int count) {
    printf("Hello %s!\nCount is %d\n", name, count);
    return 0;
}
```

more.h

```
/*
    The #ifndef/#define/#endif ensure that no recursive inclusions of this
    .h file occur.  It's standard practice to do this with all .h files.

    Typically, .h files are used to define those things that need to be
    defined/included for the associated .c files.  Thus, they will include
    #define and #include directives, as well as the function prototypes for
    the functions defined in the .c file.
*/

#ifndef _MORE_H
#define _MORE_H

#include <stdio.h>

void func(int);
```

```
#endif
```

more.c

```
#include "more.h"

void func(int val) {
    printf("func’s param is %d\n", val);
}
```

8 Compiling and Running a C Program

We’ll switch-over to the “Git Setup” document for this.