

Project 1: Binary and Hexadecimal Conversion

Corrections and Additions

1. The constant that defines the length of the hexadecimal string is 5 in the template. This should have been 9. Feel free to fix this error or leave it as is. Students have already submitted the project so we will just be testing with smaller numbers so that my mistake will not impact your results.

Files

1. [Number_Translation_Template.c](#) ⬇ if you get an unauthorized error when trying to download the template use the file tab on the left toolbar and download it from there.

Learning Goals

1. Learn some basic Linux commands.
2. Learn to transfer files between computers.
3. Start using a terminal based text editor of your choice.
4. Become familiar with the build process for a C program.
5. Write a simple C program.
6. Learn how to submit your assignments.

Logistics

1. All work for this assignment is to be done on one of the UW CS department's instructional Unix/Linux machines.
2. All assignments in this course will be graded only on CS departmental machines (e.g., royal-01.cs.wisc.edu) running Linux Operating System. It is your responsibility to make sure that your code runs on these machines correctly.

General Advice

1. If you are not using the CS departmental machines with Linux Operating System and would like to code using your laptop then:
2. Please DON'T use an Integrated Development Environment (IDE) for learning to code in C. It'll hide many of the low-level details from you. You can become a good C programmer only if you understand all these low-level details.
3. Avoid using Windows Operating System for writing C code since the compiler that we'll use for grading your assignments is gcc (GNU C Compiler) which is a Unix/Linux based C compiler.
4. If you still insist on using a computer which already has a Windows OS, then you should have a virtual machine (VMware or Virtualbox) to run a Linux OS. You'll also need a compiler like MinGW or Cygwin (which contains tools like gcc) installed on your Windows machine. To be honest connecting to the lab Linux machines is easier than getting those software packages installed correctly.

Questions

1. If you have a question, please consider these options in order.
2. Post your question on Piazza and make it public for general questions. Feel free to post problem solving strategies, but do not post code publicly.
3. If you need to post code, then make the post private. We will in general not help debug your code for syntax errors. We will however explain cryptic compiler error messages. Please post both the code segment generating the error and a screenshot of the cryptic error message.
4. Email your TA. For project 1, If you last name begins with A-L contact Yien at yien@cs.wisc.edu. If you last name begins with M-Z contact Hannah at hstrohm@wisc.edu. Mike at mdoescher@wisc.edu.

Debugging advice

1. Write your code in very small sections at a time and verify that it compiles before continuing.
2. Fix the compiler error messages from the top down. The first error may cause other compiler error messages to generate from correct code later in your file. So, scroll up and address the top error message first.
3. Ask questions about cryptic error messages. Post a screenshot of the error message on Piazza as a public post if you do not need to share any code. Post privately if you do need to share your code.

Part 1: Logging into a CSL Machine

The first challenge is simple: being able to use one of the Linux machines made available by the CSL (Computer Systems Lab) here at Wisconsin. The lab provides several different Linux systems for you to use. We will be grading all projects on the CSL machines. Note that different operating systems will produce different output and different versions of the compiler (or selecting different compiler options) may produce different output.

To log in, you must know your CS login which the Lab should have mailed to you. Contact them at lab@cs.wisc.edu if you don't have one yet. If you are new to Wisconsin or your account has expired, please [activate your account](#) .

Once you know what your login is, you must pick a machine and use SSL to log in. The set of machines available can be found at [Instructional Linux Computers](#) .

If you are using a Mac or Linux machine, you can log into one of these machines via [ssh](#) by typing, at a command shell prompt, something like this:

```
prompt> ssh username@emperor-03.cs.wisc.edu
```

My username is **doescher**, so I would type:

```
prompt> ssh doescher@emperor-03.cs.wisc.edu
```

Note that we use prompt> to indicate what the shell shows you when asking for you to type a command; undoubtedly, it will say something different than prompt> and in fact the prompt is customizable.

It may seem kind of arbitrary to pick a particular machine, instead of (say) an unused or lightly used machine. To have the CSL pick a machine for you, log into any one of the following: **best-linux.cs.wisc.edu**. In other words, just replace **emperor-03.cs.wisc.edu** in the ssh line above with **best-emperor.cs.wisc.edu** instead.

If you are using a Windows machine, you'll probably need to use [Putty](#) or another ssh client to log into the CSL machines. See [Connecting to Linux computers via SSH](#) for more options.

