



CS-2303, System Programming  
Concepts, C-term 2017

Lab 2 (10 points)  
January 24 & 25, 2017  
Due: at 11:59 pm on day of lab session

## Laboratory Assignment #2 — Eclipse CDT

### Objective

To learn to use the *Eclipse C/C++* Development Toolkit.

### Introduction

An *Integrated Development Environment* (IDE) is a software application (or set of software applications) to provide a coordinated set of editors, compilers, debuggers, and other tools for software development, both for individual programmers but more importantly for teams and large organizations. IDE's have been around for many years and have grown in features and functionality. *Eclipse* began in around 2000 as an IBM project to compete with Microsoft's *Visual Studio*. It has since grown into an open-source system capable of support a wide variety of languages and tools via *plug-ins*. Some students are already familiar with *Eclipse* for development of *Java* programs. *Eclipse CDT* is a configuration of *Eclipse* that has the plug-in for *C* and *C++* development already installed.

**Note:** *Eclipse* for *C/C++* has some idiosyncrasies that are different from *Eclipse* for *Java*. Therefore, *all* students should devote full attention to this lab assignment.

*Eclipse CDT* is installed on the course virtual machine. It is also freely downloadable to your Windows, Macintosh, or Linux personal computer or laptop for direct use, independent of the course virtual machine.<sup>1</sup> It is also available on the WPI CCC Linux system via X-Win32, but the network connection is too slow for interactive use of *Eclipse CDT* for any substantial project.

In this Laboratory Assignment, you will learn how to run *Eclipse* on your course virtual machine, import a simple *C* programming project into *Eclipse*, build it, and get it running. In subsequent assignments, we will learn how to take advantage of other tools in *Eclipse CDT*.

**Before you start:**— If you have not already done so, increase the size of the memory in your virtual machine from 2048 MB to 4096 MB. Do this by shutting down the virtual machine, changing its settings in *VirtualBox* or *VMware*, and then reboot.

### Getting Started

Please sign the attendance sheet.

---


<sup>1</sup> In a previous term, students used *Eclipse CDT* directly on their laptop and desktop computers. However, the differences between the *C* and *C++* implementations on Macintosh, Linux, and Windows made it impractical for student assignments in CS-2303. Moreover, *gcc* and *g++* were too fragile on Windows to be usable under the pressure of weekly course assignments.

Start up and log into your virtual machine. Open the *Firefox* browser in your virtual machine, and download and unzip the same zip file that you used for the **makefile** portion of Lab 1, namely from the following URL:–

[http://web.cs.wpi.edu/~cs2303/c17/Resources\\_C17/Lab1Files.zip](http://web.cs.wpi.edu/~cs2303/c17/Resources_C17/Lab1Files.zip)

You should now have a folder containing the four files **intarray.c**, **intarray.h**, **sin-wave.c**, and **makefile**.



Start *Eclipse* in your virtual machine by clicking on the  icon in the taskbar on the left of the virtual machine desktop. Alternatively, you may simply type the command **eclipse** to a command shell in the virtual machine.<sup>2</sup>

If you have never used *Eclipse* on a particular computer or virtual machine, it may ask you where you want to keep your *Workspace*. Your *workspace* is a directory where *Eclipse* stores files and information regarding your *Eclipse* projects.

A reasonable place to store your *Eclipse* workspace for projects and labs in this course is on a *flash drive*, so that your work does not get stuck inside of a malfunctioning virtual machine. Alternatively, you can store it on the desktop of your virtual machine and periodically export source files to offline storage for backup. If you already have an *Eclipse* workspace on an existing computer or elsewhere, you may simply add the projects of this lab to it.

**Note:** Workspaces for *Eclipse CDT* are machine and platform specific. This is because the workspace stores path names and environment information about where to find files, tools, and information *on that specific computer*. It is awkward and confusing to try to share a workspace between platforms such as your virtual machine as the CCC Linux systems.

Instead, you should copy, “export,” and/or “import” sources files from one workspace to another. (Exporting is discussed below.)

Once you have identified your workspace, *Eclipse* continues to load. When it finishes, you will be presented with the *Welcome Window* as shown in Figure 1 below.

- Clicking on the **Workbench** icon takes you to your *Eclipse* workspace. Once in your workspace, you can return to the *Welcome Window* by click **Welcome** in the **Help** menu of any *Eclipse* view.
- The other four icons provide an *Overview* of *Eclipse*, tutorials, and a few code samples. You may explore these on your own time.

---

<sup>2</sup> Remember from **SettingUpYourVirtualMachine.docx** that you can open a command shell by typing **CTRL-ALT-T** to the desktop.

