CS 111 - Week 8 Lab Exercise - Part 1

Deadline
Due by the end of lab on Friday, October 14

How to submit
Submit your files using ~st10/111submit on nrs-labs, with a homework number of 88.

Purpose
To set up some tools in your nrs-labs.humboldt.edu account that we will be using to start C++.

Your tasks

• Use ssh/PuTTY/Secure Shell client to log in to your nrs-labs.humboldt.edu account.

• It is a UNIX/Linux "tradition" to put personal tools into a directory/folder named bin. Create yours:

  cd
  mkdir bin
  chmod 700 bin

  – (if you happen to already have a bin directory, nrs-labs will complain -- just move on in that case.)

• Now, make your own copies of the class C++ tools in your bin directory by typing the commands:

  cd
  cp ~st10/expr_play bin
  cp ~st10/funct_play bin
  cp ~st10/funct_compile bin
  cp ~st10/compile-helper bin

• Show me you have copied these by carefully typing the following commands. First, you are making a new directory for this lab and going to it, and creating two files showing me your bin directory contents:

  cd
  mkdir lab8
  cd lab8
  ls -ld ~/bin > bin-info1.txt
  ls -l ~/bin > bin-info2.txt

• You want to be able to run these programs from any nrs-labs directory. And, you want to be able to conveniently run the C++ programs you design, write, compile, and debug. So, you need to now modify your nrs-labs' shell's path that it checks for commands, to add your new bin directory to this path, and your current working directory to this path.

  CAREFULLY follow these instructions to do this:
- GO BACK to your home directory:
  `cd`

- RUN this command to carefully make changes to a special file named `.bashrc`:
  `~st10/add-path`

- Your `.bashrc` is executed for you every time you log into nrs-labs -- now it will set your path to include your `bin` directory and your current working directory every time.

  BUT we also want to run it right now -- either log off and use PuTTY to log in again, OR run the command:
  `source .bashrc`

  And now, you will check if you did the above correctly as you also try out the simplest of these tools, `expr_play`, which simply lets you type a single C++ expression that doesn't use variables and see its value:

- if you have copied these tools over correctly, you can run this tool by simply typing `expr_play` -- we're just going to make sure you are in your `lab8` directory first:
  ```
  cd          # make sure you are in your home directory
  cd lab8     # change to your lab8 directory
  expr_play   # run expr_play tool to "play" with C++ expressions
  ```

- If all is well, you'll see:

  `----------------------------------------------`
  `Welcome to the 2014-03-14 version of expr_play!`
  `----------------------------------------------`

  Are there any already-created C++ functions (in the current working directory) which you would like to be able to use within C++ expressions?
  (type y if so, n if not)
  your answer:

  ...answer `n` here, you haven't yet written any C++ functions you want to call, yet.

- You'll now see:
  ```
Enter a C++ expression, and type enter (or type q to quit):
```

- NOTE the following:
  - C++'s `bool` literals are `true` and `false`
  - C++'s `int` literals are like BSL Racket's `number` literals that don't have decimal points
  - C++'s `double` literals are like BSL Racket's `number` literals with decimal points
  - For one of C++'s string types, `char*`, the literals are like BSL Racket's `string` literals: anything surrounded by double quotes

- Given the above, you should be able to type four simple C++ expressions, one each of each of the above four C++ types.
– You should see a message showing the value of your expression printed to the screen. Really, this script is building a little main function that just prints the value of the expression you typed, and then compiling and running it for you!

– Type as many more expressions as you would like, and when you are ready, at:

Enter next C++ expression and type enter
(or type q to quit):

...type q to quit.

– Now list your lab8 directory's contents: ls

...and you'll see your bin-info1.txt, bin-info2.txt, and a little pair of files for each C++ expression you tried in expr_play, try_expr1 and try_expr1.cpp, try_expr2 and try_expr2.cpp, etc.

– We'll be talking about these later -- but note that the .cpp files are the little C++ main functions that expr_play built to run your C++ expressions, and the files with no suffix are the executable programs resulting from compiling the .cpp files.

• To complete the lab exercise, submit your .txt and .cpp files, using:

~st10/111submit

...with a lab number of 88. If you see bin-info1.txt, bin-info2.txt and at least try_expr1.cpp, then you should be done with Part 1 of this week's lab exercise.