CMPSC 311

Scribe Notes by: Bashyam Vishnu; Bashyam Matthew; Beidler Michael; Berezanich Michael

January 15th, 2016

<u>General Unix Information:</u>

-Unix systems may be accessed via an SSH(Secure Shell) client (such as PUTTY) or a VPN (Virtual Private Network)
-An SSH allows for secure login to a remote network

<u>General C Information:</u>

-Declaring main is redundant, it has no need for a prototype
-Prototypes are also not technically needed if the function definition is in the same file
-When compiling a C program in a shell, if -o (name_of_program) is omitted, the default name is a.out

<u>Sample Code:</u>

Variables

```
#include <stdio.h>

int add (int a, int b){
        printf ("a=%d b=%d\n", a, b);
        return a+b;
}

Int main () {
        printf ("ret=%d\n", add(10, 20));
        return 0;
}
```

<u>Tips/Notes</u>

This simple code shows the declaration of a function to add two variables. Integers “a” and “b” are declared and the function returns the sum of those two variables. In this case, the function would print out 30.
-printf statements can be used to print variables with a place holder (%d, %f, etc.)
-%d tells the computer to interpret as an integer while printing

Recursion

```
#include <stdio.h>

void rec (int a) {
	printf "in%d\n", a);
	if (a > 0)
		rec (a-1);
	printf ("out%d\n, a );
}

int main ()
	rec(2);
	return 0;
}
```

Tips/Notes

-When using recursion, a base case is required to make sure the recursion stops at some point and doesn't produce an infinite loop (In the case above, this is the "if (a > 0)" condition, which will stop the recursion once the recursion function is called with 0)

Command Line Arguments

```
#include <stdio.h>

int main (int argc, char **argv) {
	int n, m;
	n = atoi (argv[1]);
	m = atoi (argv[2]);
	printf ("Argument 1: %d\nArgument 2: %d\n", n, m);
	return 0;
}
```

Tips/Notes

-When your main function accepts arguments, if the wrong number of arguments are supplied, a segmentation fault occurs

Pointers:

- A pointer is just a variable that "points to" another variable or constant, i.e., holds its virtual address
    - Every instruction and data item in your program resides at a virtual address
- Two main operations:
    - *dereference: gets the value at the memory location stored in a pointer
    - &address of: gets the address of a variable
    - Int *my_ptr = &my_var;