# Building C/C++ projects

The CDT relies on an external make utility, such as GNU make, to build a project. The CDT can generate makefiles automatically when you create a Managed Make C project or a Managed Make C++ project. You have the option of creating a Standard Make C project or a Standard Make C++ project and providing the makefile yourself.

## Required utilities

You must install and configure the following utilities:

* Build (e.g. make).
* Compile (e.g. gcc).
* Debug (e.g. gdb).

**Note:**  while make, gcc and gdb are the examples used in the documentation, virtually any similar set of tools or utilities could be used.

**Tip:** Cygwin contains these utilities (make, gcc and gdb) for a Windows environment.  While running the cygwin installation, ensure gcc and make are selected since they are not installed by default. For more information, see <http://www.cygwin.com>. If you are a Red Hat user, all that you need to do to build your project is included in the Red Hat Linux installation. For other operating systems, please refer to your installation documentation.

## Build terminology

The CDT uses a number of terms to describe the scope of the build.

### Build Project

This is an incremental build (make all, assuming all is defined in your makefile). Only the components affected by modified files in that particular project are built.

### Rebuild Project

Builds every file in the project whether or not a file has been modified since the last build. A rebuild is a clean followed by a build.

For more information on builds, see:

* **Workbench User Guide > Concepts > Workbench > Builds**
* **Workbench User Guide > Tasks > Building resources**

Build-related information is displayed as follows:

* The Console view displays the output of the build tools.
* The Problems view displays a list of compiler errors and warnings related to your projects.
* For Standard Make projects, the Makefile targets are displayed in the Make Targets view.

For more information about the Problems view, see **Workbench User Guide > Reference > User interface information > Views and editors > Problems view**.

## Getting a makefile

You can either create a C/C++ project for which you supply the **makefile** or create a C/C++ project for which the CDT generates makefiles automatically.

To create a new project, from the menu bar choose **File > New > Project**. In the dialog that appears, expand the C/C++ group and choose e.g. C Project

* In the resulting wizard page, to create a project for which you supply the **makefile**, select **Makefile project** and choose one of the alternatives under that. An empty project, or a simple "Hello World" can be created. You edit and manage the makefile yourself.
* To create a project for which the CDT supplies a basic **makefile**, select another project type, e.g. **Executable** and choose one of the examples under that, or choose **Empty Project**.

## Setting build preferences

You can set build preferences in Eclipse:

Build order If certain projects must be built before others, you can set the *build order*. If your project refers to another project, the CDT must build the other project first. To set the build order, from the menu bar select **Window > Preferences** and choose  **General > Preferences > Build Order**.

When you set the build order, the CDT does not rebuild projects that depend on a project; you must rebuild all projects to ensure all changes are propagated.

Automatic save You can set the CDT to perform an *automatic save* of all modified resources when you perform a manual build. In the preferences dialog, select **General > Workspace** and check **Save automatically before build**. By default, this feature is *not* enabled.

## Controlling the building of your project

For a Makefile project, the C/C++ compiler that a project uses is controlled by the project's **Properties** setting. To view a project's properties, right-click on the project and select **Properties**. In the dialog that appears, the **C/C++ Build** page enables you to control a variety of settings, including:

Build Command On the **Builder Settings** tab, this controls which make is used. To change it, uncheck **Use default build command** and change it or add arguments to the make command. Build Setting On the **Behaviour** tab, this controls whether the compiler will **Stop on first build error** or not (keep going). Unchecking **Stop on first build error** will force the compiler to attempt to build all referenced projects even if the current project has errors. Workbench Build Behavior On the **Behaviour**  tab, this controls which makefile target will be built depending on the scope of the build, e.g. all or clean.

For a standard (non-Makefile) project (often called "Managed Build" or "Managed Make" project from earlier CDT version), the project properties dialog enables you to manage the build configurations of your project. For additional information see:

* **Reference > C/C++ Properties > C/C++ Project Properties > Managed Make Projects**
* **Reference > C/C++ Properties > C/C++ Project Properties > Managed Make File Properties**

## Viewing build information

Build-related information is displayed as follows:

* The **Console** view displays the output of the make utility.
* The **Problems** view displays a list of compiler errors and warnings related to your projects.
* For a Standard Make project, build actions display in the **Make Targets** view.



[CDT Projects](http://docs.google.com/cdt_c_projects.htm)

[Project file views](http://docs.google.com/cdt_c_proj_file_views.htm)



[Building projects](http://docs.google.com/tasks/cdt_o_build_task.htm)



[Project Properties](http://docs.google.com/reference/cdt_u_properties.htm)