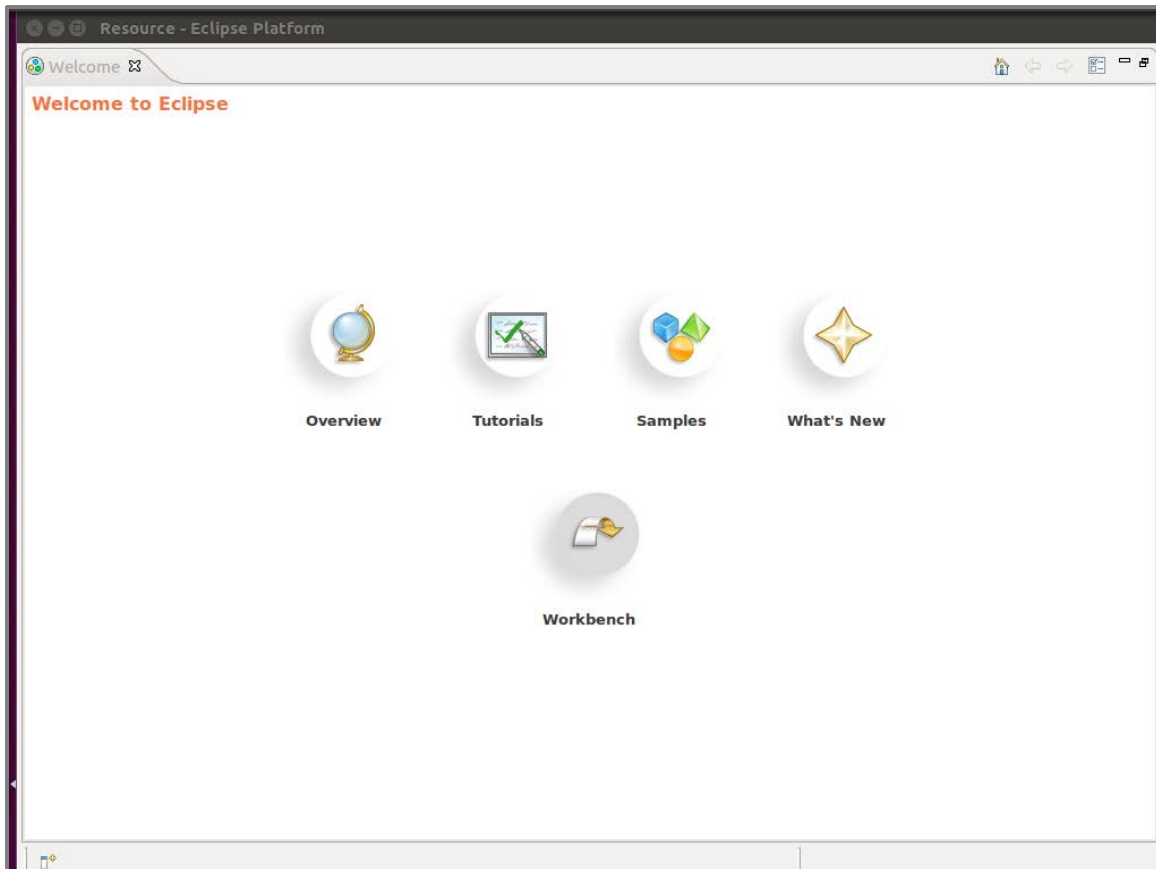


Figure 1

For this lab, you need to switch to the *CDT perspective* by doing the following:–

- Move the cursor to the top of the Ubuntu desktop; this causes the application menu bar to display itself. Open the **Window** menu, hover over **Open Perspective**, and click **Other**.

Note that whenever you need a command in the Ubuntu graphical user interface of your virtual machine, you can usually find a way to get to it via the menu bar, simply by moving the cursor to the top of the screen.

- Select **C/C++** from the list and click OK.

## Creating a new Project

In the *C/C++ perspective*, you can create new *C* or *C++ Projects*. There are several ways to create new *projects* in *Eclipse*. In this Laboratory assignment, we will create a project from existing code and an existing makefile. In future assignments in this course, we will use a different approach to create a new project from scratch.

To create the new project for this laboratory assignment:–

- Open the **File** menu, hover over **New**, and click **Project**. A **New Project** dialog appears. Click on the arrow new to the item **C/C++** to expand it, as shown in Figure 2 below.

Select **Makefile Project with Existing Code** and click **Next**.

- As an alternative, you may select **Makefile Project with Existing Code** direct from the **File > New** menu.

Either way, you will be presented with the dialog of Figure 3.

