EECS 211
Code::Blocks Notes

Jump to Installing, Creating a project, Tips and traps, the 211 Code::Blocks FAQ.

All you really need to build a C++ application is a compiler, a Makefile to document what needs to be compiled and how, and a text editor to edit the source code. But life is a lot easier with an integrated development environments (IDE). An IDE provides a convenient interface for editing, compiling, running, and debugging code. There are many IDEs. Most are quite similar to each other, and use similar terms, such as "project" (a collection of source code files that combine to make an application) and "workspace" (a collection of projects).

On MacOS X, I recommend using XCode. It comes on the CD-ROM that came with your Macintosh, and can also be downloaded from the Apple Developer site. It's also relatively simple to set up for makefiles.

On Windows and/or Linux, I recommend Code::Blocks. Code::Blocks is a free integrated development environment (IDE) for C and C++ on Windows, Linux and MacOS X.

Code::Blocks is actually a graphical front-end to the C/C++ compiler. By default, it uses the Gnu C compiler (GCC). GCC comes with Linux and MacOS X. On Windows, GCC is available via MinGW or Cygwin. We'll use Cygwin in this class because Cygwin also provides a Unix-like shell and a nice installer.

By default, Code::Blocks uses its own internal project management system, but in this class, we are going to use Code::Blocks with *custom makefiles*. This takes a little bit of extra care when creating projects, but pays off in portability.

These instructions are for Windows.

Requirements

You need Windows XP or Vista. If you have Vista, you will need an account with administrative privileges.

You need Cygwin with the GNU debugger and CppUnit. See these installation instructions for setting up Cygwin. Do this before installing Code::Blocks.

Download and test the example Makefile. You need the directory and files from that test to do these instructions.

Download and install Code::Blocks

Downloading and installing Code::Blocks takes just a few steps.

- