CS 201
Eclipse CDT Tutorial

Eclipse is a framework for IDEs. It provides the infrastructure for the IDE, and then the IDE is completed by using a plug-in for a particular programming language. Plug-ins exist for Java (which was its first application), C/C++, Python, Perl and many others. **Eclipse** is multi-platform, runnable on Linux, Windows , etc. **Eclipse** came out of the Websphere project at IBM. Though the **Eclipse** project is now fully open-source, it is still strongly supported by IBM.

In this tutorial, we will describe in step by step how to edit, compile and execute a C project using Eclipse in your EPS 254 lab computer.

Step 1: Staring Eclipse

Log in to the Linux side of your machine. Start up Eclipse by typing eclipse in a command window. A `Workspace Launcher` window will appear as shown in Fig.1 which will ask you to specify a directory to use. Choose your working directory and select OK.

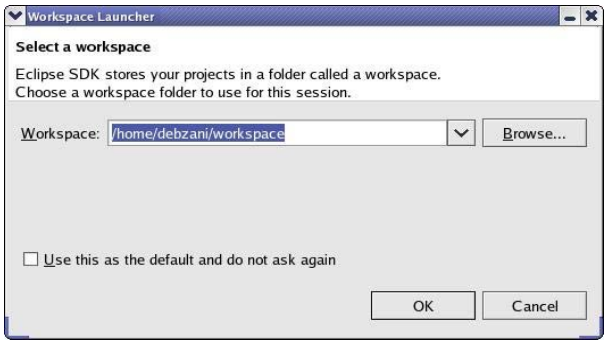

Fig.1 Workspace Launcher

Step 2: Eclipse Perspectives and views

Eclipse is launched and your screen should look like Fig.2 . The main kinds of components in **Eclipse** are:

- perspectives
- views

The current perspective is the environment under which your **Eclipse** screen is currently operating. The current perspective consists of a number of subwindows, called *views*. For example, currently we are in C/C++ perspective as shown in the snapshot of Fig. 2. If you are not in C/C++ perspective, you can go there by choosing **Window | Open perspective | Other | C/C++** and then pressing OK. Take some time to familiarize with different views available in C/C++ perspective such as listing your various C/C++ projects (and files within those projects), an editor view for each file you currently are editing, an outline view showing each of the major components of your code (functions, global variables, etc.), a **Console** view in which standard input/output would be done, and so on. During your work on your C/C++ code, you will use the debugger, in which case you will temporarily switch to the Debug perspective.

