OS 1/30

Reagan Shirk

January 30, 2020

Code

- Lecture is going to be a lot of live coding today, I'll try to get what I can in my notes
- In Linux, hidden files are shown with a .

```
#include <stdio.h>
#include "string.h"

#define PI 3.14159265358979
#define TRUE 1
#define FALSE 0
#define myname "Reagan Shirk"
#define HALFOF(x) x/2
#define TRACE { printf("Executing %s line %d\n"), __FILE__, __LINE__); }

// void main() is still allowed but unconventional
int main(int argc, char** argv)
{
    // other types are char, long, short
    long int val = 6553000000000;
    unsigned val2 = -65530;
    printf("My val: %ld", val);
    printf("My val: %d", val2);

    printf("The size of 'char' is: %ld", sizeof(char));
    printf("The size of 'int' is: %ld", sizeof(int));
    printf("The size of 'long' is: %ld", sizeof(long));
    printf("The size of 'short' is: %ld", sizeof(short));

    printf("We have %d command line arguments\n", argc);

    if (argc > 1)
    {
        short i = 1;
        for (i; i < argc; ++i)
        {
            printf("%d, %s\n", i, argv[i]);
        }
    } // command line arguments are typed in along with the executable file (i.e. it'd be ./a.out <comma

    // Time for Pointers
    int box = 7;
    int* boxpointer = &box; // returns boxes location in memory

    printf("box value = %d\n", box); // should print out the value
    printf("box address = %p\n", &box); // location in memory
    printf("box pointer = %p\n", boxpointer); // location in memory
```

```
    printf("box pointer value = %d\n", *boxpointer); // should print the value

    // Describing (deciphering) variables/expressions
    // * = 'pointer to'
    // [] = 'array of'
    // () = 'function returning'

    // Step 1: Find intendifier
    // Step 2: Look for symbols on the right until you hit the end
    // Step 3: Look to the left

    // int *p[];
    // - p
    // - is an array of
    // - a pointer to
    // - integer
    // So we say: "p is an array of a pointer to an integer"

    return 0;
}
```