Part 2: The Shell

Now, you need to start getting familiar with the command shell you are using. Most of you are probably using **bash**. However, you are free to use any shell you like, such as **tcsh**, **zsh**, **ksh**, etc. The important thing: pick one, read about it, and learn how to use it. There are many online tutorials for shells, and the best way to learn is to just spend some time practicing.

One thing you'll need to know is how to move around the file system hierarchy. A Unix file system is just what CS types call a **tree**, which has a single root and many children. At the root is a directory (or folder) called the **root directory**. In Unix, we refer to the root as / (a forward slash). All other files and directories (folders) are located somewhere in that tree.

When you log into one of those Linux machines, you'll be placed into what is called your **home directory**. To see what directory that is, type:

```
prompt> pwd
```

The **pwd** command Prints the Working Directory, which should be something like this:

```
/u/d/o/doescher
```

This entity is called a **pathname** and it should be interpreted as follows. In the root directory (/), there is another directory called **u** (I think this is short for users); in that directory, there is another directory called **d** (my username begins with d); in that one, there is another directory called **o** (second letter of my username), etc., on down to the last directory called **doescher**. The symbol / (in addition to being the name of the root directory) is used as a separator in the path; in other words, it separates each directory (or file) name from the others in the path. Note: Windows machines use the name of the drive as the root (i.e. c:\) and the backslash \ as the separator.

Now, let's do some simple shell stuff just to warm up. First, check out the files and directories in your home directory. To do this, you will run a program called **ls** which lists files and directories.

```
prompt> ls
```

You should see some interesting output there, including some directories (like private/ and public/ among others) and perhaps a few files.

Let's go create a directory to do the work for Project 1 in to keep things organized. First, let's move into the private directory. To do this we will run a program called **cd** which changes the active directory and moves one level deeper into the tree

```
prompt> cd private
```

Now try pwd again and verify that your Working Directory ends in private.

```
/u/d/o/username/private
```

Three special directories include .. (2 dots), . (1 dot), and ~ (tilde)

```
cd ..
```

This will take you up one directory in the tree. You should end up back at /u/d/o/username if you do this.

```
cd .
```

The single dot . represents the current directory. This command takes you to the directory you are currently in.

cd ~ takes you back to your home directory. Your starting location when you first logged in.

Let's move back into private and then create a new directory for CS354

```
mkdir CS354
```

The mkdir program will create a new directory called CS354. Note do not use spaces in directory or file names. (Windows will allow this if you enclose the directory name in quotation marks – but in general using spaces even on windows is not a good practice)

Move into this new CS354 directory. Create another new directory for Project1 and move into that directory.

The next command we'll learn is echo. This command simply outputs whatever text you pass as an argument. For example:

```
prompt> echo Hello World
```

just displays the text Hello World on the screen.

Not super useful by itself but when combined with the file redirection operator > can be quite useful. Type the following at the command prompt:

```
prompt> echo Hello World > file.txt
```

The file redirection operator will write the contents of anything that was supposed to be printed on the screen to a file. Note this will destroy the contents of whatever was already in this file and replace them. To append use the >> command instead.

To check if your file exists now, type:

```
prompt> cat file.txt
```

```
Hello World
```

If you did it correctly, you should see the text "Hello World" as shown above.

If you don't want that file around, use the file deleting command rm to remove it:

```
prompt> rm file.txt
```

However, please be VERY careful with this command. It can delete your files(!). There are many stories about accidental deletes of huge amounts of information; here is a story about how Pixar almost deleted [Toy Story 2](#) .

Part 3: File transfer

Download the file Number_Translation_Template.c from canvas to the computer you are using. File links for all projects will be at the top.

Open a terminal and navigate to the directory where you saved this file.

To copy the file to the CSL machines we'll use a program called scp which is short for secure copy. On Windows this program comes packaged with Putty and is called pscp (short for Putty Secure Copy) provide the following arguments to scp:

```
prompt> scp Number_Translation_Template.c username@best-  
linux.cs.wisc.edu:private/CS354/Project1/Number_Translation_Template.c
```

or on Windows:

```
prompt> pscp Number_Translation_Template.c username@best-  
linux.cs.wisc.edu:private/CS354/Project1/Number_Translation_Template.c
```

You should be prompted for you password and see a message indicating that the file transfer was successful.

