“Hello World” Project

Object-Oriented Data Structures in C++ — Week 2
University of Illinois

Introduction

This project gives you a chance to demonstrate that you can comfortably navigate the IDE you’ve chosen. We’ll take the opportunity to demonstrate how to set up the project using Cloud9 as the IDE, but the basic principle should be similar if you are using a different Linux-compatible coding environment on your own computer. The previous reading lessons gave some tips about how to set that up. If you followed the Cloud9 tutorials, then you already know how to create a new Cloud9 workspace. If not, you might want to go back and read through those steps to make sure that you’re ready!

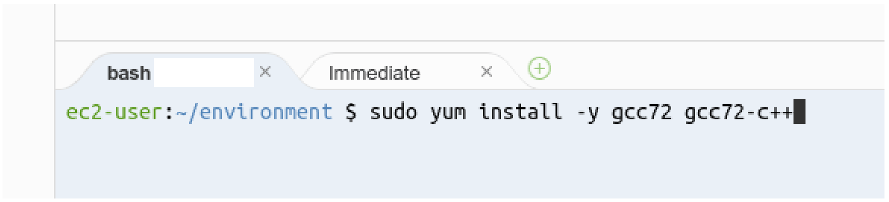
The classic “first program” to write in any given programming language is the *hello world* program, in which you print a simple message to the screen.

C++ Environment Setup

The Week 2 Project item on Coursera, where you found this PDF, includes a download link to the provided files: **`hello-world-given-files.zip`**. You should download it to your local computer so that you can then place it on your Cloud9 workspace (or load the files into your local IDE). In the Cloud9 environment you were able to create following the Week 1 readings, make sure you've updated the compiler first, by typing this command in the terminal:

```
sudo yum install -y gcc72 gcc72-c++
```

On the default Cloud9 system, this is the command for updating the C++ compiler that we’ll use.



```
ec2-user:~/environment $ sudo yum install -y gcc72 gcc72-c++
```

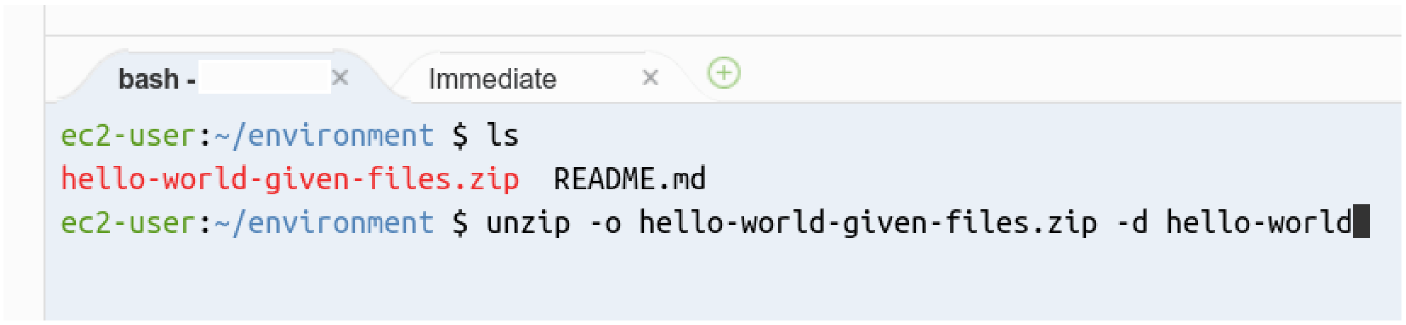
Now, in the Cloud9 interface, click **File** > **Upload Local Files**, and upload your local copy of hello-world-given-files.zip file to your Cloud9 workspace. After that, you'll see that the zip file appears in the environment file listing on the left of the screen.

