Level 0x05

Intro to C

Topics

- Events
- Hacker History
- C Stuff
  - Compiling
  - Printf
  - Arithmetic / Logic
  - Conditionals

Upcoming Events

- Blue Hens CTF (University of Delaware)
  - October 27, 2023, 3:00pm – October 29, 2023, 3:00pm
  - High school, Undergraduate, Pros
  - https://bluehens.ctfd.io/categories
  - Categories include Minecraft challenges
  - Prizes for 1st-3rd place in each category

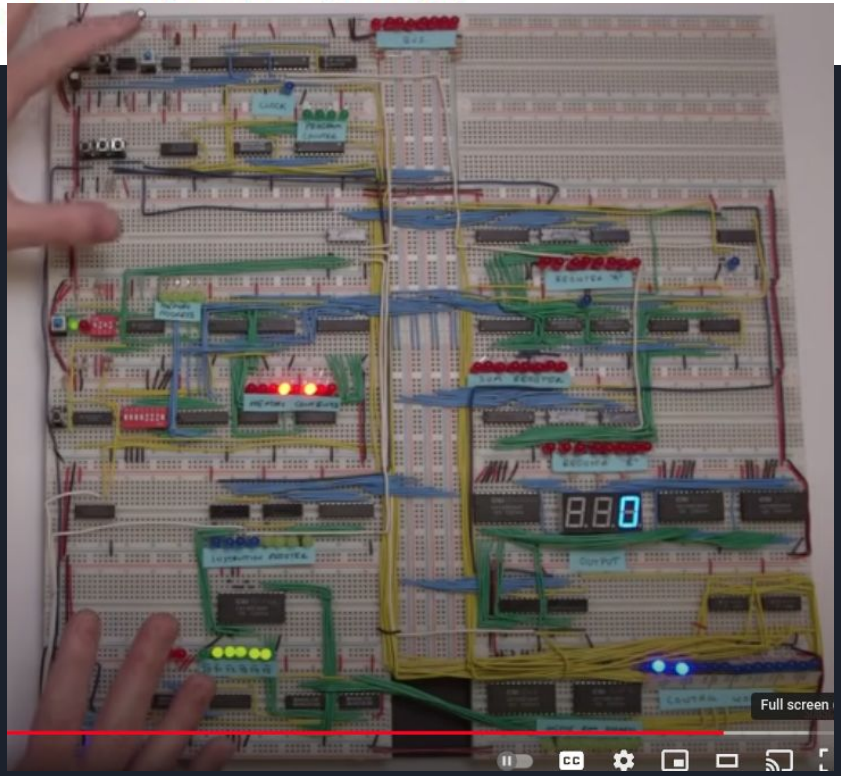
Upcoming Events

- Maker Faire Orlando - “Greatest Show & Tell on Earth”
  - Saturday Nov 4th - Sunday Nov 5th, 2022
  - https://www.makerfaireorlando.com/
  - Orange County Fairgrounds
  - $25 Adults, $20 Students, $5 off if pre-purchased

Ben Eater

@BenEater 1.16M subscribers 118 videos

Subscribe to see tutorial-style videos about electronics, computer architect...



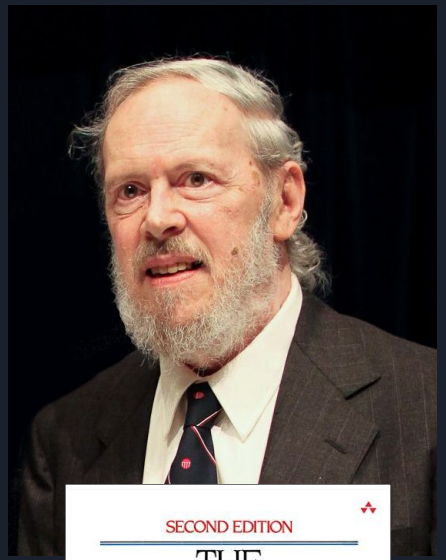
- Youtube tutorials
  - Digital circuits
  - Assembly
  - Networking

