

# 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



DEPARTMENT OF COMPUTER SCIENCE  
LEVEL 1

SITTERSON HALL  
(SN ROOM PREFIX)



# C Versus Java

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

# Anatomy of a Single-File C Program

From my research: thrasher.c

```

0..
+ Copyright 2022 Joshua Sakita
+ Created September 19 2019
+ Description: This program is designed to stress the L3R0 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 10*1024*64 // 10 ways, 1024 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  10 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 [x(sequential)/r(random)] [number of iterations]\n",
            argv[0]);
        fprintf(stdout,
            "Program will iterate forever if the number of iterations is not specified.\n");
        return 1;
    }
}

```



```

1 # Copyright 2016 Amazon.com, Inc. or its affiliates. All rights reserved.
2 # Licensed under the Apache License, Version 2.0 (the "License");
3 # you may not use this file except in compliance with the License.
4 # You may obtain a copy of the License at
5 # http://aws.amazon.com/apache2.0/
6 # or http://aws.amazon.com/free
7 # In the AWS CLI "help" text displayed at running "aws help", these lines
8 # and this comment are replaced with the AWS CLI CLI "help" text.
9
10 # This program is designed to stress the 1 MB per memory hour
11 # and throughput as much as possible by repeatedly generating random strings.
12
13 # Usage:
14 #   awscli stress.py [options]
15 #
16 # Options:
17 #   -c, --count COUNT          Number of iterations to run
18 #   -s, --size SIZE            Size of the random string to generate
19 #   -t, --threads THREADS      Number of threads to run
20 #   -h, --help                  Display this help message
21
22 import sys
23 import random
24 import string
25 import threading
26
27 def main():
28     # Parse the command line arguments
29     count = 1000000
30     size = 1024
31     threads = 1
32
33     # Parse the command line arguments
34     for i in range(1, len(sys.argv)):
35         if sys.argv[i].startswith('-'):
36             option = sys.argv[i].replace('-', '')
37             value = sys.argv[i+1]
38             if option == 'c':
39                 count = int(value)
40             elif option == 's':
41                 size = int(value)
42             elif option == 't':
43                 threads = int(value)
44             else:
45                 print 'Unknown option: %s' % option
46                 sys.exit(1)
47
48     # Run the stress test
49     for i in range(1, threads+1):
50         t = threading.Thread(target=stress_test, args=(count, size))
51         t.start()
52
53     # Wait for all threads to finish
54     for i in range(1, threads+1):
55         t.join()
56
57     print 'Stress test complete'
58
59 def stress_test(count, size):
60     for i in range(1, count+1):
61         # Generate a random string
62         s = ''.join(random.choice(string.ascii_uppercase + string.digits)
63                     for _ in range(size))
64         # Write the string to stdout
65         sys.stdout.write(s + '\n')
66         # Flush the stdout buffer
67         sys.stdout.flush()
68
69 if __name__ == '__main__':
70     main()

```

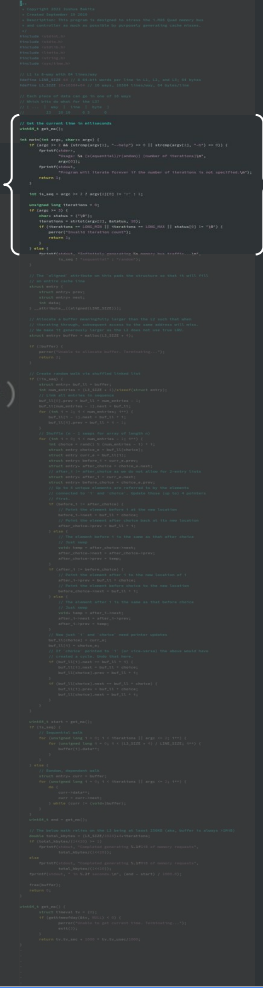
7

```
// 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(quential)/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 = strtoul(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",
            is_seq ? "sequential" : "random");
    }
}
```





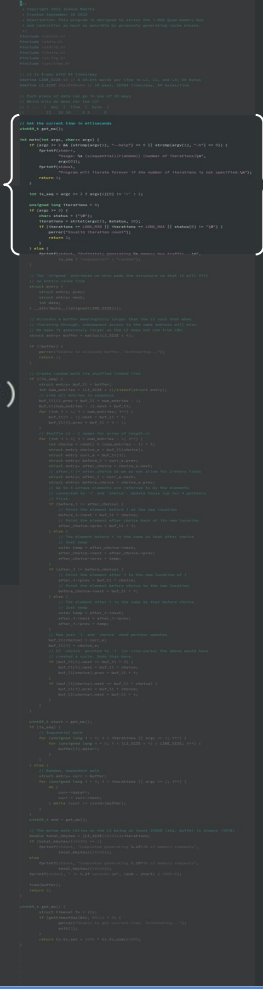
```
// Get the current time in milliseconds
uint64_t get_ms();

int main(int argc, char** argv) {
    if (argc >= 2 && (strcmp(argv[1], "--help") == 0 || strcmp(argv[1], "-h") == 0)) {
        fprintf(stderr,
            "Usage: %s [s(quential)/r(andom)] [number of iterations]\n",
            argv[0]);
        fprintf(stdout,
            "Program will iterate [number of iterations] times\n");
        return 1;
    }

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

    unsigned long iterations = 0;
    if (argc >= 3) {
        char* status = {'\0'};
        iterations = strtoul(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",
            is_seq ? "sequential" : "random");
    }
}
```

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

```
/*
 * This program is a simple implementation of a linked list.
 * It uses a single file to store the data and a single file to store the index.
 * The program is designed to be run on a Unix-like system.
 * It uses the following command line options:
 * -h: Display this help message.
 * -s: Specify the size of the memory block in bytes.
 * -l: Specify the number of elements in the linked list.
 * -t: Specify the type of traversal to perform.
 * -v: Display the version number.
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <time.h>
#include <math.h>

#define MAX_SIZE 1024 * 1024
#define MAX_ELEMENTS 1000000
#define MAX_DEPTH 1000000

typedef struct node {
    int data;
    struct node *next;
} node_t;

node_t *head = NULL;
node_t *tail = NULL;
int count = 0;

/*
 * Function to allocate a large memory block.
 */
void allocate_block(int size) {
    char *buffer = malloc(size);
    if (buffer == NULL) {
        perror("malloc");
        exit(1);
    }
    memset(buffer, 0, size);
}

/*
 * Function to initialize a randomly shuffled linked list.
 */
void init_list(int elements) {
    srand(time(NULL));
    for (int i = 0; i < elements; i++) {
        node_t *new_node = malloc(sizeof(node_t));
        new_node->data = rand() % MAX_ELEMENTS;
        new_node->next = NULL;
        if (head == NULL) {
            head = new_node;
            tail = new_node;
        } else {
            tail->next = new_node;
            tail = new_node;
        }
        count++;
    }
}

/*
 * Function to traverse the linked list.
 */
void traverse_list() {
    node_t *current = head;
    while (current != NULL) {
        printf("%d\n", current->data);
        current = current->next;
    }
}

/*
 * Function to traverse the memory block sequentially.
 */
void traverse_block() {
    char *buffer = malloc(MAX_SIZE);
    if (buffer == NULL) {
        perror("malloc");
        exit(1);
    }
    memset(buffer, 0, MAX_SIZE);
    for (int i = 0; i < MAX_SIZE; i++) {
        printf("%d\n", (int)buffer[i]);
    }
}

int main(int argc, char *argv[]) {
    if (argc < 2) {
        printf("Usage: %s -s size -l elements -t type\n", argv[0]);
        return 1;
    }
    int size = 0;
    int elements = 0;
    int type = 0;
    for (int i = 1; i < argc; i++) {
        if (strcmp(argv[i], "-h") == 0) {
            printf("Usage: %s -s size -l elements -t type\n", argv[0]);
            return 1;
        }
        if (strcmp(argv[i], "-s") == 0) {
            size = atoi(argv[i+1]);
            i++;
        }
        if (strcmp(argv[i], "-l") == 0) {
            elements = atoi(argv[i+1]);
            i++;
        }
        if (strcmp(argv[i], "-t") == 0) {
            type = atoi(argv[i+1]);
            i++;
        }
        if (strcmp(argv[i], "-v") == 0) {
            printf("Version 1.0\n");
            return 0;
        }
    }
    allocate_block(size);
    init_list(elements);
    if (type == 0) {
        traverse_list();
    } else {
        traverse_block();
    }
    return 0;
}
```





```
}
}

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 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

```
jbakita@jbakita:/tmp$ vim gdtbath.c
jbakita@jbakita:/tmp$ gcc gdtbath.c -o gdtbath
jbakita@jbakita:/tmp$ ./gdtbath
Hello jbakita! It's a GDTBATH!
```

Only colored if the terminal your program is running in supports color.

Your onyen

## 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](mailto:hacker@unc.edu)

Twitter: [@JJBakita](https://twitter.com/JJBakita)

Web: <https://cs.unc.edu/~jbakita>