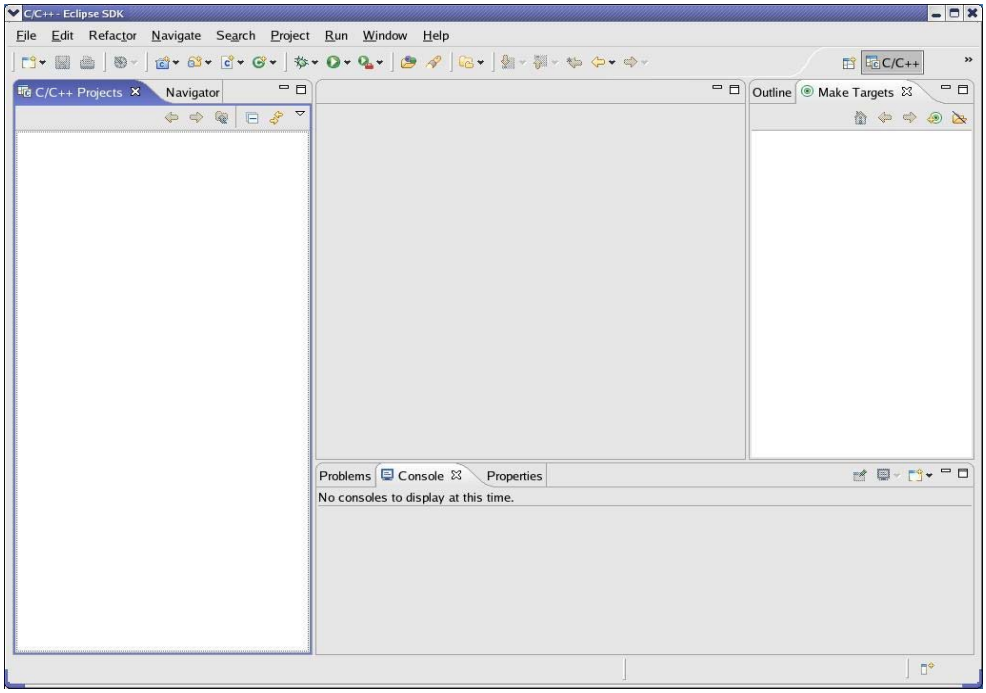

Fig.2 Eclipse perspective and views

Step 3: Creating a C project

As with many IDEs, Eclipse organizes things into *projects*. A project is a directory within your workspace. To make a new project:

- Select **File | New | Project**
- Select **Standard Make C Project** as we are creating a C project and will be writing our own **makefile.** If you want Eclipse to generate the **makefile f**or you, you can select **Managed Make C project.**
- A **New Project** dialog box will appear. Enter your project name (in my case, I entered **helloworld** as project name) and hit **Finish** unless you know the advanced stuff.
- If you are not already in the C/C++ perspective, it will ask you if you want to go to that perspective; say Yes

To go to a project, either previously existing or one you just created: You can double-click the project name in the Navigator view or the C/C++ view, etc. If you don't see that view, select **Window | Show View**. You may have to right-click on the project name, and select **Open Project**. At this stage, the screen will look like Fig. 3 with the newly created project **helloworld.**

Step 4: Creating/Editing C Source Files

To make a new file, say **hello.c** under the project **helloworld**

- right-click on the project name, then select **New | Source File**
- make sure that the project name in **Source Folder** is **helloworld**; enter the file name, **hello.c** ; hit **Finish**
- the editor will now open a subview for editing **hello.c**

To edit an existing file:

- go to **Navigator** (or **C/C++ Projects**, if doing C/C++)
- if the files don't appear, click on the arrow next to the project name
- double-click on the file you want

