Lab 1 : The ECE244 Programming Environment

1 Objectives

The objectives of this assignment are multifold:

- to setup your remote connection to ECF machines so you can do your assignments,
- to install the `ECE244VM` virtual machine (only if you wish to use it),
- to learn basic UNIX commands you will need during the course,
- to setup and learn how to use the NetBeans IDE (*Integrated Development Environment*) to enter, browse, build, and debug your code.
- to learn how to submit your solutions to the `autotester` program,
- to provide you with practice on the edit-compile-run-debug cycle of C++ programs in a UNIX environment, and
- to practice how to connect with your lab TAs during your lab session to ask, questions and get help.

Even though the assignment uses C++, which you are still learning in class, it requires only knowledge of the C subset of the language that you are already familiar with. Thus, doing the assignment in C++ should be straightforward.

2 Preparation: Tutorials and Background Documents

Please read the three background documents listed below and do the exercises contained within. Some of these documents are located in the tutorials directory in lab1 release material. Others are released on Quercus in the "Course Handouts" module.

- Remote Access to ECF.pdf: A guide for remotely accessing ECF from your home machine/laptop
- ECF Unix guide.pdf: The ECF UNIX Environment, commands and filesystem
- Getting Started with NetBeans.pdf: A guide to getting started with NetBeans
- Introduction to Debugging.pdf: An introduction to Debugging with NetBeans
- Connecting to Lab TAs.pdf: a tutorial on how to connect with your lab TAs during your regularly scheduled lab session
- ECE244VM Handout.pdf: a guide to installing and using `ECE244VM`

It is critical that you read the entirety of these documents – taking shortcuts now will lead to you having a poor understanding of the tools we will use in this course, and make later labs very difficult to complete. Make sure in particular that you understand the process to check and submit your work. Some students receive a grade of 0 on a lab, not for failing to do the work, but because they did not submit the files correctly.

3 Problem Statement

In this lab you will setup your remote connection to an ECF machine and exercise connection to a lab TA during your lab session. Further, you will create a directory for ECE244 assignments in the file system under your UNIX account on ECF. You will then copy a simple C++ program, compile it, check it for correct output, edit the program, recompile and submit to the autotester. Next, you will use the GNU Debugger (GDB) from within NetBeans to debug several programs. The first few programs give in-depth guidance on how to accomplish the goals, while the latter programs leave you with just hints. To follow the hints, consult the debugging guide, which will have more details. Finally, and only if you decide to use it, you will install the `ECE244VM` virtual machine on your home computer.

4 Procedure

4.1 Remotely Accessing ECF Machines

All ECF labs are closed to in-person access. Thus, you must carry out your assignments by remotely accessing ECF machines from your home machine or laptop. The "Remote Access to ECF" handout describes multiple ways you can do so. Read the handout and follow the instructions to setup your remote access. **You will not be able to do the rest of this assignment until you remotely connect to an ECF machine**.

If you encounter problems setting up your remote access, then seek help from the TAs. You can do so outside your regularly scheduled lab session by posting the issue under the folder titled "remote_access_to_ecf" on Piazza. Alternatively, you can connect "live" with a TA `during` your lab session, as described in Section 4.2.

4.2 Connecting to Lab TAs

Even though there are no in-person labs in ECE244, there are regularly scheduled lab sessions (consult your timetable to find out your lab session). During these sessions, TAs will be online for you to have a "live" connection with, ask questions and receive help. You will be able to share your screen with the TA to point out an issue, a problem or an error. Further if necessary, you can grant the TA control of your screen, effectively giving him or her control to your keyboard and mouse. While at the end this is not as effective as face-to-face communication, it is the closest that can be.

Please read the handout titled "Connecting to Lab TAs", released in the "Course Handouts" module on Quercus. Follow the instructions to connect to one of the lab TAs. Once connected to a TA over Zoom, do the following:

- Share the ECF remote connection window with the TA
- Type `hostname` at the command prompt
- Stop sharing your screen and leave the Zoom meeting
- Proceed to the rest of the assignment

4.3 Setting up your directory

Create a new directory for this course called `ece244`, and protect it by setting the permissions so only you can read or write it. See the "ECF Unix Environment" document reading for details (hint:

`chmod` $-R$ `go-rwx ece244`). Make your `ece244` your working directory. Create a ditectory called `lab1` in your `ece244` directory and set its access permissions similar to the `ece244` one. This `lab1` directory will contain all your lab1 files.