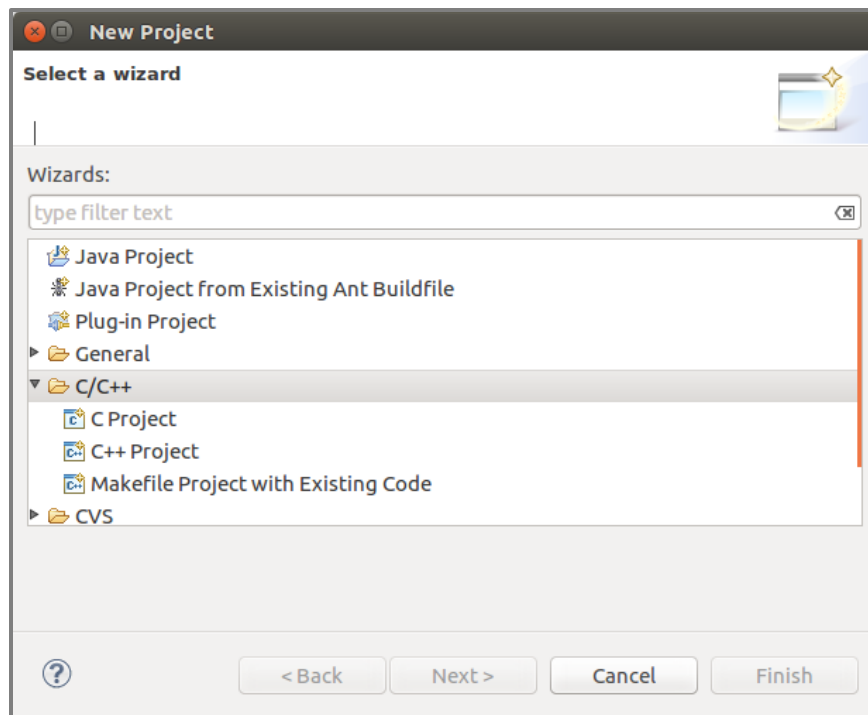


Figure 2

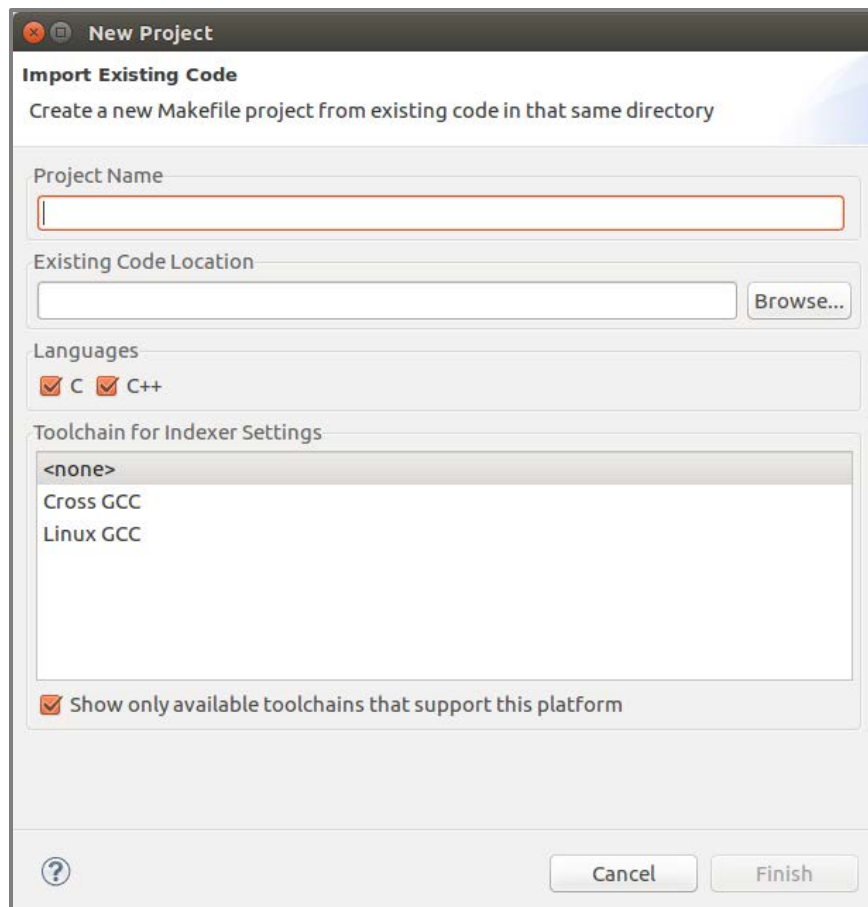


Figure 3

In the window of Figure 3, enter a name for your project in the box labeled *Project Name*.

**Note:** In this course, projects *must* be named **LabN\_username** and **PAn\_username**, where **username** is replaced by your WPI login name *or* your team name, in the case of team projects.

This requirement is for the benefit of the graders, so that they can import and grade a group of projects at one time. Eclipse does not support multiple projects with the same name. *Failure to adhere to this rule is a non-appealable penalty worth 10% of the project or lab grade.*

Next, click on the **Browse** button next to the box labeled *Existing Code Location* to find the folder containing the code to be imported to the project — in this case, the **Lab1Files** folder that you unzipped on page 2 of this document. For the *Toolchain* setting, select **Linux GCC**. Finally, click **Finish**.

The result should be a new project listed in the Project Explorer pane on the left side of the *Eclipse* workspace window, such as shown in Figure 4.

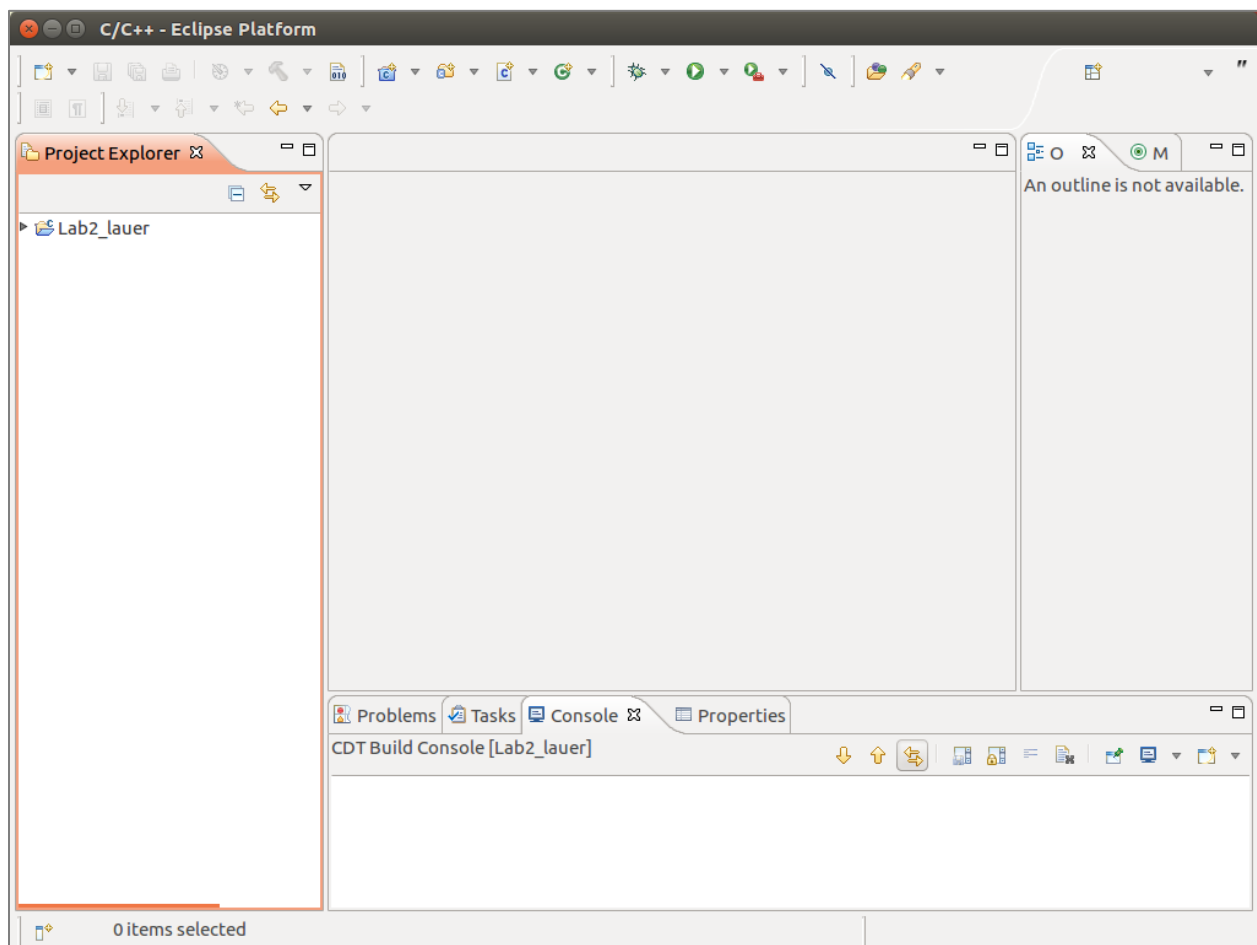


Figure 4

This is the *C/C++ Perspective* of *Eclipse*. The word *perspective* is used in *Eclipse* to mean a particular arrangement of tabs and tab groups adapted to a particular programming language and set of software development tools and tasks. There are four tab groups in the *C/C++ Perspective*:—

- The *Project Explorer* is the vertical panel to on the left. It enumerates the projects in the *Workspace*. Each project can be expanded to show its individual files by clicking on the tiny triangle to its left. In Figure 4, only one project is shown. By the end of the course, you will have many.
- The group of tabs in the center is the *Edit group*. When you open any file for editing, it shows up here. Double-click on each of the four files that you just brought into your project in order to open them. Figure 5 shows an example with file tabs for **intarray.c**, **intarray.h** and **sinewave.c**.

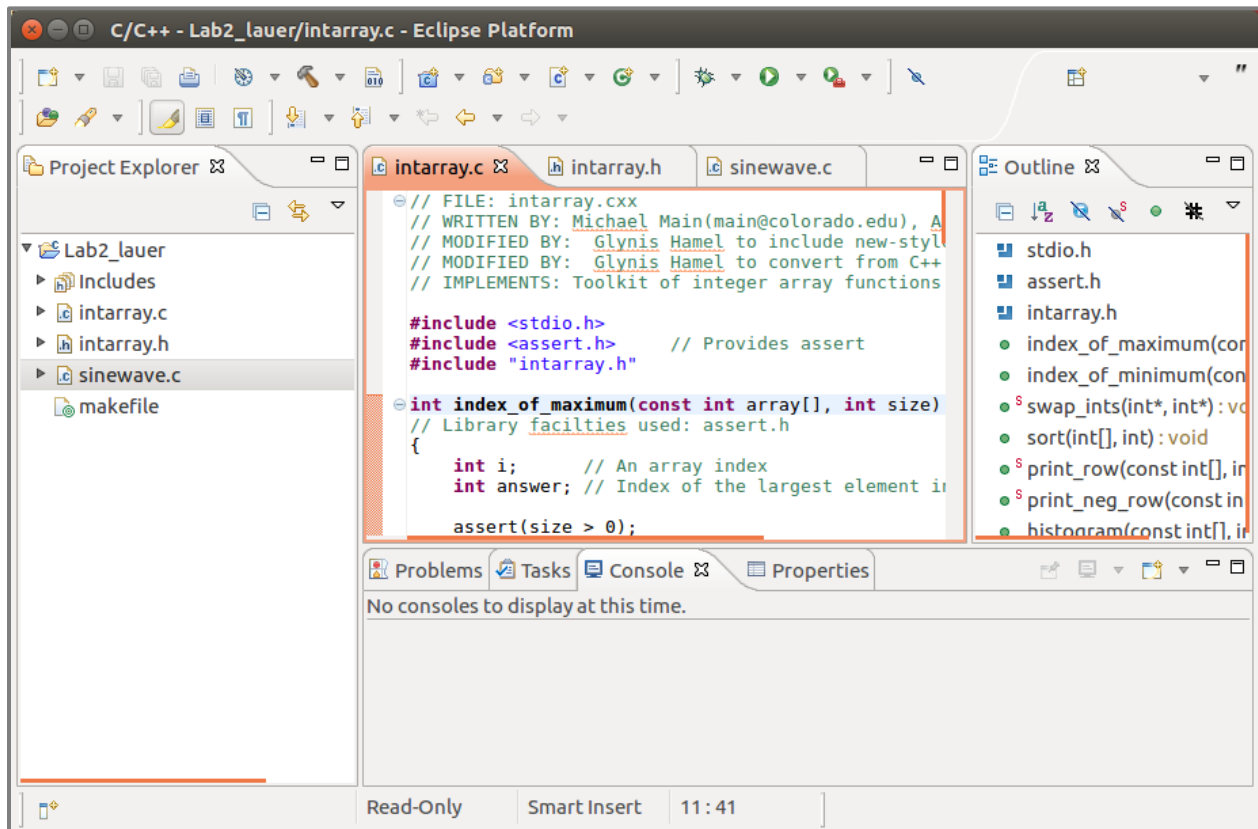


Figure 5

- The group of tabs on the right is the *Outline group*. The *Outline* tab changes, depending upon which file of the *Edit group* is open. For the open file, it shows a summary of the contents of that file. For example, for a **.c** or **.cpp** file, it will list the header files included by your file, along with the global variables and functions defined in it. Click on the name of a particular function, variable, or include file, and it will take you directly to that item in your file.
- The group of tabs along the bottom is the *Console group*. This is expanded in Figure 6 in the next section. *Eclipse* sends the output and error messages of compilations, program executions, and other activities to this group.

## Getting Credit for this Lab (part 1)

Edit one of the files in Figure 5 — for example **intarray.c** — by adding your name and the date to the list of comments at the top. Make sure the edited file shows in the *Eclipse* window.

Next, capture a *screenshot* of the *Eclipse* window. To do this, open (or go to) a Command Shell in your virtual machine and execute the command

```
gnome-screenshot -w -d 5
```

Within five seconds of typing return at the end of the command, click in the *Eclipse* window. The **-w** switch of the **screenshot** program means “take a picture of a window” (instead of the entire desktop). The **-d 5** switch means “wait five seconds to allow the user time to navigate to and/or click in the desired window.” When the picture has been taken, the **screenshot** program will present a dialog asking you to save it. Click **Save**. The image will be saved as a **.png** file. Submit this picture to *Canvas* under the assignment *Lab2*.

For more information about **gnome-screenshot**, consult its **man** page in a command shell.

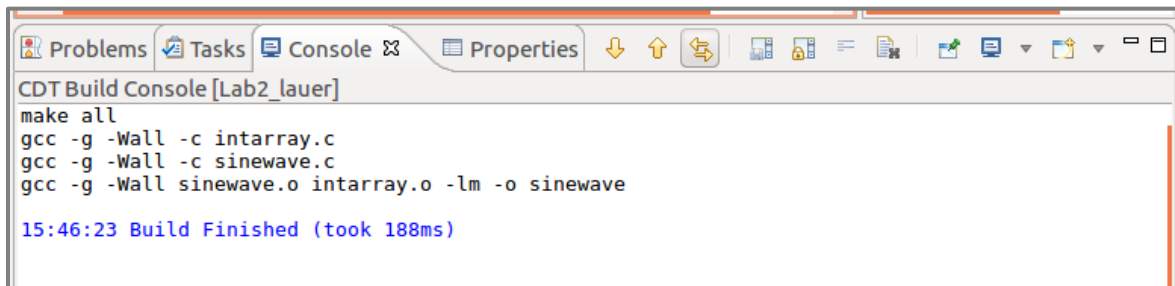
## Building the project

To build the project,

- Right-click on the project in *Project Explorer*
- Click **Build Project**

Alternatively, you can select **Build Project** under the *Project* menu of the top menu bar.

If you open the *Console* tab in the bottom window of the *C/C++ Perspective*, you should see that your project compiled successfully, as shown below:–



```
CDT Build Console [Lab2_lauer]
make all
gcc -g -Wall -c intarray.c
gcc -g -Wall -c sinewave.c
gcc -g -Wall sinewave.o intarray.o -lm -o sinewave
15:46:23 Build Finished (took 188ms)
```

Figure 6

To run the project,

- Right-click on the project in *Project Explorer* and hover over **Run As**
- Click **Local C/C++ Application** and click OK.

Assuming everything went well, your program should run and you should see the output of the **sinewave** program on the *Console* tab. You can expand the *Console* tab by double-clicking on the tab itself. The *Console group* will expand to fill the entire *Eclipse* window, as shown below. Double-click again on the tab, and the *Console group* will shrink back to its original size and place.

