OS 2/11

Reagan Shirk

February 11, 2020

C stuff

- `int main()` is the entry point to a program
- When we say program, process, thread, we mean: set of code + data
  - A set of instructions and the data associated with the set of instructions
- We have the process control block, also known as the PCB
  - It has an identifier, some state, a program counter, memory pointers, context, and more
- We wrote a lot of code today that I'll try to remember to transfer to my notes
  - Although I definitely missed something that Grant did because my output wasn't correct

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

#define PI 3.1415926535
#define TRUE 1
#define FALSE 0
#define myname "Christan Grant"
#define HALFOF(x) x/2

#define TRACE { printf(Executing %s line %d\n"", __FILE__, __LINE__); }


struct roster {
    short number;
    char team[60];
    char *name;
};
typedef struct roster roster;
//typedef short teamtype
// pgrma pack

/**
 * Will print the roster
 */
int printroster(roster myroster) {
    printf("<%x,%s: %s>\n", myroster.number,
                myroster.team,
                myroster.name);
    return 0;
}

// void main ()
int main (int argc, char** argv) {
    // char
    // short
```

```
// long

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

int val = 5;
unsigned val2 = 65530.5;
printf("My val %d\n", val);
printf("My val2 %d\n", val2);

printf("We have %d command line arguments.\n", argc);
if (argc > 1) {
    // We have extra cmd line arguments
    short i = 0;
    for (i = 0; i < argc; ++i) {
        printf("%d: %s\n", i, argv[i]);
    }
}


// Pointers
int box; //Make space to hold 4 bytes [ ]
box = 7;

int *boxpointer = &box;

printf("box value = %d\n", box); // 7
printf("box address = %p\n", &box); // some address?
printf("box pointer = %p\n", boxpointer);
printf("box pointer value = %d\n", *boxpointer); // 7
printf("\n");

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

// Step 1: find identifier
// Step 2: look for symbols on the right
// Step 3: Look to the left
//
// int *p[];
// - p
// - is an array of
// - a pointer to
// - integer


printf("Fifteen value = %x\n", 15); // 7

// int* var; -- int *var;
```

```
    //int* myvar, var; // <-- confusing

    struct roster r;

    printf("Size of basic roster: %ld\n", sizeof(struct roster));

    //r.name = "Kobe";
    //r.team = "Lakers";
    r.number = 24;

    r.name = malloc(20 * sizeof(char));
    strncpy(r.name, "Kobe", 20-1);
    strncpy(r.team, "Lakers", 20-1);

    printroster(r);

    free(r.name);

    unsigned int flag = 0; // 2^(4 bytes * 8 bits) 2^32

    // 0x4 & 0x1
    // 0x100 | 0x001 -> 0x101
    u_int8_t myu8; //

    flag |= 0b0001 ; // flag = flag | 0b0001;

    return 0;
}
```