It is your responsibility to ensure that others cannot copy your work. If another student is able to copy your work, and we determine that two submitted solutions stem from the same source, then we have no option but to take action against both parties.

Download the lab1 files from the course website. They will be in an archive called `lab1_release.zip`. Extract the file into your `lab1` folder using the command `unzip lab1_release.zip`[1] This will create the folders and files of the assignment.

4.4 Importing and building the "Hello, World" program

The classic "Hello, World!" program is shown below. It can be found in the `hello` folder under `lab1` which was extracted from the archive. Use the `cd` command to change your current directory to the `hello` directory.

```
1  // hello.cpp -- "Hello, world" C++ program
2
3  #include <iostream>
4  using namespace std;
5
6  int main () {
7
8      cout << "Hello " << endl;
9      cout << "world!" << endl;
10     return (0);
11 }
12
```

1. Open NetBeans (from the command line, you can type `netbeans` to start it).
2. Create a new project (File → New Project) from existing sources
   (a) Choose C/C++ from existing sources since the files are already there, then click next
   (b) Click the Browse button and select the folder you just created (`<...>/ece244/lab1/hello`)
   (c) Leave all other settings at default
3. In the project navigator at left, open `hello.cpp` by double-clicking. Note the C++ keywords are highlighted.
4. NetBeans will by default have built (compiled and linked) the program after importing. If you want to build again after making source changes, you can do so by clicking the hammer, or through the menu Run → Build Main Project. The hammer-and-broom icon is for clean and build (ie. delete all non-source files first)
5. Run the program by clicking the green right arrow button in the toolbar or from the menu Run → Run Main Project. You should see the expected output in the window pane below the source editor.

[1] How to use the zip and unzip commands is covered in the ECF Unix Environment document.

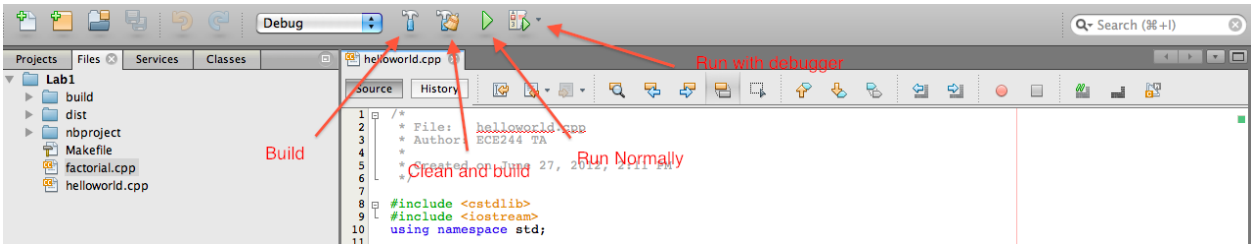

Figure 1: NetBeans basic toolbar for building and running projects

4.5 Starting with the debugger: using breakpoints

Before going on, be sure you have read the "Introduction to Debugging" document, and worked through the examples within it.

One of the most fundamental functions of the debugger is to "pause" a program either at a specified source line, or when a certain condition is true to let you inspect what is going on. The most basic way to do this is through setting a breakpoint, which will be illustrated below:

1. Set a breakpoint on the line where "world" is printed
2. Run the program normally, noting that it runs just like before: the breakpoint does nothing unless we start the debugger.
3. Now run with the debugger. Either click the button that shows a breakpoint with a green triangle at lower right (see Fig 2), or menu Debug → Debug Main Project[2].
4. Note that the program prints "Hello," and stops. The source line is also highlighted in green. The program execution has paused, waiting for your action. In this case, simply continue. The program should print the rest and finish.

