CS1 Linux && SVN LAB!

To start this lab you will need to boot into Ubuntu. If you are currently on windows restart your machine when the GRUB bootloader appears choose Ubuntu. (It should be the choice right below windows 7) You log in with your CS username and password.

This lab will teach you how to use subversion and get you familiar with Linux and how to compile code without Code::Blocks! Let's start by opening up the terminal.

Subversion: Repository

Subversion (svn) is a type of "software repository" which holds any kind of file resource in a central server so large changes can be made from multiple locations and by multiple users without causing conflicts. In this course, we will use it as a project submission system, though it is more than that. It also allows for undoing changes since it keeps all history, and it can be used to track how regularly changes are being made.

The first thing we want to do is checkout our repository. This will pull down all of the source code that is stored on the subversion server. To do this enter the following command.

```
svn checkout https://dev.cs.uakron.edu/svn/cs209xx15/NNN/students/UANETID
```

Be sure to change xx to the appropriate semester (fa || sp), NNN is your section number and replace UANETID with your UA NetId. You should now have a folder in your home directory. Issue the command "ls" to print out the files in the current directory.

Navigating The Terminal

Change directories into your svn repository. To do this use the "cd" command followed by the directory name. In linux you current directory is referenced by "." and the previous directory is can be referenced by "..". To leave this directory and go back you can enter "cd ..". This should take you back to your home directory. The '~' character will take you back to your home directory no matter where you are. To go to your home directory you simply issue the command "cd ~". If you get lost you can print out your working directory using the command "pwd". If you type this you will see that the full path starts with a '/' instead of C:. This is called the root directory of linux.

Navigate to your subversion repository we just checked out. Print out the contents of this repo (HINT: Remember the ls and cd command). Lets begin by creating a README.txt document. To create a file use the touch command.

```
touch README.txt
```

This will create the README.txt in your current directory. To edit this file you could open it up with any text editor but you can also edit it within the terminal. We will use vim to edit this document.

```
vim  README.txt
```

This will start vim and you can add text. To start typing press the insert key or the "i" key to get into insert mode. From here type a simple message about this lab. When you are finished typing to save and exit press the escape key then type ":x" and hit enter. You can also type ":wq" instead of ":x" and that will write then quit.

Now that we have a new file with some information in it we need to add it to our repository and send it to the SVN server. To add a file to subversion use the svn add command.

```
svn add README.txt
```

Now that we added it to our repository we are ready to commit and send our file to the server. Enter the following command to commit your changes to the subversion server.

```
svn commit -m "Your commit message goes here!"
```

Make sure you replace the commit message with a detailed message about your changes. To check the status of your files you can use the svn status command. This will print out your files and let you know if you have any files that have been modified, deleted or added and need to be committed to the repository. Next issue the the status command to see what the current status of your repository is.

```
svn status
```

Now create a main.cpp and write a simple hello world program. Make sure you add it to svn and commit it.

Once you have a cpp file compile the code from the commandline. The compiler we will use is g++. To use g++ you type the command followed by the file(s) that you want to compile. In this case it is our main.cpp file. If you want to use C++11 features you need to add a flag when compiling. This is accomplished with the -std=c++11. This will enable C++11 features such as the stoi function. The -Wall will enable all warnings.

```
g++ main.cpp -std=c++11 -Wall
```

By default your program will be called "a.out" to run your program from the command line enter the following command.

