CSCI 2500: Computer Organization

Lab 5 – Exercises C Programming

1. Practice with Pointers: Download lab5_ex1.c from LMS; Labs → Lab 5 (9/30/2015)). Finish the C program by defining a pointer to char, a pointer to int, a pointer to float, and a pointer to double. Then use the `calloc` to locate 4 members (elements) for each pointer. Display to standard output the address pointed by the pointer and the address of the pointer incremented by 1. Do this for each pointer type. Examine the distance (i.e. number of bytes) between the two addresses pointed by the pointer and its increment for each of the different data types.

2. Little Endian vs. Big Endian: Download lab5_ex2.c from LMS. The purpose of this C program is to determine if your machine uses little endian or big endian byte order. Use the union structure contained in the program to find the byte order display the result to standard output. Note: You only need to display the `char` array data and then write the `if` statement inside of the `isLittleEndian()` function that will check the "endianess" of the machine and then return the correct value to the `main` function.

3. C Preprocessor and Macros: Download lab5_ex3.c from LMS. You are to implement a macro that computes the area of a circle. Please call your macro `CIRCLE_AREA`. An example function called `circleArea` is provided in the code already. Your macro must only take one input parameter (the radius of a circle). Once completed experiment with conditional compilation by setting and unsetting the `CIRCLE_MACRO` and `CIRCLE_FUNCTION` define directives.

   Other things to try:

   Try showing the output of running the preprocessor only on your completed lab5_ex3.c file (use the `cpp` command line program).

   Remove the macro definitions for `CIRCLE_MACRO` and `CIRCLE_FUNCTION` (lines 3 and 4). What is the effect on your program?

   Now try compiling your program using the additional command line options `-DCIRCLE_MACRO` and `-DCIRCLE_FUNCTION.` What effect do these options have on your program results?

4. Linking Nightmare: As we discussed in class, the linker handles symbol resolution between multiple object files. Symbol resolution for global variables that are not `static` (symbols) are tricky because the linker must somehow choose one of the definitions and discard the rest. Download modules lab5_ex4a.c and lab5_ex4b.c and compile them into a program (Note: you will get a warning from the linker). Run the program and note the results. Print out the addresses of variables `x` and `y` (Note: do this in the both c source files). What is happening? Fix the program so that `x` and `y` print correctly.

5. The Main Issue Is? Download modules lab5_ex5a.c and lab5_ex5b.c and compile them into a program. Run the program. The program prints a value even though function `cfunc` never initializes the variable `main`. Explain what the linker is doing.