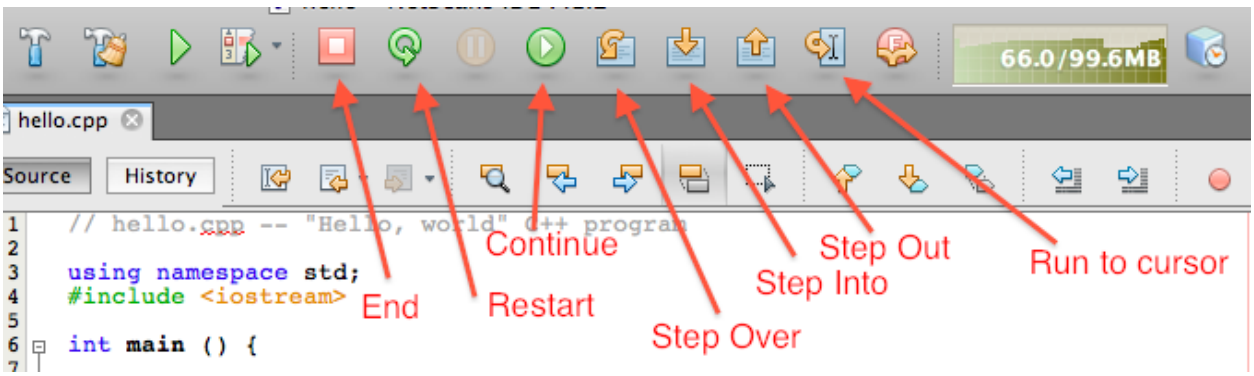

Figure 2: The debugging toolbar

[2]If you have multiple projects open, be sure that `hello` is the main before you do this. See the NetBeans intro for more detail

4.6 Using exercise to test your program

One utility provided to help you is the `exercise` script[3]. It will run your program and provide it with test input, then compare the output expected against what is actually provided. You will be notified whether or not the two differ. When you run the `hello` program, you should see:

```
Hello
world!
```

Figure 3: Output produced

Now let's test the program before submitting it to see if that works:

1. Run the program and verify that you get the output shown in Fig 3
2. Use the `exercise` script to test your `hello` program
3. You should find that it fails - it was expecting something more like Fig 4
4. Edit the code so that it produces the expected output
5. Test again, hopefully it works

```
Hello, world!
```

Figure 4: Expected output

You should run through this process for every lab that you submit to ensure you are producing correct output. Remember that output produced must be exactly what is specified in the handout - no difference is permitted in spaces, capitalization, spelling, etc. The functionality marks are produced by a test script which checks for exact equivalence. Note also that (where user input controls the program) the exercise script will only run one or a few test cases. In the future, you will be evaluated on more cases than are run in `exercise` to ensure your program can handle a range of input.

4.7 Finding a run-time bug with `harmonic.cpp`

The next program, found in the `harmonic` folder, is supposed to ask the user for a number and then print the corresponding harmonic number. The N-th harmonic number is defined as

$$\sum_{1}^{N} \frac{1}{i} = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{N} \tag{1}$$

In this program, it has been implemented as a `for` loop with a variable used to accumulate the sum as it goes. However, you will find that although the program compiles correctly and runs it does not produce the expected output.

1. Create a project and compile the file

[3]For details, read the document on the ECF Unix Environment

2. Run for a few test cases to see if correct output is produced
3. Find the bug. Suggested procedure:
   (a) Set a breakpoint within the loop body
   (b) Run the program with a reasonable (small) input number, say 3
   (c) Check the value of the variables (Variables window) when the breakpoint is encountered
   (d) Hit continue to go through the loop step-by-step and watch the values
4. Change the code and recompile
5. Re-test to ensure correct functioning

When you believe you have fixed the bug, you should run
`~ece244i/public/exercise 1 harmonic`
to confirm that the program generates the expected output.

4.8 Another run-time bug: temperature conversion with `convert.cpp`

Our third program takes a temperature in degrees Fahrenheit, Celsius, or Kelvin, and prints out its equivalent in the other scales. Where T_F, T_C, and T_K are the temperatures in each of the scales, the conversion between them is defined by the equations below:

$$T_F = \frac{9}{5}T_C + 32.0 \tag{2}$$

$$T_K = T_C + 273.0 \tag{3}$$

For your convenience in testing, several equivalent values with physical significance are given below.

Definition	° C	° K	° F
Boiling point (liquid nitrogen)	-196.0	77.0	-320.8
Freezing point (pure water)	0.0	273.0	32.0
Boiling point (pure water)	100.0	373.0	212.0
Normal body temperature	37.0	310.0	98.6

Table 1: Selected temperature values

1. Create a NetBeans project
2. Compile the program, fixing errors as needed[4]. NetBeans will help you navigate the error messages, and flag syntax errors. You can run

   `~ece244i/public/exercise 1 convert`

   to automatically run some test cases, but you should also try some test cases of your own.
3. If you find that it does not, try to find and fix the error(s). As often happens with software, be aware there may be more than one bug to be fixed.

[4]Read the "Getting Started with NetBeans" document which discusses some common errors

HINT Sometimes when debugging a process involving many steps, the *single-step* function in the debugger can be very useful to follow along. This can help you confirm that program control is following the path you think (or not!). Often, this is a far more efficient process than inserting random print statements or staring at code.

