ECE 4122/6122 Lab 0: Getting PACE-ICE Access and Using the g++ Compiler

(100 pts)

Category: Getting Started
Due: Tuesday September 1st, 2020 by 11:59 PM

This assignment is very simple. Write a C++ program using the insertion stream operator and escape sequences that outputs the following text to your terminal screen when executed:

```
My name is: (your first and last name separated by a space)
This (‘) is a single quote.
This (“) is a double quote.
This (/) is a forward slash.
This (\) is a backslash.
```

This program is very simple with no user input, command arguments, or file output. You can place all the code in your *main*() function.

Your PACE-ICE accounts have already been created. As a part of this assignment, we would like for you to create and run your program on coc-ice.pace.gatech.edu. Please use Appendix A to learn how to gain access to this cluster.

If you are trying to logon off campus, you will need to setup a VPN. Instructions are on the OIT website at Georgia Tech.

Turn-In Instructions

Please upload your source code (i.e. the text file that contains your program) on the assignment section of Canvas. In the file header comment section, at the beginning of your code please include the following statement:

```
/*
Author: your name
Class: ECE ????
Last Date Modified: date

Description:

I,          your name         , have successfully accessed my pace-ice account. I am
able to transfer files from my pace-ice account to my local machine using an
ftp client. I am ready for the next assignment on pace-ice.

*/
```

CREATING AND SUBMITTING LAB0 PROGRAM ON PACE-ICE

Creating Source Code. For this assignment you need to create the file on the PACE-ICE system and debug any issues. Once logged into PACE-ICE, you will need to use one of several available text editors found on the PACE-ICE cluster. Those are vi/vim, nano and emacs. Other instructors might recommend that you become familiar with vi (I, on the other hand, will likely use nano). To that end, there are many online tutorials to help you get used to vi. You could get started here:

https://coderwall.com/p/adv71w/basic-vim-commands-for-getting-started

Basic Unix Commands. Please make a directory (i.e. "folder") called `Lab0` where you keep your files. To make this directory use the following command in your home directory:

```
mkdir Lab0
```

To go into this directory from your home directory, you can use the following command:

```
cd Lab0
```

To get back to your home directory you can use the command:

```
cd
```

Compiling Source Code. On the PACE-ICE system we will be using the gnu g++ and gcc compiler. Use the following command to compile your source code and create an executable file called `testProgram`.

```
g++ testProgram.cc –o testProgram
```

To run your program at the command prompt, type in the executable file name

```
./testProgram
```

Submitting Assignments. You will need to submit your SOURCE code on Canvas. You will need to download the SOURCE code from PACE-ICE to your local machine, and then upload to the Canvas assignment. There are instructions in the class slides to help you do this.

The TAs in the class will compile and run your code for grading. Make sure that your code compiles correctly on PACE-ICE. If it does not you could lose up to 30-40 points off your lab!

Grading Rubric:

If a student's program runs correctly and produces the desired output, the student has the potential to get a 100 on his or her homework; however, TA's will look through your code for other elements needed to meet the lab requirements. The table below shows typical deductions that could occur.

AUTOMATIC GRADING POINT DEDUCTIONS PER PROBLEM:

Element	**Percentage Deduction**	**Details**
Does Not Compile	40%	Code does not compile on PACE-ICE!
Does Not Match Output	10%-90%	The code compiles but does not produce correct outputs.
Clear Self-Documenting Coding Styles	10%-25%	This can include incorrect indentation, using unclear variable names, unclear/missing comments, or compiling with warnings. (See Appendix A)

LATE POLICY

Element	**Percentage Deduction**	**Details**
Late Deduction Function	score - (20/24)*H	H = number of hours (ceiling function) passed deadline note : Sat/Sun count as one day; therefore $H = 0.5*H_{weekend}$

Appendix A

Accessing PACE-ICE Instructions

ACCESSING LINUX PACE-ICE CLUSTER (SERVER)

To access the PACE-ICE cluster you need certain software on your laptop or desktop system, as described below.

Windows 10 Users:

Search on your startbar for the application called Power Shell. Once you launch this you will have a bash window that you can use to access files on your machine AND access the PACE-ICE server. Once you have a command prompt, you can use the following commands.