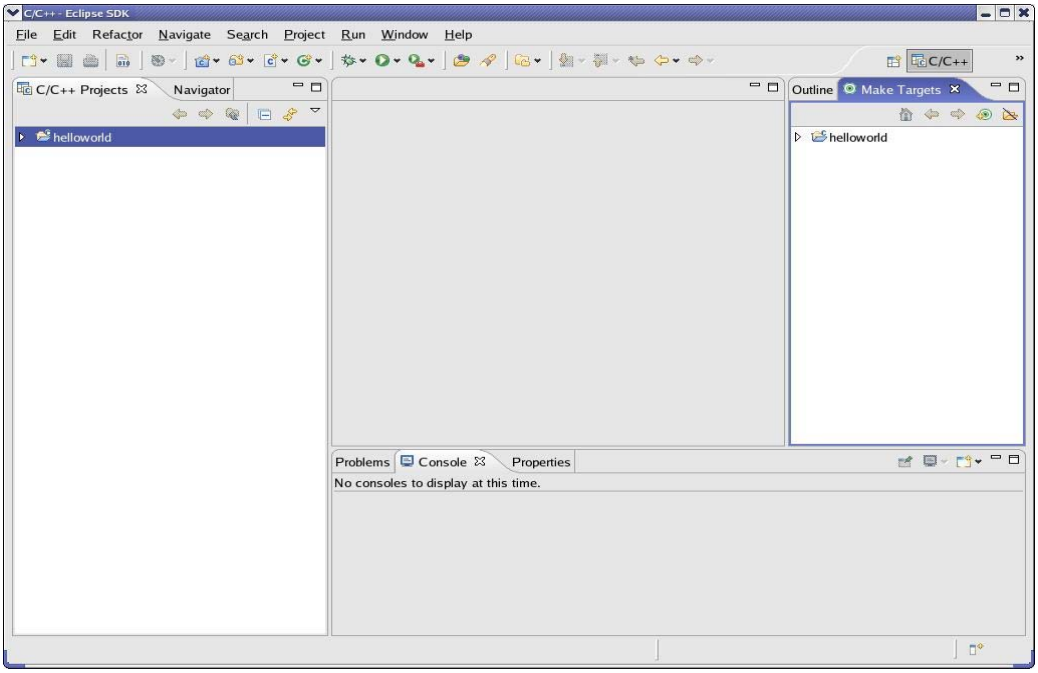

Fig.3 Project helloworld has been created

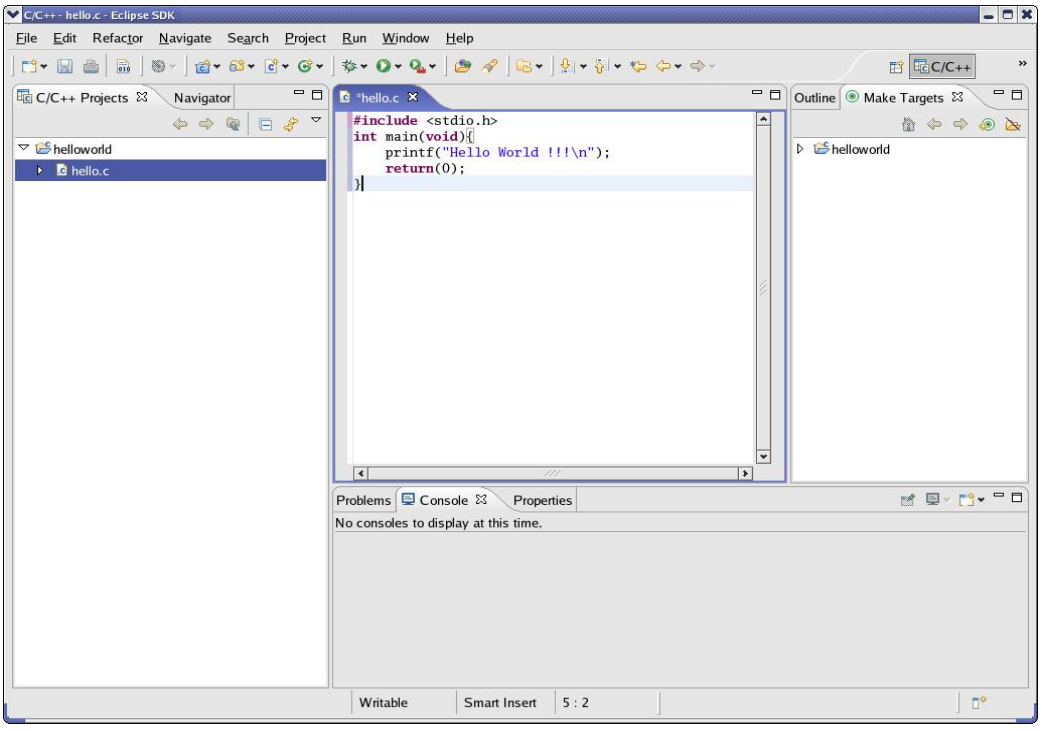

Fig.4 Adding code to hello.c

Step 5: Adding Source Code

Add the source code to **hello.c** as shown in Fig.4 and save it.

Step 6: Creating makefile

You have to create a **makefile** for each project. To create a **makefile,** right click on the appropriate project and select **New | File** (remember makefile is an ordinary file, not a source file). In the editor, create your **makefile** as follows. Be aware that your makefile must have entries **all:** for successful compilation. Please visit the **Some Useful Links** in your course web page to know more about makefiles.

```
all: hello

hello: hello.o
	gcc -o hello hello.o

hello.o: hello.c
	gcc -c hello.c
```

of course, there are TABs in there; don't forget to save the file.

Step 6: Compiling your project

To compile, save your files if needed, right-click the project name, then select **Build Project**, which will run **make.** If there were any compilation errors, they'll show up in several places:

- In the **Console** view, which will show the result of the make
- In the **Problems** view; if you click on any problem, the mouse pointer will be moved to the offending line in your editor view