Next, you'll want to extract the contents of the zip file to a new directory. Your terminal on Cloud9 already has some commands for this, **`zip`** and **`unzip`**. First, type **`ls`** in the terminal and make sure you see the zip file in the current directory listing. If not, then as described in the earlier readings, you should use the **`cd`** command to change directory until you see the file in the same directory. You can use the file listing on the left side of the window to help you see where you are. The **`pwd`** command (“print working directory”) also reports where your terminal is currently browsing in the file system.

Once in the same directory as the zip file, enter this command to extract the contents of the file:

```
unzip -o hello-world-given-files.zip -d hello-world
```

This will unzip the files into a "hello-world" subdirectory, but be careful, because this will overwrite files in the **`hello-world`** subdirectory if it already exists!

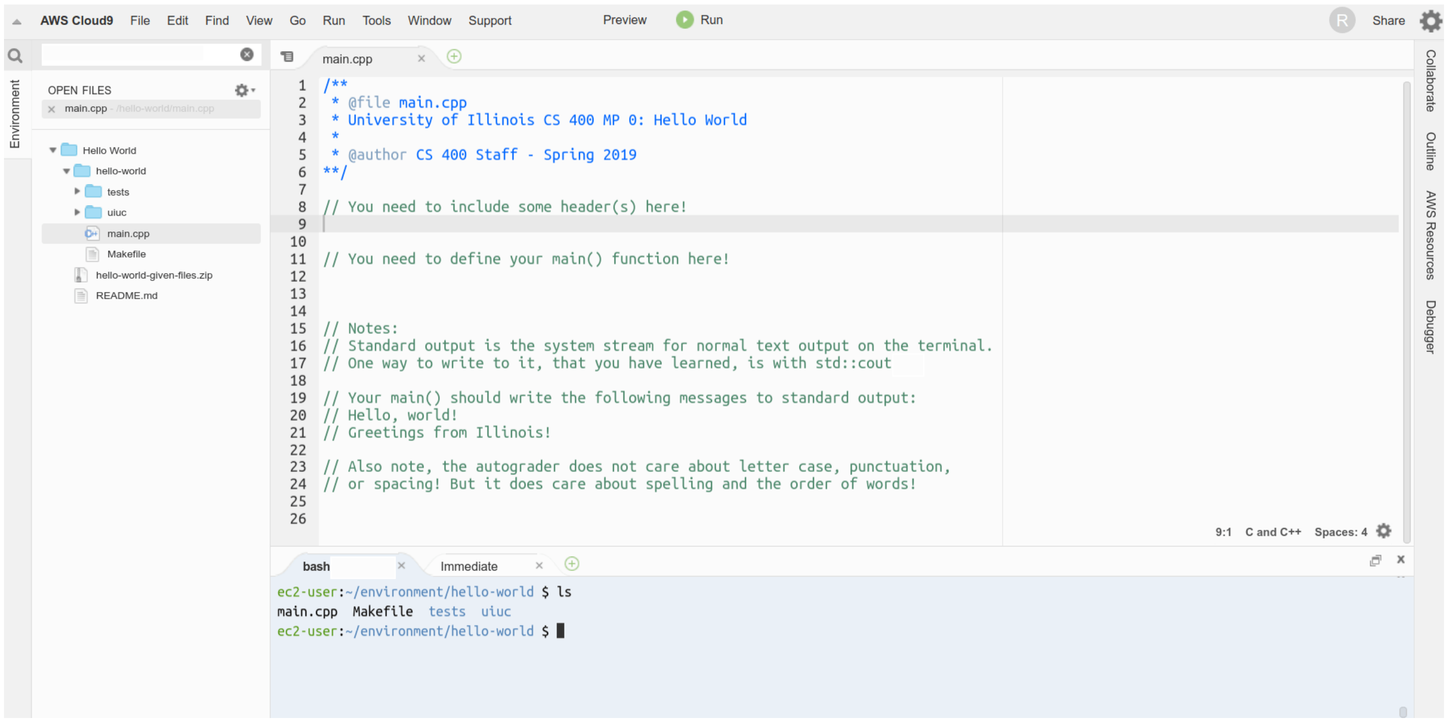


```
ec2-user:~/environment $ ls
hello-world-given-files.zip  README.md
ec2-user:~/environment $ unzip -o hello-world-given-files.zip -d hello-world
```

After you extract the code files, you can double-click them in the file list pane to see the editable document in the viewer. For this assignment, the file that you need to edit and submit is **`main.cpp`**, and the other files are there to help you compile and test your code.