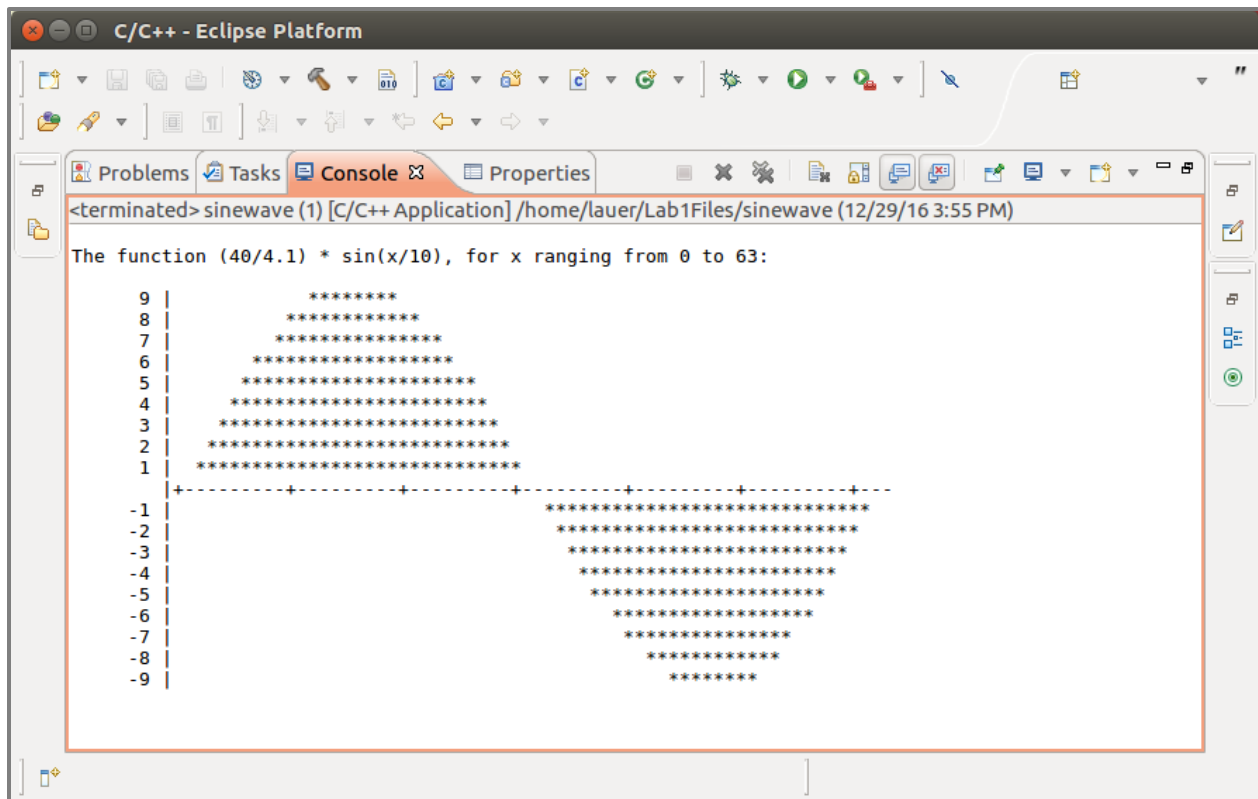

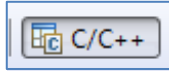


Figure 7

It is possible that you don't see the image of Figure 7. That is because *Eclipse* has more than one *Console* window — in fact, it has a lot of them. Notice the sequence of icons at the top right of the *Console* tab. Click in the *Console* window (whatever it shows) and hover your cursor over each icon until it displays a legend telling what it is. The two at the far right of Figure 7 have associated pull-down menus. The second from the right — resembling  — contains the list of active consoles. Select from this list to show your output. The icons to the left of this allow you to pin the console and/or force certain consoles to be displayed when one of them changes.

## Navigating, viewing, and editing your code

Return to the main *C/C++ perspective*. (If you cannot see the *C/C++ perspective*, you can reach it by selecting *Window > Open Perspective* or by clicking on the button  in a pulldown menu accessed by a tiny down-arrow at the upper right of the *Eclipse* window.) In the *Project Explorer*, open the project folder for your project and double-click on each of the source files in turn — i.e., **intarray.h**, **intarray.c**, and **sinewave.c**. This should open a new tab for each file.

In the first function of **intarray.c** — i.e., **index\_of\_maximum** — select **answer** in the declaration of **int answer**. Notice that all occurrences of the symbol **answer** in the function are highlighted. Likewise, select the parameter **size** of the same function and see that all of its occurrences are highlighted. Next, in the body of that function, allow the cursor to hover over the symbol **array** inside the **for**-loop. Notice that the declaration of the symbol appears. Finally, in the file



**sinewave.c**, hover the cursor over an instance of **printf()**. Notice that information is provided about this function, along with the header file in which it is declared.<sup>3</sup>

In **sinewave.c**, hover over the call to **histogram()** at the end of **main()**. *Eclipse* brings up a popup box showing the first few lines of code of the **histogram** function as it is declared in another **.c** file. At the bottom of this popup box is the instruction to “press F2 for focus.” Do this now; you will see a scrollable dialog that shows you the entire code of the **histogram** function. This is very handy when you are working on a multi-file project and want to refer to some code in another file.

### *Compilation errors*

Make a deliberate error in the **main** function by deleting the semicolon after the call to **histogram**. Save **sinewave.c** and close its tab. Now, rebuild the project using the **Project > Build Project** command as above. Note that the error is reported in the **Console** tab of the bottom window.

Double-click on the first error line (you may have to scroll up to find it). *Eclipse* automatically opens the file that was in error and takes you to the reported error line.

**Remember** that *C* compilers typically report errors one or more lines *after* an error. *Eclipse* takes you to the line where the compiler first realizes that something is wrong. It is your responsibility to work backwards in your code to find the actual problem. Fix the error and rebuild the project.

### *Exploring Eclipse*

Spend some more time exploring *Eclipse*:-

- Type some code and see how the editor helps you with indentation.
- Insert a comment using **/\*** and **\*/** to see how the editor displays it.
- Make deliberate errors by “forgetting” to terminate a comment and by “forgetting” to close a string. See what *Eclipse* does to help you.
- Insert a call to **printf**. Note that when you type one **(** character, *Eclipse* adds a matching one and positions the cursor in between. When you type double quote to begin a string, *Eclipse* adds a matching close quote and positions the cursor in between. It does the same with curly brackets and square brackets.
- Open the file **intarray.c**. You will see several functions — **index\_of\_maximum**, **index\_of\_minimum**, **swap\_ints**, etc. Next to the header of each one is a small circle with a minus sign inside it. Click on this circle; you will see that *Eclipse* collapses the function so that it does not take up space on the screen. The minus sign is now replaced by a plus sign, which you can use to expand it again. Collapse all of the functions (and anything else that you can find), and then expand them. Note that you do not have to expand functions before building the project.
- Finally, in the **Project Explorer**, after you have built a project, click on the arrow next to the name of a source file. This tells you much more about that file, such as what functions it defines and what functions that it calls.