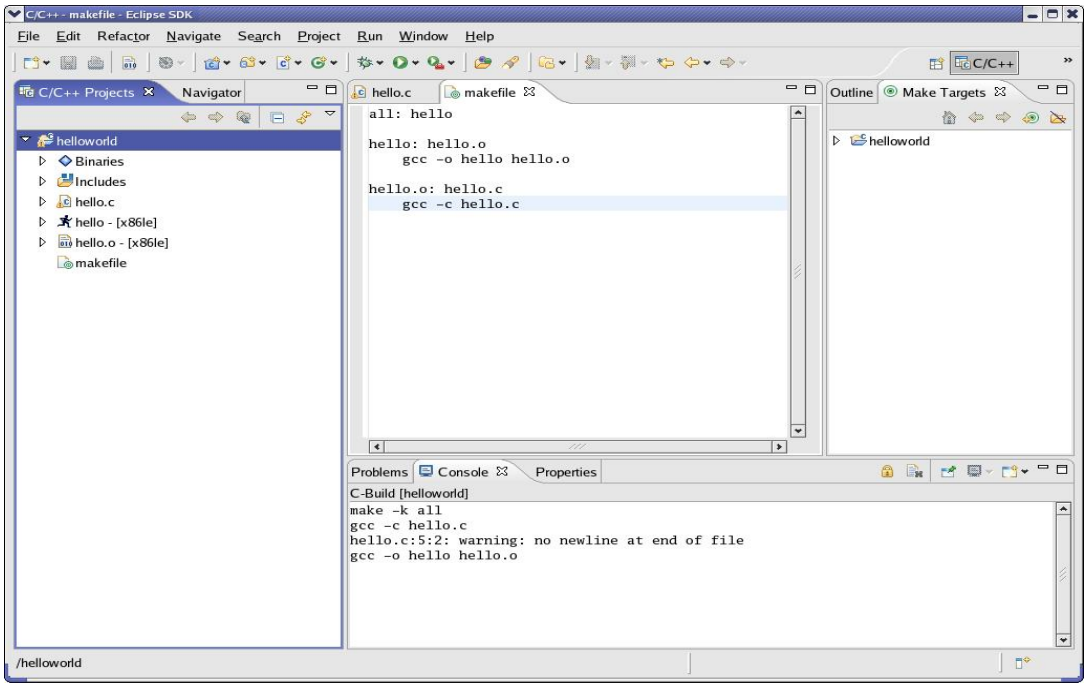

Fig.5 After compilation

- In the editor view itself, where the compilation errors will be flagged with red disks just to the left of source code lines in your editor subviews; if you move the mouse pointer to one of the red disks, a yellow window will pop up to tell you what errors were found at that line

Once compiled successfully, other related files will be created under project **helloworld** as shown in Fig. 5. Don't worry about the “warning: no newline at end of file” appears in the console window.

Step 6: Run your project

If there were no compiler errors, run your code:

- Select **Run | Run...**; you will be asked for a run configuration
- If you already have one for this program, click on it in the **Configurations:** view.
- If you don't have one, click on (C/C++ local application) and then on **New**; choose a configuration name, and fill in your project and executable file names (on C/C++ application area, click on the **Search Project** and it should show the executable's name. Fig. 6 shows a new configuration **helloworld** with project name and executables filled up properly.
- Hit **Run**
- Your program's output will appear in the **Console** view as shown in Fig.7.

Note that once you have your run configuration set up, you can bypass a step in future runs by selecting **Run | Run Last Launched.**

- If there were compiler errors, you can go to the error line either via using the mouse pointer in the editor, or by clicking on the error message in the **Problems** view
- to invoke the debugger, right-click on the project name, and select **Debug | Debug** highlight your executable name on the left, set any arguments, and hit **Debug**; you'll be asked whether you want to switch to the **Debug** perspective--say yes.
- to go back to edit/compile/run mode, go back to the **C/C++** perspective

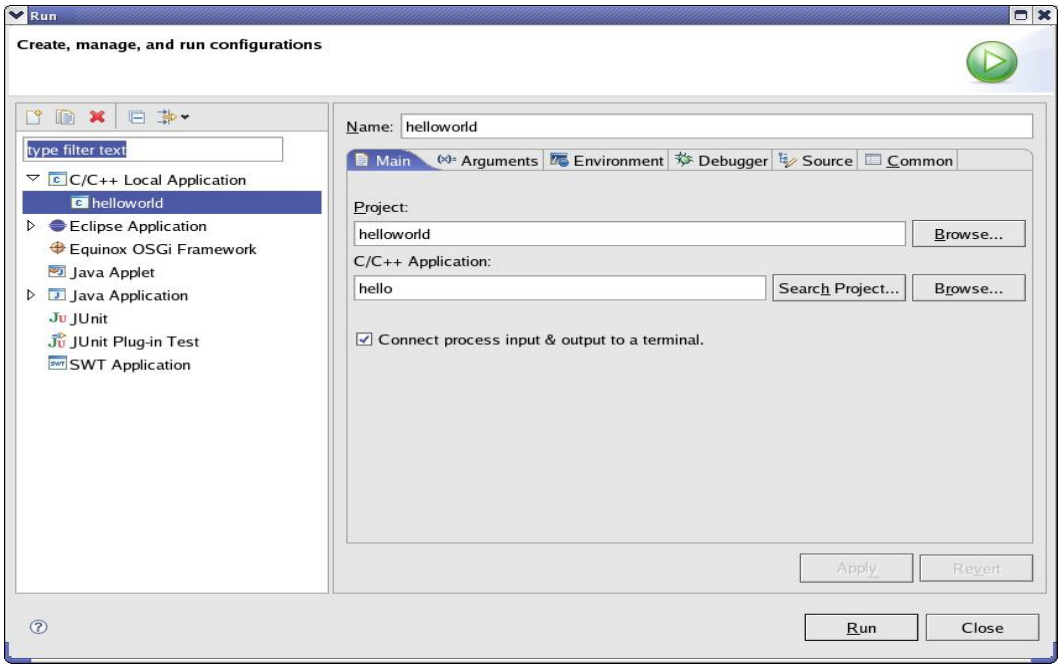

Fig. 6 Run configuration

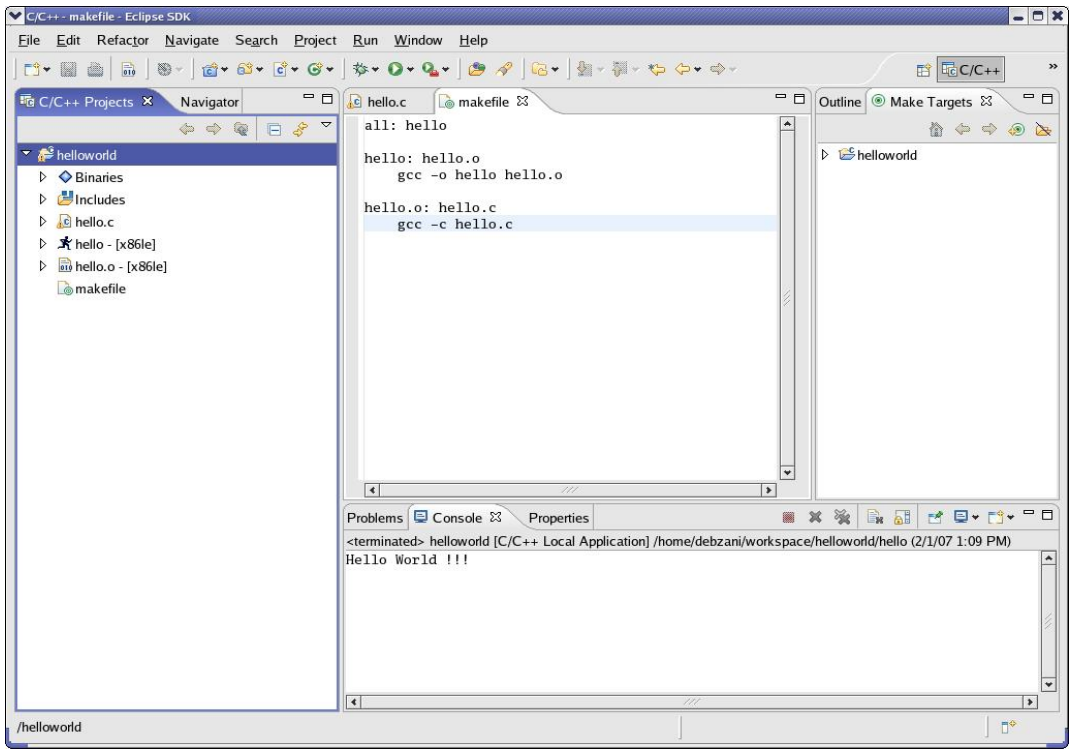

Fig. 7 Execution of hello