Hello World!

Lecture 2
Jan 12th 2023 | COMP 211-002 | Joshua Bakita

Welcome!

Today:

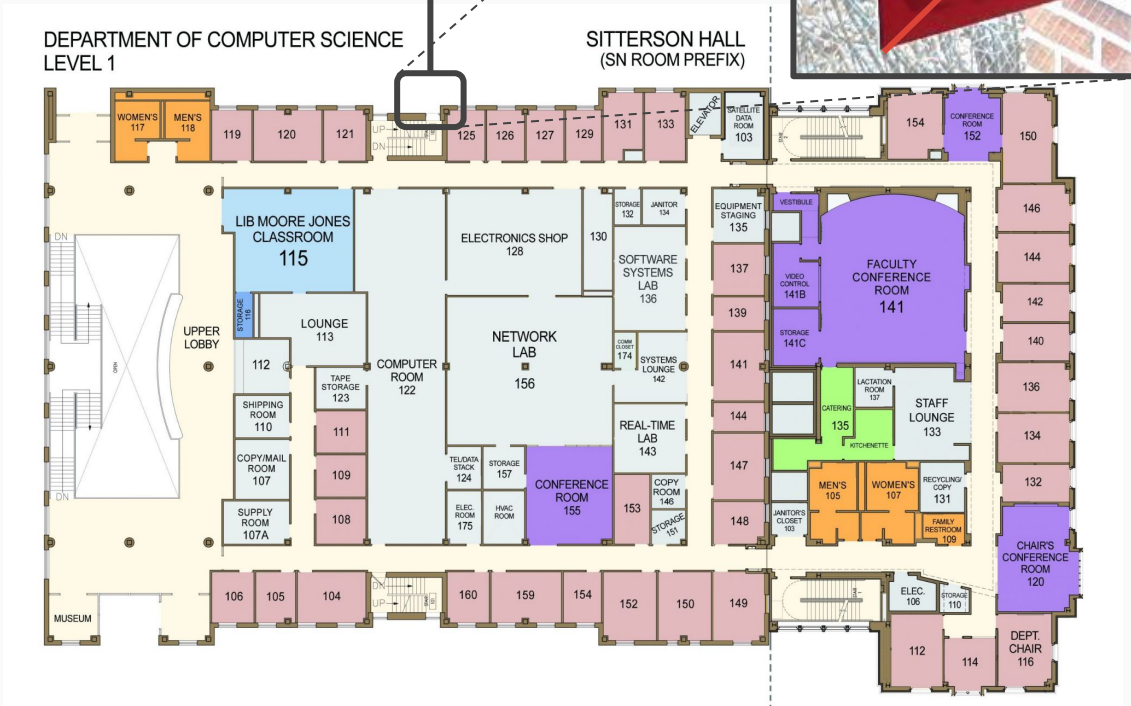
- C versus Java
- Anatomy of a C Program
- Command Line Essentials
- Assignment 1 Overview

Logistics:

- Choose a seat and stick with it! We'll be looking for you in the same location each week for attendance.
- Piazza and Sakai up.
- Assignment 1 up tonight.

Did you know...

This fire exit in the northeast stairwell is not actually alarmed



C Versus Java

Comments (`//` or `/* */`)	Identical
Operators (`+`, `-`, `/`, `*`, `%`, `=`, ...) and ordering/precedence	Largely the same
Scoping and Nesting (`{}`)	Largely the same
Function calls (`my_func(my_param)`)	Largely the same
Control Flow (`if`, `else`, `for`, `while`, `do`, `switch`)	Very similar
Array Access (`my_array[i]`)	Similar
Function declaration (`int my_func(int my_param)`)	Similar
Input/Output (println?)	Different
Types (int, char, struct, arrays…)	Very different
Memory Management (new?)	Very different
Data Structures (classes? objects?)	Completely different

Today

Next Week

Following Weeks

Anatomy of a Single-File C Program

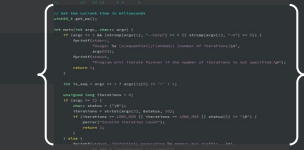
From my research: `thrasher.c`

```c
/**
 * Copyright 2021 Joshua Bakita
 * Created September 19 2019
 * Description: This program is designed to stress the i.MX6 Quad memory bus
 * and controller as much as possible by purposely generating cache misses.
 */
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>
#include <string.h>
#include <sys/time.h>

// L1 is 8-way with 64 lines/way
#define LINE_SIZE 64 // 8 64-bit words per line in L1, L2, and L3; 64 bytes
#define L3_SIZE 16*16384*64 // 16 ways, 16384 lines/way, 64 bytes/line

// Each piece of data can go in one of 16 ways
// Which bits do what for the L3?
// [ ---  |  way  |  line  |  byte  ]
//         23   20 19     6 5      0

// Get the current time in milliseconds
uint64_t get_ms();

int main(int argc, char** argv) {
    if (argc >= 2 && (strcmp(argv[1], "--help") == 0 || strcmp(argv[1], "-h") == 0)) {
        fprintf(stderr,
                "Usage: %s [s(equential)/r(andom)] [number of iterations]\n",
                argv[0]);
        fprintf(stdout,
                "Program will iterate forever if the number of iterations is not specified.\n");
        return 1;
```

```c
/**
 * Copyright 2022 Joshua Bakita
 * Created September 19 2019
 * Description: This program is designed to stress the i.MX6 Quad memory bus
 * and controller as much as possible by purposely generating cache misses.
 */
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>
#include <string.h>
#include <sys/time.h>

// L1 is 8-way with 64 lines/way
#define LINE_SIZE 64 // 8 64-bit words per line in L1, L2, and L3; 64 bytes
#define L3_SIZE 16*16384*64 // 16 ways, 16384 lines/way, 64 bytes/line

// Each piece of data can go in one of 16 ways
// Which bits do what for the L3?
// [ ...  |  way  |  line  |  byte  ]
//          23   20 19     6 5      0

// Get the current time in miliseconds
uint64_t get_ms();

int main(int argc, char** argv) {
```

```c
/**
 * Copyright 2022 Joshua Bakita
 * Created September 19 2019
 * Description: This program is designed to stress the i.MX6 Quad memory bus
 * and controller as much as possible by purposely generating cache misses.
 */
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>
#include <string.h>
#include <sys/time.h>

// L1 is 8-way with 64 lines/way
#define LINE_SIZE 64 // 8 64-bit words per line in L1, L2, and L3; 64 bytes
#define L3_SIZE 16*16384*64 // 16 ways, 16384 lines/way, 64 bytes/line

// Each piece of data can go in one of 16 ways
// Which bits do what for the L3?
// [ ...  |  way  |  line  |  byte  ]
//         23   20 19     6 5      0

// Get the current time in miliseconds
uint64_t get_ms();

int main(int argc, char** argv) {
```

```c
// Get the current time in miliseconds
uint64_t get_ms();

int main(int argc, char** argv) {
    if (argc >= 2 && (strcmp(argv[1], "--help") == 0 || strcmp(argv[1], "-h") == 0)) {
        fprintf(stderr,
                "Usage: %s [s(equential)/r(andom)] [number of iterations]\n",
                argv[0]);
        fprintf(stdout,
                "Program will iterate forever if the number of iterations is not specified.\n")
        return 1;
    }

    int is_seq = argc >= 2 ? argv[1][0] != 'r' : 1;

    unsigned long iterations = 0;
    if (argc >= 3) {
        char* status = {'\0'};
        iterations = strtol(argv[2], &status, 10);
        if (iterations == LONG_MIN || iterations == LONG_MAX || status[0] != '\0') {
            perror("Invalid iteration count");
            return 1;
        }
    } else {
        fprintf(stdout, "Infinitely generating %s memory bus traffic...\n",
```

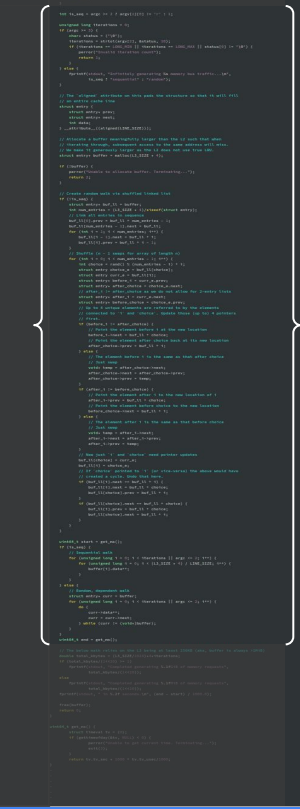


```c
// Get the current time in miliseconds
uint64_t get_ms();

int main(int argc, char** argv) {
    if (argc >= 2 && (strcmp(argv[1], "--help") == 0 || strcmp(argv[1], "-h") == 0)) {
        fprintf(stderr,
                "Usage: %s [s(equential)/r(andom)] [number of iterations]\n",
                argv[0]);
        fprintf(stdout,
                "Program will ite                    mber of iterations is not specified.\n")
        return 1;
    }

    int is_seq = argc >= 2 ? argv[1][0] != 'r' : 1;

    unsigned long iterations = 0;
    if (argc >= 3) {
        char* status = {'\0'};
        iterations = strtol(argv[2], &status, 10);
        if (iterations == LONG_MIN || iterations == LONG_MAX || status[0] != '\0') {
            perror("Invalid iteration count");
            return 1;
        }
    } else {
        fprintf(stdout, "Infinitely generating %s memory bus traffic...\n",
```

More on this next week

Program logic in pseudocode:

- Allocate a large memory block
- Initialize a randomly shuffled linked list
- Depending on command line options, either:
  - Repeatedly traverse the linked list OR
  - Repeatedly traverse the memory block sequentially



```
    }
  }
  uint64_t end = get_ms();

  // The below math relies on the L3 being at least 256KB (aka, buffer
  double total_kbytes = (L3_SIZE/1024)*4*iterations;
  if (total_kbytes/(1<<20) >= 1)


  else


  fprintf(stdout, " in %.2f seconds.\n", (end - start) / 1000.0);

  free(buffer);
  return 0;
}

uint64_t get_ms() {
      struct timeval tv = {0};
      if (gettimeofday(&tv, NULL) < 0) {
            perror("Unable to get current time. Terminating...");
            exit(3);
      }
      return tv.tv_sec * 1000 + tv.tv_usec/1000;
}
```

Think, pair, share: What is this `if` statement checking?

To see the code in context:

https://github.com/JoshuaJB/imx6q-thrasher/blob/amd-3950x-thrasher/thrasher.c

```
        }
    }
    uint64_t end = get_ms();

    // The below math relies on the L3 being at least 256KB (aka, buffer is always >1MiB)
    double total_kbytes = (L3_SIZE/1024)*4*iterations;
    if (total_kbytes/(1<<20) >= 1)
        fprintf(stdout, "Completed generating %.1fGiB of memory requests",
                total_kbytes/(1<<20));
    else
        fprintf(stdout, "Completed generating %.1fMiB of memory requests",
                total_kbytes/(1<<10));
    fprintf(stdout, " in %.2f seconds.\n", (end - start) / 1000.0);

    free(buffer);
    return 0;
}

uint64_t get_ms() {
        struct timeval tv = {0};
        if (gettimeofday(&tv, NULL) < 0) {
                perror("Unable to get current time. Terminating...");
                exit(3);
        }
        return tv.tv_sec * 1000 + tv.tv_usec/1000;
}
```