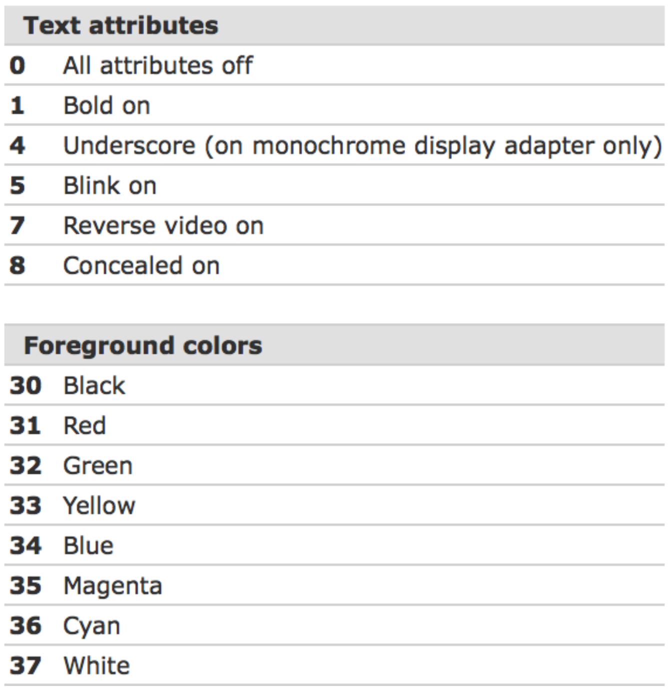
```
./a.out
```

To set the name of your executable you can add the -o option followed by the name you want your program to be called when using g++. Try and compile your program again but this time set the output program to be called "my_program" then run it via the commandline. When type commands at the terminal you can press the tab key and it will try to autocomplete the the command for you! You can also use the up and down arrow keys to look through previous commands so you don't have to keep typing them.

Now that you are familiar with the command line open up the main.cpp file again. You will create a function that will manipulate the color of the text in the terminal. To do this we will use ANSI codes. The ANSI code to print a color is “\033[0;##m” where ## is a number 30 - 37. This will change all text appearing after this until it reaches another ANSI code. It is important to turn off color before your program ends otherwise it will continue to print in color until it reads “\033[0m”.

Your function should take two arguments. One should be a Color and the other is a string that should be changed. The color we be determined by an enumeration. An enumeration is a distinct type whose value is restricted to one of several explicitly named constants. Use the C++ keyword *enum* called Color. Inside of your enum you should hold all of the foreground colors found in the table below. Make sure that they are in the correct order and that black starts at 30. The function will return a string with the color ANSI code at the beginning and a trailing ANSI code to turn off color. (HINT: Use the c++ 11 to_string() function to turn the value of your color to a string)
std::string printColor(Color, std::string);

```
enum Color{
      black = 30,
      ...
};
```



Text attributes	
0	All attributes off
1	Bold on
4	Underscore (on monochrome display adapter only)
5	Blink on
7	Reverse video on
8	Concealed on

Foreground colors	
30	Black
31	Red
32	Green
33	Yellow
34	Blue
35	Magenta
36	Cyan
37	White

Once you have the printColor function working remember to commit your changes. Do not add the executable to svn!

Now that you can print color to the command line we will create a simple program that will take two commands from the command line and print the message in the specified color. Open your main.cpp file again and set up your main function to accept command line arguments. To do this add two parameters to the main function. The first parameter is an integer called argc and the second parameter is an array of character pointers called argv.

```
int main(int argc, char *argv[]){
      …
}
```

The first parameter argc (argument count) will hold the number of arguments passed in from the command line. The second parameter argv (argument value) is an array of C-Strings which will be the arguments passed in. The first argument (argv[0]) will always be the program name. and argc will always be at least 1.

Our program will echo the user’s input to the color they specified. For example the user could run our program by entering the following command and the message in the quotes will be printed the color that was specified as the first argument.

```
./a.out green “This will appear green”
```

Inside of your main function make sure that the user entered exactly 2 arguments to our program. (Hint: use argc to check) Print out a help message to standard error (cerr) if the user did not enter the correct number of arguments.

If the user entered the correct number of arguments read the first argument and determine which color to print. (Hint: read this into a standard string so you can use the == operator) Set a Color variable (from your enumeration) equal to the correct color. If the user entered an invalid color inform the user then print out a message stating all of the available colors and exit the program. Then use your print color to print out the message that was entered at the command line (argv[2]) and exit your program.

Compile your program and name the executable echo_color. You now have a program that will print out a message to the console with different colors. Make sure you commit your changes once you have completed the lab.

If you have any questions be sure to ask your lab instructor!