Go back to the CSL Linux terminal and verify the file was transferred correctly with ls.

Next let's rename this template file. We have two choices here: cp (short for copy) and mv (short for move). Copy will retain the original file and mv will delete the original file. Both programs take two arguments: the first is the filename of the file to be copied and the second is the new filename.

```
prompt> cp Number_Translation_Template.c Number_Translation.c
```

```
prompt> mv Number_Translation_Template.c Number_Translation.c
```

My goal by providing files with the word Template is that students do not turn in the original template instead of the file they did their work in. This is the number 1 mistake that results in getting a zero on an assignment. Don't worry if you make this mistake it happens to several students every semester. We will work with you if this happens so you can submit your real assignment.

From here on do all your work in the Number_Translation.c file.

Part 4: Converting Integers to Binary and Hexadecimal

The actual homework part of this homework is pretty easy. It's just about converting some numbers from decimal to binary and hex forms.

Compile and run the code with:

```
prompt> gcc -o Number_Translation Number_Translation.c -m32 -Wall  
prompt> ./Number_Translation
```

The template is hard coded to prompt for user input, but then doesn't wait for the user to input a number and instead just assumes the user entered 19. It then prints out the number as decimal, binary, and hexadecimal representations.

Now look at the code in your favorite text editor. Some popular text editors used in the Unix/Linux OS environment are:

1. vim
2. emacs

Remember learning to use vim or emacs will definitely help you in the longer run to become a better programmer!

Use the text editor of your choice to open the file Number_Translation.c.

At the top you will see a section for compiler directives which so far have only a single include statement and some defines. The defines declare numerical constants that we will use for array lengths. Using this will prevent us from needing to hardcode array lengths in the rest of the program.

Below that are the function prototypes or declarations. These are needed because the C compiler needs to check correctness of function calls as it reads the code. It needs to know the expected types of the input and output. If you add any helper functions while doing the assignment you will need to add prototypes for those functions here. Do not change the prototypes for the three functions already here.

The main function comes next. There is no need to change main for this assignment. Feel free to add additional lines to help debug your functions as needed but remove them before you submit your assignment. The main function gets a number from the user, converts it into a binary representation with the bits stored in a character array, and does the same thing with the symbols of the hexadecimal representation. Then prints these out separated by tabs.

Following main are the three functions you will need to write for this assignment. Note that they have currently been hard coded to use the number 19 so that you could run the code and see what the output is supposed to look like.

Get_Number_From_User()

Prompt the use to enter a positive integer. Then use [scanf](#) to get the integer entered by the user and return it.

Convert_Number_To_Binary()

This function has two parameters. The first is the number to be converted to binary and the second is a pointer to the array that we will store the result in. Remember that you can use pointers just like arrays. If you are new to pointer you can think of this as a way to pass by reference, but pointers are very powerful, and you have many options for how you may use them.

Currently this function is hard coded to replace the contents of the binary string with the bits representing 19. Note that strings when stored in character arrays must be terminated by the \0 null character.

To complete this function. Go to the Internet and find some code for turning an integer into a its binary representation as a string. You may need to modify what you find to fit the parameters of this assignment. Add comments to the function you found to describe how it works to demonstrate that you understand the code you are using.

Cite the URL where you found the code you are using.

Convert_Number_To_Hexadecimal()

This function has three parameters. Use either num or binary as input (your choice) and translate the number into hexadecimal. Write this function on your own without copying from the Internet. I have hardcoded the output for the hexadecimal representation of 19 so you could see what the output should look like. Replace the code provided.

Testing

Please make sure that your program prints the output exactly as the template file, because we use automatic scripts to test your code.

Part 5: Turning in your work

From a terminal on your home computer use the following command to download your work to your home computer:

```
prompt> scp username@best-linux.cs.wisc.edu:private/CS354/Project1/Number_Translation.c  
./Number_Translation.c
```

or on Windows use pscp:

```
prompt> pscp username@best-linux.cs.wisc.edu:private/CS354/Project1/Number_Translation.c  
./Number_Translation.c
```

Not that if you open the file on your home computer and then resave it may change the line ending symbols. Linux uses one bit pattern to represent end of line characters and Windows uses another. It would be best to make all changes on the CSL machines.

Upload your assignment to Canvas.