Programming Objective: Say Hello!

If you open the **`main.cpp`** file in your editor, you'll see this:



```cpp
/**
 * @file main.cpp
 * University of Illinois CS 400 MP 0: Hello World
 *
 * @author CS 400 Staff - Spring 2019
**/

// You need to include some header(s) here!


// You need to define your main() function here!



// Notes:
// Standard output is the system stream for normal text output on the terminal.
// One way to write to it, that you have learned, is with std::cout

// Your main() should write the following messages to standard output:
// Hello, world!
// Greetings from Illinois!

// Also note, the autograder does not care about letter case, punctuation,
// or spacing! But it does care about spelling and the order of words!

```

```
ec2-user:~/environment/hello-world $ ls
main.cpp  Makefile  tests  uiuc
ec2-user:~/environment/hello-world $
```

The comments in the code file (which is otherwise empty!) let you know what to do this time: you should write a **`main()`** function that outputs text to the system standard output stream (**`cout`**). You might need to add an include before you can accomplish that. Here are the messages you need to write in particular:

"Hello, world!"

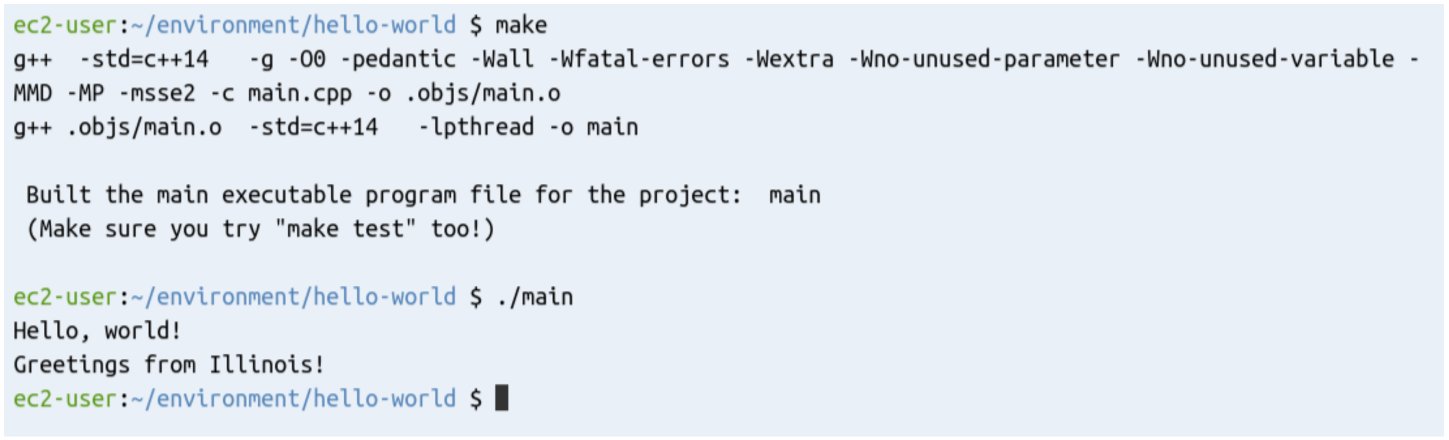
"Greetings from Illinois!"

For simplicity, the grader will ignore any punctuation or whitespace in your output, and the case of the letters won’t matter. (However you need to output all the correctly-spelled words in the proper order. Please mind how to spell “Illinois!”)