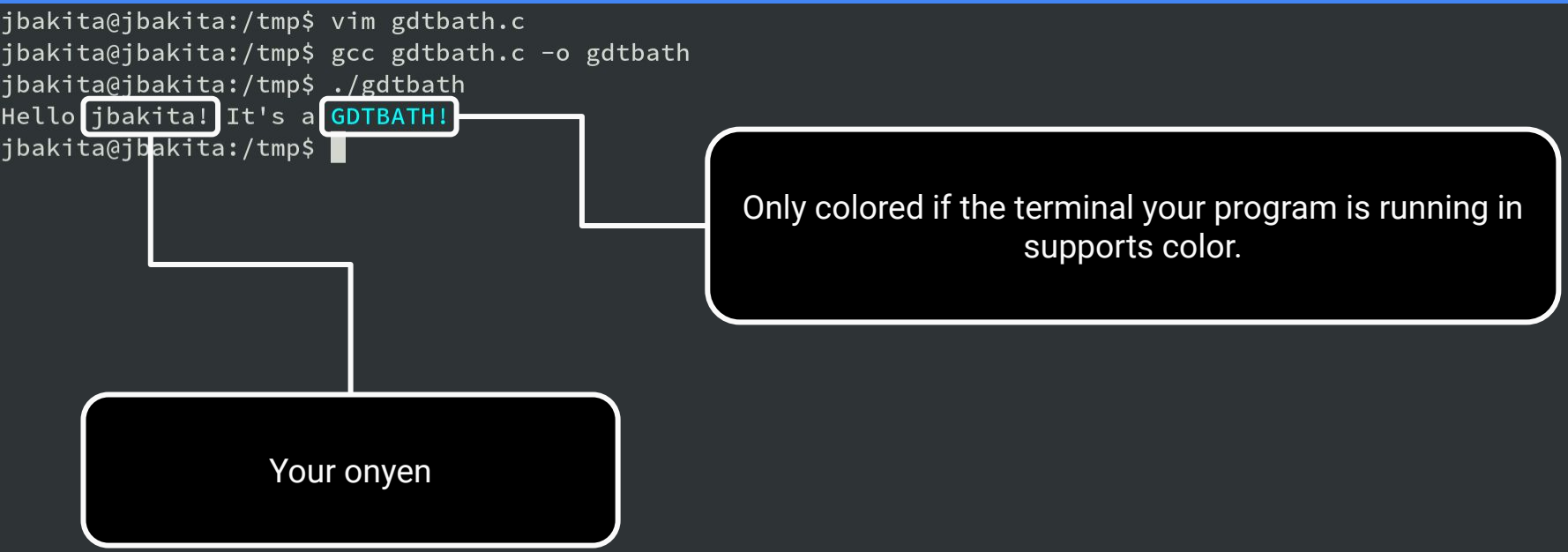
```
        }
    }
    uint64_t end = get_ms();

    // The below math relies on the L3 being at least 256KB (aka, buffer is always >1MiB)
    double total_kbytes = (L3_SIZE/1024)*4*iterations;
    if (total_kbytes/(1<<20) >= 1)
        fprintf(stdout, "Completed generating %.1fGiB of memory requests",
                total_kbytes/(1<<20));
    else
        fprintf(stdout, "Completed generating %.1fMiB of memory requests",
                total_kbytes/(1<<10));
    fprintf(stdout, " in %.2f seconds.\n", (end - start) / 1000.0);

    free(buffer);
    return 0;
}

uint64_t get_ms() {
        struct timeval tv = {0};
        if (gettimeofday(&tv, NULL) < 0) {
                perror("Unable to get current time. Terminating...");
                exit(3);
        }
        return tv.tv_sec * 1000 + tv.tv_usec/1000;
}
```

Assignment 1 Overview

It's a good day to be a Tar Heel!

Assignment 1 Overview

What your program will do



Assignment 1 Overview

What you'll need

- ssh your_onyen@comp211-2sp23.cs.unc.edu
- vim (or nano)
- gcc gdtbath.c -o gdtbath
- man
- Some C functions:
  - printf
  - getlogin
  - …

Assignment write-up will provide more detail

Thanks!
Questions?

Come chat with me now, or drop by my office Monday morning! (Sitterson 311, 8:30-11:30 AM)

LA office hours coming soon…

Assignment 1 up tonight.

Contact:
Email: hacker@unc.edu
Twitter: @JJBakita
Web: https://cs.unc.edu/~jbakita

Old Well, University of North Carolina at Chapel Hill, Winter 2017