1. Once you're in the PowerShell, ssh into PACE-ICE by typing:

   *ssh **YourGTUsername**@ coc-ice.pace.gatech.edu*

   where **YourGTUsername** is replaced with your alphanumeric GT login. Ex: bkim334

2. When it asks if you are sure you want to connect, type in:

   yes

   and type in your password when prompted (Note: When typing in your password, it will not show any characters typing)

3. You are now connected to PACE-ICE. You can edit files using vim by typing in:

   vi filename.cc OR *nano filename.cpp*

   For a list of vim commands, use the following link:

   https://coderwall.com/p/adv71w/basic-vim-commands-for-getting-started

4. You are able to edit, compile, run, and submit your code from this command line.

Windows Users:

Option 0 (Using SecureCRT)

The Georgia Tech Office of Information Technology (OIT) maintains a web page of software that can be downloaded and installed by students and faculty. That web page is:

http://software.oit.gatech.edu

From that page you will need to install SecureCRT.

To access this software, you will first have to log in with your Georgia Tech user name and password, then answer a series of questions regarding export controls.

Connecting using SecureCRT should be easy.

1. Open SecureCRT, you'll be presented with the "Quick Connect" screen.
2. Choose protocol "ssh2".
3. Enter the name of the CoC machine you wish to connect to in the "HostName" box (i.e. *coc-ice.pace.gatech.edu*)
4. Type your username in the "Username" box.
5. Click "Connect".
6. A new window will open, and you'll be prompted for your password.

Option 1 (Using Ubuntu for Windows 10):

Option 1 uses the Ubuntu on Windows program. This can only be downloaded if you are running Windows 10 or above. If using Windows 8 or below, use Options 2 or 3. It also requires the use of simple bash commands.

1. Install Ubuntu for Windows 10 by following the guide from the following link:

   https://msdn.microsoft.com/en-us/commandline/wsl/install-win10

2. Once Ubuntu for Windows 10 is installed, open it and type the following into the command line:

   *ssh **YourGTUsername**@ coc-ice.pace.gatech.edu*

   where **YourGTUsername** is replaced with your alphanumeric GT login. Ex: bkim334

3. When it asks if you are sure you want to connect, type in:

   yes

and type in your password when prompted (Note: When typing in your password, it will not show any characters typing)

4. You are now connected to PACE-ICE. You can edit files using vim by typing in:

   vi filename.cc   OR   *nano filename.cpp*

   For a list of vim commands, use the following link:

   https://coderwall.com/p/adv71w/basic-vim-commands-for-getting-started

5. You are able to edit, compile, run, and submit your code from this command line.

Option 2 (Using PuTTY):

Option 2 uses a program called PuTTY to ssh into the PACE-ICE cluster. It is easier to set up, but does not allow you to access any local files from the command line. It also requires the use of simple bash commands.

1. Download and install PuTTY from the following link:

   www.putty.org

2. Once installed, open PuTTY and for the Host Name, type in:

   coc-ice.pace.gatech.edu

   and for the port, leave it as 22.

3. Click Open and a window will pop up asking if you trust the host. Click Yes and it will then ask you for your username and password. (Note: When typing in your password, it will not show any characters typing)

4. You are now connected to PACE-ICE. You can edit files using vim by typing in:

   vim filename.cc   OR   *nano filename.cpp*

   For a list of vim commands, use the following link:

   https://coderwall.com/p/adv71w/basic-vim-commands-for-getting-started

5. You are able to edit, compile, run, and submit your code from this command line.

MacOS Users:

Option 0 (Using the Terminal to SSH into PACE-ICE):

This option uses the built-in terminal in MacOS to ssh into PACE-ICE and use a command line text editor to edit your code.

1. Open Terminal from the Launchpad or Spotlight Search.

2. Once you are in the terminal, ssh into PACE-ICE by typing:

   *ssh **YourGTUsername**@ coc-ice.pace.gatech.edu*

   where **YourGTUsername** is replaced with your alphanumeric GT login. Ex: bkim334

3. When it asks if you're sure you want to connect, type in:

   yes

   and type in your password when prompted (Note: When typing in your password, it will not show any characters typing)

4. You are now connected to PACE-ICE. You can edit files using vim by typing in:

   vi filename.cc OR *nano filename.cpp*

   For a list of vim commands, use the following link:

   https://coderwall.com/p/adv71w/basic-vim-commands-for-getting-started

5. You are able to edit, compile, run, and submit your code from this command line.

Linux Users:

If you are using Linux, follow Option 0 for MacOS users.

