MP4 Base Conversions in C

Due Wednesday, September 24, 2014 at 10:00 p.m.

Overview

This week you'll be creating a simple program that converts the base of a user input integer. The user enters an integer and a conversion operation, and the converted number is printed to the console. The `convert` function takes two integer values, a value to be converted and a key that indicates the conversion operation, executes the conversion and prints the value to the console. You will implement this function.

Although the programming task itself will be quite easy, this MP will help introduce the new environment, language, and tools. This MP should be done individually.

The Pieces

The distribution includes only a few files. The file that concerns you is the main.c file which contains the program's `main` function and `convert` function. This is where your code will go.

There are a couple other files that may interest you:

- Makefile - This is a file that's used with the Linux make utility. The purpose of using a makefile is that it's easy to describe file dependencies, compile and link code, and recompile only necessary files when rebuilding large projects. Take a look at it to see how this MP is built.
- printBinary.h - Header file for the printBinary function.
- printBinary.o - Object file (machine code) for the printBinary function.

Details

The function you write is at the bottom of main.c:

```
void convert(int value, int key);
```

The key argument specifies the conversion operation performed on the value argument. You need to implement four conversions for this assignment using a switch statement:

0 - Binary
1 - Octal
2 - Decimal
3 - Hex

You must use this key to operation mapping (i.e. 0 performs a binary conversion, 1 an octal, and so on). Each operation should only print the converted value and a newline character. If the user enters a key that isn't one of these you must print "Invalid option\n".
We provide you with a function that prints the binary conversion:

```
void printBinary(int value, int size);
```
The code for this function isn't provided (printBinary.o contains the binary for this function) due to the the challenge, but see the header file printBinary.h for the function definition.

HINT: Look up printf format specifiers. Many of these base conversions can be done simply by formatting within a printf statement.

Challenges for Program 4

If you're feeling adventurous you can do the challenge for this program.

(5 points) Write your own printBinary function and use it in your program. Do this by creating your own printBinary.c file and writing the function in there. Then uncomment the lines in the Makefile to build your code and link it with the rest of the program. You must also explain how your function works in the introductory paragraph. NOTE: Building your own printBinary will overwrite the printBinary.o we give you.

Specifics

- Your code must be written in C and and must be contained in the main.c file provided to you - we will NOT grade files with any other name.
- You must implement the four base conversions and print the error message when an invalid operation is input.
- Your code must be well-commented. You may use either C-style (/* can span multiple lines */) or C++-style (// comment to end of line) comments, as you prefer.

Building and Testing

To build this MP we have provided you with a makefile. To build the program you have to run the following command in the MP4 directory:

```
make
```

To clean the directory of all unnecessary object files and the mp4 binary run:

```
make clean
```

We have not provided any test files for this MP, however testing should be fairly straight forward. Once your program is functional test your program with different kinds of inputs to ensure it works. You can also run the code with the gdb debugger by running:

```
gdb mp4
```

Grading Rubric

Functionality (70%)

15% - (each conversion) four required conversions work correctly

10% - Error is printed when user enters an invalid conversion

Style (10%)

10% - uses a switch statement for control flow in convert function

Comments, clarity, and write-up (20%)

5% - introductory paragraph explaining what you did (even if it's just the required work)

15% - code is clear and well-commented

Note that some point categories in the rubric may depend on other categories. If your code does not compile, you may receive a score close to 0 points.