Compiling and Testing Your Code

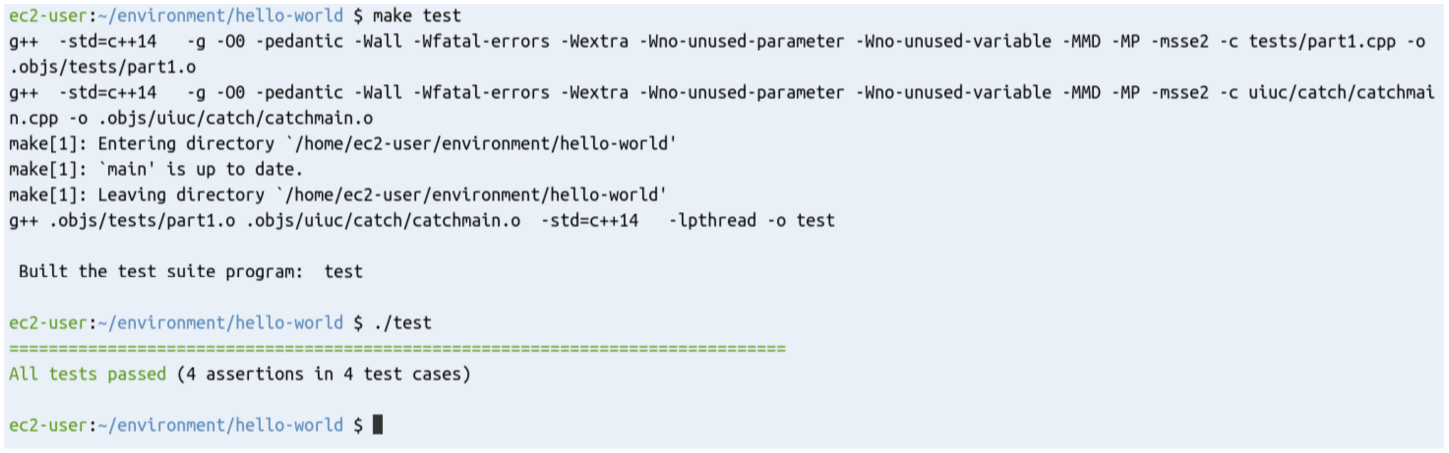
To compile the code, you must use the terminal to enter the same directory where the **`Makefile`** is stored; it's in the same directory where **`main.cpp`** is located. As explained in the readings, use **`cd`** to change to the appropriate directory, use **`ls`** to view the file list in that directory (just to make sure that you’re in the right place), and then type **`make`** to automatically start the compilation. If you need to clear out the files you've previously built, type **`make clean`**. If you encounter any warnings or errors during compilation, study the messages in the terminal carefully to figure out what might be wrong with your code.

If your compilation is successful, you’ll get an executable file simply called **`main`** in the same directory as **`main.cpp`**. You can try running it by typing **`./main`** while you’re in that directory. If successful, the output should look something like this:



```
ec2-user:~/environment/hello-world $ make
g++  -std=c++14   -g -O0 -pedantic -Wall -Wfatal-errors -Wextra -Wno-unused-parameter -Wno-unused-variable -
MMD -MP -msse2 -c main.cpp -o .objs/main.o
g++ .objs/main.o  -std=c++14   -lpthread -o main

 Built the main executable program file for the project:  main
 (Make sure you try "make test" too!)

ec2-user:~/environment/hello-world $ ./main
Hello, world!
Greetings from Illinois!
ec2-user:~/environment/hello-world $ █
```

As the compilation message suggests, you should also run the unit tests included in the test program. To compile it automatically, just type **`make test`** at the same prompt. If this compilation is also successful, you’ll see a file named **`test`** appear in the directory. Then, you can run it by typing **`./test`** similarly to before:



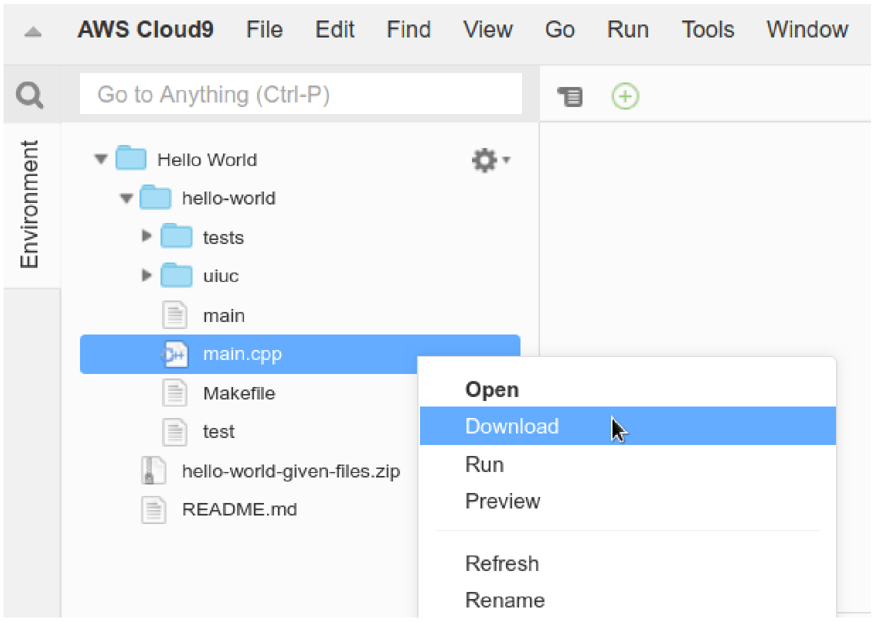
```
ec2-user:~/environment/hello-world $ make test
g++  -std=c++14   -g -O0 -pedantic -Wall -Wfatal-errors -Wextra -Wno-unused-parameter -Wno-unused-variable -MMD -MP -msse2 -c tests/part1.cpp -o
.objs/tests/part1.o
g++  -std=c++14   -g -O0 -pedantic -Wall -Wfatal-errors -Wextra -Wno-unused-parameter -Wno-unused-variable -MMD -MP -msse2 -c uiuc/catch/catchmai
n.cpp -o .objs/uiuc/catch/catchmain.o
make[1]: Entering directory `/home/ec2-user/environment/hello-world'
make[1]: `main' is up to date.
make[1]: Leaving directory `/home/ec2-user/environment/hello-world'
g++ .objs/tests/part1.o .objs/uiuc/catch/catchmain.o  -std=c++14   -lpthread -o test

 Built the test suite program:  test

ec2-user:~/environment/hello-world $ ./test
===============================================================================
All tests passed (4 assertions in 4 test cases)

ec2-user:~/environment/hello-world $ █
```

If you get any errors, the simplest thing to do is type **`make clean`** to delete any temporary files, then inspect your code to see if there's a typo. Also make sure you are writing your **`main()`** function in the standard format. Remember that it should have return type **`int`** and return **`0`** to show that no problems occurred. There are many more examples of outputting strings to the terminal in the lectures and reading lessons.

Submitting Your Work

When you're ready to hand in your work, you need to download a copy of your finished **`main.cpp`** file to your local computer. Then, you will submit that file on Coursera for autograding.

To download a copy of the file, you can just right-click on the file in the list pane on the left, then choose Download:



You can submit the **`main.cpp`** file alone as your answer for the Hello World Project on Coursera. After a few minutes, you'll see a grade report appear in the submission screen.

(Please also note: If you see a link on the Week 2 project submission page mentioning "upgrading" your assignment to a new version, be sure to click it, so that Coursera will update the assignment page you've already signed into to connect to the newest version of the autograder.)

If you have any questions, be sure to ask on the discussion forums!