Appendix B

Transferring files from PACE-ICE to your local machine

<u>MacOS or Linux Users:</u>

<u>Option 1:</u>

After you open a terminal window you can use the following command at the command prompt. I would encourage you to be in the directory on your local machine that you are transferring files to or from PACE-ICE.

sftp username@coc-ice.pace.gatech.edu

After you are logged into PACE-ICE, you can then use the following basic commands to transfer files.

get filename1.cc

This command will “get” the file (with name ‘filename1.cc’) from the server and put into your local machine.

put filename2.cc

This command will “put” the file (with name ‘filename2.cc’) from your local machine onto the server.

bye

This will end your sftp session.

You can also use the following command to manipulate the current directory on PACE-ICE.

cd directoryName

This will change the current directory on the server.

ls

This will list the current directory files on the server.

lcd directoryName

This will change the current directory for your local machine.

lls

This will list the files in the current directory on your local machine.

Option 2:

From your terminal while logged into to your local machine, you can use the following commands to transfer files or whole directories. You will be prompted to enter your password. Here are a few examples of the command `scp`:

To copy the file ‘filename.cc’ from your local machine to server (PACE-ICE), assuming you have a directory ‘Lab0’ on PACE-ICE in your home directory.

```
scp filename.cc username@coc-ice.pace.gatech.edu:Lab0
```

To copy a file from the server’s (PACE-ICE) home directory to your local machine’s current directory

```
scp username@coc-ice.pace.gatech.edu:filename.cc ./
```

To copy the entire directory ‘Lab0’ from your local machine to the server (PACE-ICE) Lab1 in home directory (Note that this will overwrite the existing Lab1 on your server).

```
scp –r Lab0 username@coc-ice.pace.gatech.edu:
```

To copy entire ‘Lab0’ directory from the server (PACE-ICE) to your local machine in current directory. (Note that this will overwrite the existing Lab1 on your local machine).

```
scp –r username@coc-ice.pace.gatech.edu:/Lab0 ./
```

Windows Users:

For Window users, you can use WinSCP application to transfer files from PACE-ICE to your local machine using a simple drag and drop method. You can download the software at the following URL. Instructions are also on this website.

https://winscp.net

Appendix C: Coding Standards

Indentation:

When using *if/for/while* statements, make sure you indent 4 spaces for the content inside those. Also make sure that you use spaces to make the code more readable.
For example:

```
for (int i; i < 10; i++)
{
    j = j + i;
}
```

If you have nested statements, you should use multiple indentions. Each { should be on its own line (like the *for* loop) If you have *else* or *else if* statements after your *if* statement, they should be on their own line.

```
for (int i; i < 10; i++)
{
   if (i < 5)
   {
       counter++;
       k -= i;
   }
   else
   {
       k +=1;
   }
   j += i;
}
```

Camel Case:

This naming convention has the first letter of the variable be lower case, and the first letter in each new word be capitalized (e.g. firstSecondThird). This applies for functions and member functions as well! The main exception to this is class names, where the first letter should also be capitalized.

Variable and Function Names:

Your variable and function names should be clear about what that variable or function is. Do not use one letter variables, but use abbreviations when it is appropriate (for example: “imag" instead of “imaginary”). The more descriptive your variable and function names are, the more readable your code will be. This is the idea behind self-documenting code.

File Headers:
Every file should have the following header at the top

```
/*
Author: your name
Class: ECE4122 or ECE6122
Last Date Modified: date

Description:

What is the purpose of this file?

*/
```

Code Comments:

1. Every function must have a comment section describing the purpose of the function, the input and output parameters, the return value (if any).
2. Every class must have a comment section to describe the purpose of the class.
3. Comments need to be placed inside of functions/loops to assist in the understanding of the flow of the code.