Important Make sure you do not change the output formatting. The exact same spacing and number of decimal places must be given to receive credit.

4.9 Setting up your environment to use C++ 2011

The C++ 2011 standard will be assumed in both developing and marking assignments. Thus, it is **essential** that your ECF environment be set up to use the appropriate C++ compiler, i.e., one that suppots the 2011 standard. This can be done with a simple command, and be done only **once** at the begining of the term. Login into your account and start a `terminal` window (i.e., an `xterm`). If you do not know how to start `terminal`, ask for help from one of the TAs.

Once the terminal starts, type the following command at the prompt (usually a `%` or a `$`):

```
ecf-sw-env -a ece244
```

The command produces no output. However, to take effect, you must **logout** from your ECF account and then **login again**. To make sure that you have the correct environment set up, start a terminal and type the following command at the prompt:

```
g++ -v
```

The command will print a few lines of output that should end with the following line if the environment is set up correctly to use C++ 2011 compilers:

```
gcc version 5.3.1 20160406 (Red Hat 5.3.1-6) (GCC)
```

If the `gcc` version is not `5.3.1` and you have typed the `ecf-sw-env` command correctly, please ask one of the TAs for help (see Section 4.2 for how to do so).

4.10 Setting up NetBeans to use C++ 2011

It is also essential to make sure that NetBeans is using the appropriate C++ compiler. Normally, this would be the case. However, you may have used NetBeans before with an older version of the C++ compiler (e.g., in APS105). Thus, NetBeans needs to be told to update the compiler version to the one you setup with the `ecf-sw-env -a ece244` command above. This can be done in a few simple steps. These steps need to be done only **once**.

1. Start NetBeans (if not already started).
2. Click on `Tools` menu and select `Options` from the dropdown menu.
3. Select `C/C++` to display the locations of the C/C++ compiler and associated tools (see Figure 5 below). Observe the location of the C++ compiler.
4. Click on `Restore Defaults` button at the bottom (see Figure 5 below) and click `OK` in the resulting dialog box. You may observe that the locations of the C/C++ tools has changed.

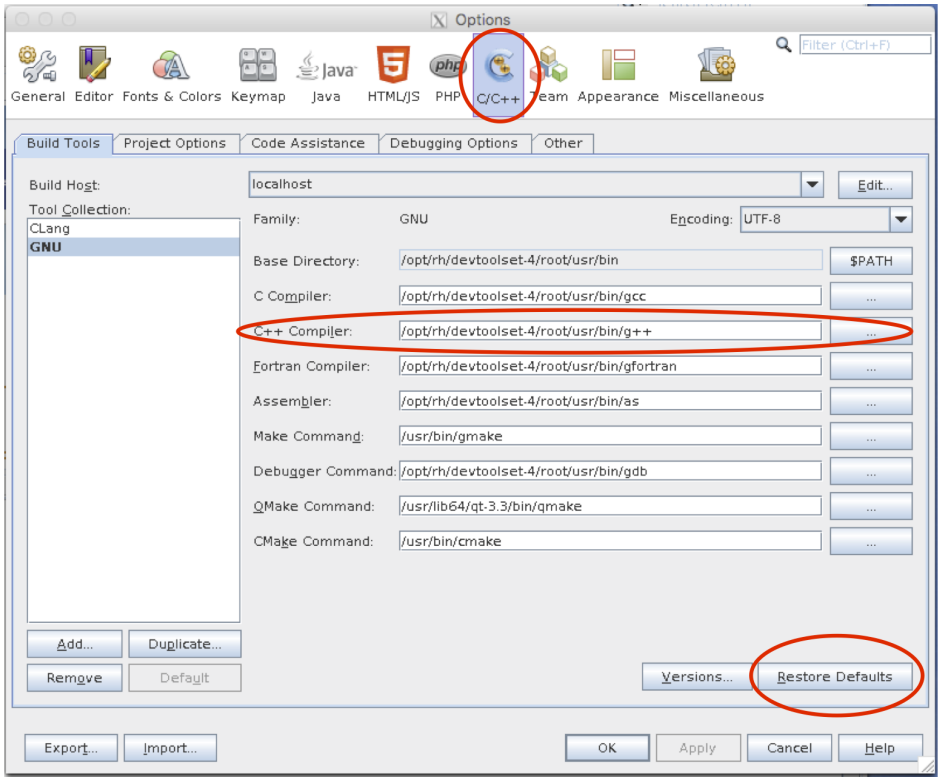

Figure 5: Window for restoring C/C++ Default Compiler

You can verify that NetBeans is setup correctly using the `test11.cpp` code shown below. You do not need to understand what the code is doing at this stage. The code can be found in the `test11` folder under `lab1`, which was extracted from the archive. Use the `cd` command to change your current directory to the `test11` directory.

```
#include <iostream>
using namespace std;

class Test11 {
  public:
   int x=9;
};

int main() {
  Test11 a;

  cout << a.x;

  return 0;
}