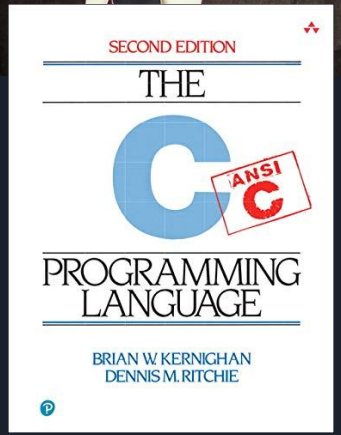
Club Leadership Position

- T-Shirt Design
- Contest coordinator
  - CTF signups
  - Align teams for Code Quest / Cyber Quest
- Comms channel
- Presentations
  - TJHS Cyber Club Presentations

Background of C

- History
  - Developed in 1972 by Dennis Ritchie at Bell Labs
  - Standardized in 1989 (ANSI C)
- Purpose
  - General purpose programming language
  - Compiles down to machine instructions
- Applications:
  - Operating systems
  - Bare-metal / Embedded
  - Performance
  - Not typically used for Web
- Influenced 70+ other languages:
  - C++, Objective-C, Java, PHP, C#, Go, Rust





Hello World

- Comments
  - /* orig C multi-line */
  - // C++ style single line
- Single main function
  - You can access command line args
  - Return code to the calling script
    - 0 = success
- printf = print formatted
- Indentation doesn't matter

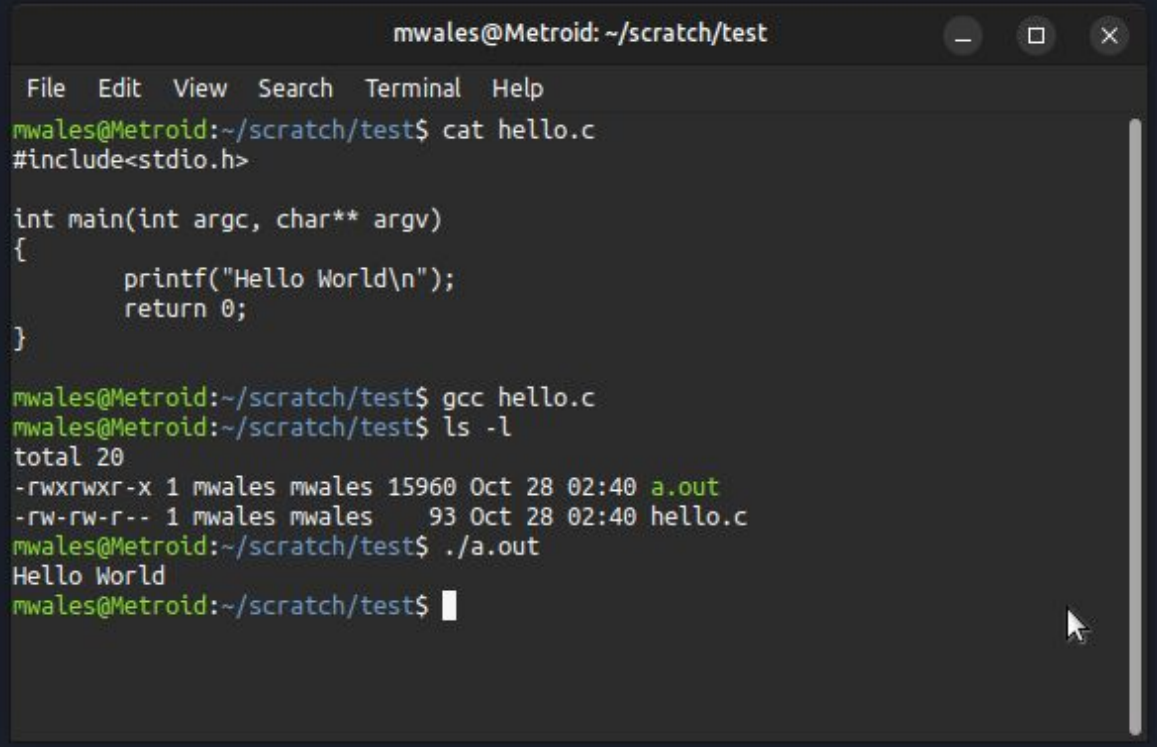
```c
#include <stdio.h>

/*
  Multi-line comments!

  argc is number of args to program
  argv is list of the args
*/

int main(int argc, char** argv)
{
   // Every C program has a main function
   printf("Hello World\n");
   return 0;
}
```