Play around with the program editor and the **Project Explorer** to learn more about what they can do. There are many other tricks documented under the *C/C++ Development Guide* that can be found

---

<sup>3</sup> Learning how to annotate your code so that the same information appears for your own functions is beyond the scope of this course.

under the *Help > Help Contents* menu. There are also many, many on-line resources about how to use *Eclipse* in particular settings.

## Exporting your Project

To submit your project or to easily transfer it to another computer, you should *export* the project as a zip archive from *Eclipse*. To do this,

- **Clean** the project first. Do this before anything else, so that you don't submit a lot of temporary and/or object files. Right click on the project name in the *Project Explorer* and select **Clean Project**. Alternatively, select **Clean** from the *Project* menu.
- Right-click on the Project and click **Export**. Under the **General** category, choose the **Archive File** option and click **Next**. In the panel on at the top right of the next window, you can select what you want to export. Lower in the window, select **Browse** and choose where you would like to save the project archive.
- In the **Options** area at the bottom, select "Save in *zip* format." Finally, click **Finish** to export the Project.<sup>4</sup>

**Note:** There is a penalty of 10% of the value of the project for submitting "unclean" projects — i.e., projects containing compiled files, object files, and other intermediate files resulting from building.

You should now have a zip file suitable for submitting to *Canvas* or for porting to another environment. If you open the zip file, you should find updated copies of the three original source files, your **makefile** (with any modifications that your or *Eclipse* might have made), plus several other files and/or folders with names that begin with a dot. These are *Eclipse's* configuration files for this project.

To run the program on another platform outside of *Eclipse*, unzip the archive file and run the **make** command to build on that platform. You can then run the program there. (The graders may or may not use this method when grading your projects during the rest of this course.) To import it into another *Eclipse* workspace, simply invoke **Import...** from the *File* menu.

## To get credit for this lab

In addition to submitting the screenshot on page 6 for credit for this lab, you must also submit your archive file to the *Canvas* system for the course **CS2303-C17-LABS**. This project is *Lab2 -- Eclipse* under the *Laboratory Assignments* tab of the *Assignments* page. Be sure to complete this step before midnight on the day of your lab session.

If you have extra time, port your implementation of Programming Assignment 2 (due on Saturday) to *Eclipse*. Then proceed the next section.

## Providing input and arguments to your program

Most programs need some sort of input or some arguments on the command line. These can be provided in the project properties.

- Right-click on the project name and select *Properties*.

---

<sup>4</sup> Note that the Professor's computer is **tar**-file challenged, **rar**-file challenged, and challenged in all other types of archive files. Only *zip* files will be accepted.

- Select *Run/Debug settings* in the left panel, select the name of the executable file in the middle panel, and click the *Edit* button on the right. This should bring up a dialog resembling the following:–

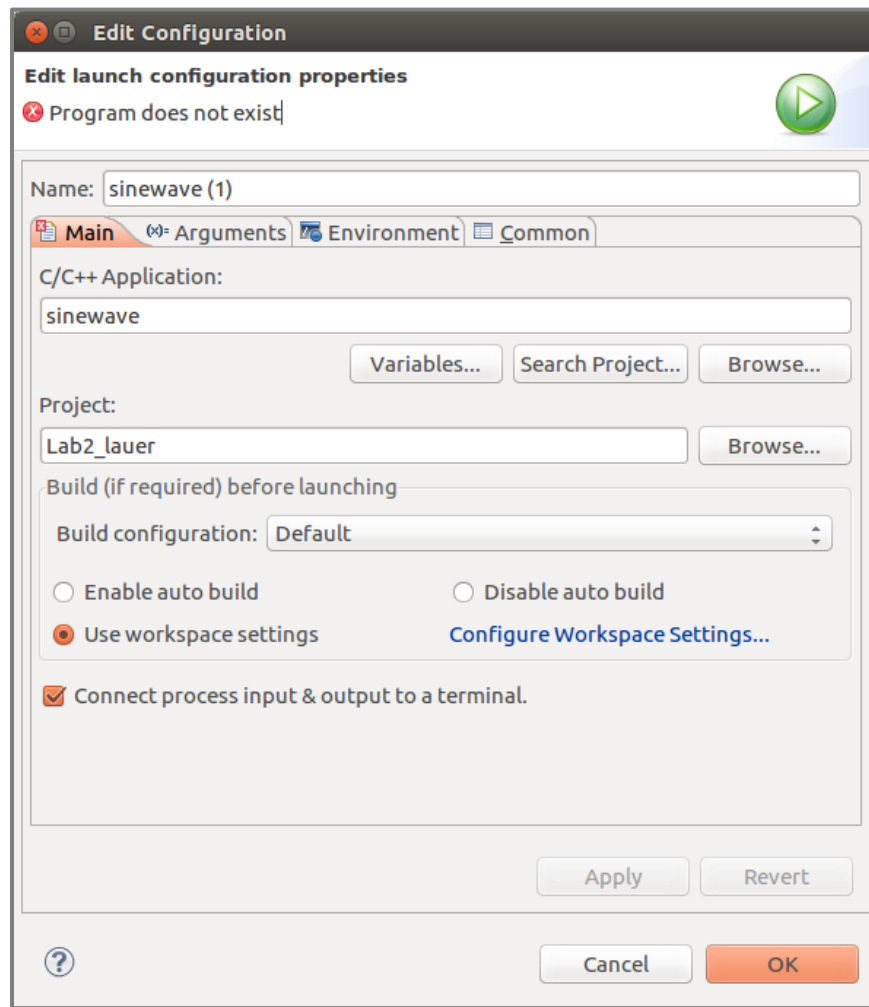


Figure 8

- Click the *Arguments* tab and enter the arguments, file redirections, and pipes exactly as you would have entered them on a command line. Do not, however, include the name of the program, because that will be provided automatically by *Eclipse*.