```

1. Open NetBeans
2. Create a new project (File → New Project) from existing sources
3. Choose C/C++ from existing sources since the files are already there, then click next

4. Click the Browse button and select the folder you just created (`<...>/ece244/lab1/test11`)

5. Leave all other settings at default

Build and run the code and if NetBeans is setup correctly, you will get no build or runtime errors.

Should you create a new project from scratch in NetBeans, make sure that you select the `C++11` option as shown in Figure 6.

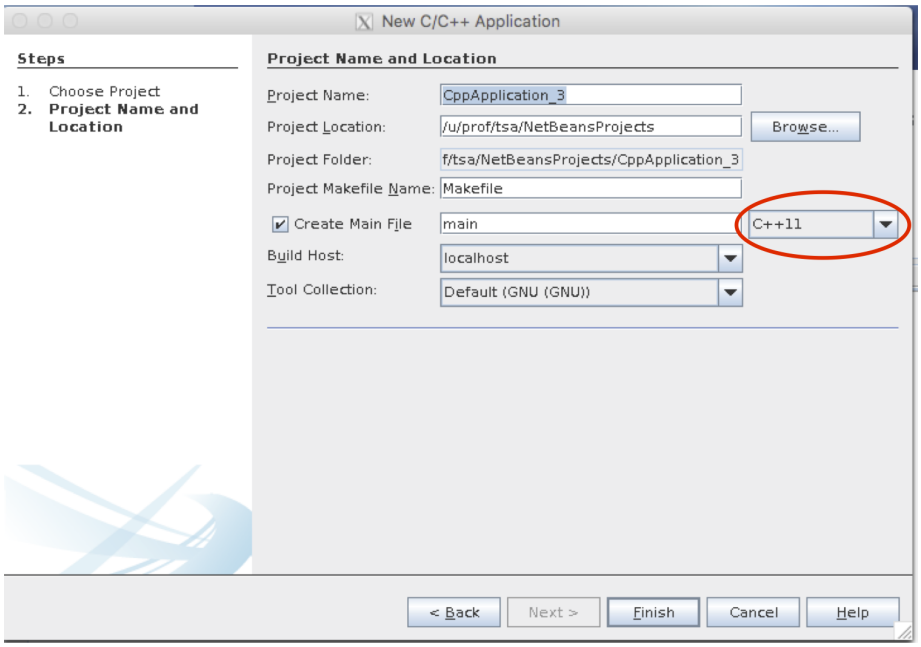

Figure 6: Make sure a new project is created for C++11

5 Installing `ECE244VM`

`ECE244VM` is a virtual machine (VM) that allows you to run a computing environment that is very similar to that of an ECF workstation on your home computer, whether it is a Windows or a Mac machine. Once installed, it enables you to do your assignments on your home computer as if you are doing them on an ECF workstation, without having to remotely connect to ECF. There are pros and cons to using `ECE244VM`. Please keep in mind that you do not need this VM to do any of the assignments for the course. Its use is entirely optional.

Please read the handout titled "ECE244VM handout" available in the "ECE244VM" module on Quercus. The handout details the pros and cons of using the VM as well as how to download, install and use it.

Should you encounter issues with the VM, please report them on Piazza in the folder titled "ece244vm_issues".

6 Deliverables

Please submit the following files to the autotester, after you have fixed the code in each so they produce correct output and pass `exercise`.

- `hello.cpp`
- `harmonic.cpp`
- `convert.cpp`

Remember to use `exercise` to test each program. When you are satisfied both programs are correct and perfectly match the expected output, copy all three files into one directory and use

`~ece244i/public/submit 1`

to submit the assignment for marking. **Carefully examine the output from the `submit` command to ensure your program was submitted correctly and passed basic checks**, or you may receive 0 for this part of the lab. You can submit your work as many times as you like; only the last submission will be marked. Also check that you have set your file permissions correctly to prevent others from copying your work.