Compiling C applications



Variables / Integer Types

- Character type + Integer Type
  - char
- Integer types
  - short
  - ***int***
  - long
  - long long
- Floating point type
  - float
  - ***double***
  - long double
- Size Type
  - size_t
- Integer types can also be unsigned
- Arrays / lists

```
// old standard declaration
char smallNum, otherNum;
smallNum = 10;
otherNum = 42;

// typical declaration + assign
int numPoints = 10000;
unsigned long altitude = 5000;

// OK to reassign later
smallNum = 0xb7;  // hex

// doubles are more precise
float pi = 3.14159;
double r = 45.0;

double area = pi * r * r;

int listOfFourNumbers[4];
```

#include<stdint.h> (C99)

stdint.h type	Num Bytes	Minimum	Maximum
int8_t	1	-128	127
uint8_t	1	0	255
int16_t	2	-32,768	32,767
uint16_t	2	0	65,535
int32_t	4	-2,147,483,648	2,147,483,647
uint32_t	4	0	4,294,967,295
int64_t	8		
uint64_t	8		

Output

- Some terminology
  - A character (or char).
    - Single letter or number.
    - Single quote. ‘A’
  - A string.
    - Series of characters or numbers.
    - Double quotes. “Abcde”
    - Ends with a null character (0x00)
- A few different ways to output from standard C library stdio.h
  - `printf(“text and a number %d. ”, 42);  // By far the most common`
  - `puts(“just a single string”);  // Built-in newline`
  - `putchar(‘a’);`

Printf formatting

- Escape sequences
  - `\n` is a newline
  - `\t` is a tab
- Conversions / extra args
  - `%d` or `%i` for signed integer, `%u` for unsigned integer
  - `%x` or `%X` for hexadecimal numbers
  - `%c` for single characters
  - `%s` for strings
  - `%f` for floating point
- Flags, width, and precision
  - `printf("Name: %10s  Weight%3.1f\n", name, weight);`
  - `printf("32 bit num in hex is 0x%08x\n", 0xbeef);`
- Not just C uses it...
  - Python: `"format str %s = %d" % ("age", 12)`
  - Java: `System.out.println(format, args)`

Arithmetic / Logic

- Basic operations: add (+), subtract (-), multiply (*), divide (/)
- Modulus operations: % (division remainder of)
- Assignment and operations can be combined
  - Operations followed by =
  - `varX *= 2;  // equivalent to varX = varX * 2`
- Increment (++) and Decrement (--)
  - Prefix form: `int xVal = ++i;` // i is incremented, xVal is then set to value of i
  - Postfix form: `int xVal = i++;` // xVal is set to the value of i, then i is incremented
- Logic
  - Equality: `var1 == var2` or `var1 != var2`
  - Comparison: <, <=, >=, >
  - Logical And is `&&`, Logical Or is `||`
  - String comparison: `strcmp(string1, string2)` returns negative, zero, or positive int

Main Function Arguments

- Main function: `int main(int argc, char* argv[])`
  - `argc`: number of arguments passed to your program
  - `argv`: List of argument strings
- `argv[0]` is the name of the program
- Convert strings to int / float using stdlib.h
  - `int firstNumArg = atoi(argv[1]);`
  - `double secondFloatArg = atof(argv[2]);`

Conditionals

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main(int argc, char* argv[]) {
        if(argc < 4) {
                printf("Usage: %s num1 operation num2\n", argv[0]);
                return -1;
        }
        int num1 = atoi(argv[1]);
        int num2 = atoi(argv[3]);
        if (strcmp(argv[2], "+") == 0) {
                printf("%d plus %d = %d\n", num1, num2, num1 + num2);
        }
        else if(strcmp(argv[2], "-") == 0) {
                printf("%d minus %d = %d\n", num1, num2, num1 - num2);
        }
        else {
                printf("%s is invalid operations\n", argv[2]);
        }
        return 0;
}
```

Running

```
$ gcc test.c -o calc
$ ./calc 23 + 55
23 plus 55 = 78
$ ./calc 99 - 44
99 minus 44 = 55
$ ./calc 4 ^ 6
^ is invalid operations
$ ./calc
Usage: ./calc num1 operation num2
```

Links

- Many graphics from Wikipedia / wikimedia.org (Creative Commons License)
  - https://en.wikipedia.org/wiki/Dennis_Ritchie#/media/File:Dennis_Ritchie_2011.jpg
- Amazon.com - K&R The C Programming Language
  - https://www.amazon.com/Programming-Language-2nd-Brian-Kernighan/dp/0131103628
-