Command line arguments are required for Programming Assignments #2 and #3.

## Viewing more than one source file at the same time

Frequently, you will find that you need to have more than one source file visible at the same time. For example, you may want to view *both* a **.c** or **.cpp** file and its associated **.h** file side-by-side. Normally, *Eclipse* shows only one tab of a group at one time. However, you can “pull apart” a group by dragging one of its tabs to the side or down.

Try this now with **intarray.h** in Figure 5. Point the cursor to the **intarray.h** tab, press the left mouse button and drag to the right edge or bottom edge of the group. Eventually a green “split” line will appear. Depending upon where you drag it, the green line will split the tab horizontally or

vertically, giving you two sets of tabs in the *Edit group*, as exemplified in Figure 9.

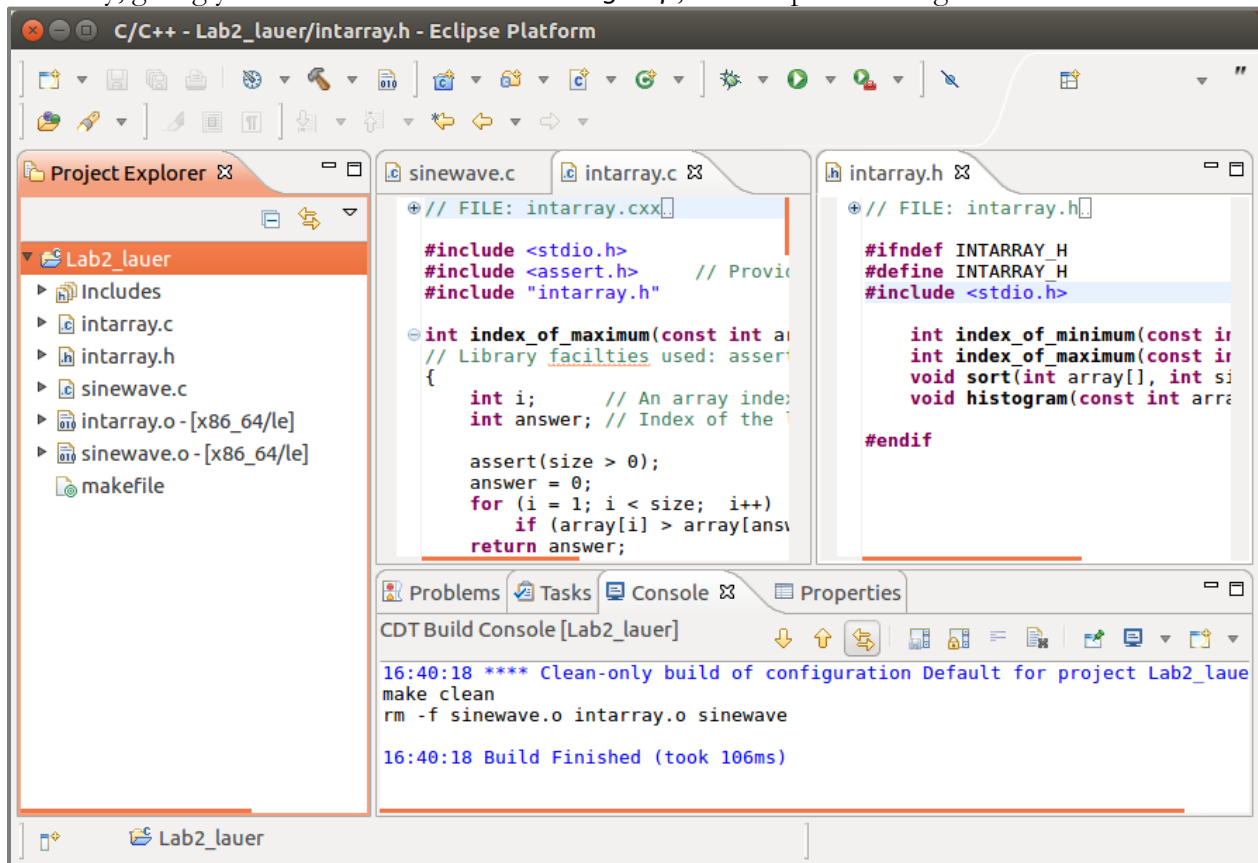


Figure 9

You can restore the tab to its original position by dragging back to its group.

## Online Help

In the *Help* menu of *Eclipse*, select *Help Contents*. This will open a new window in your virtual machine desktop and will provide access to an enormous amount of help and documentation. Included are both a *Workbench User Guide* and a *C/C++ Development User Guide*. Both documents are comprehensive and well-written. They provide access to far more than we can cover in the time of one laboratory session.

## Using Eclipse on other systems

*Eclipse CDT* is installed on the WPI CCC Linux system. This is easily accessible from **PuTTY** or your **SSH** client, provided that you have an *X-window* server running on your own computer. In a command shell, type

```
eclipse &
```

This will bring up *Eclipse* in a new window on your desktop. Everything works the same as the version on Windows or Macintosh personal computers, but the compilation is done by the *GNU gcc* compiler and debugging tools.

While *Eclipse* on the CCC system works well functionally, it is not nearly so responsive as *Eclipse* running locally on your own or a public computer.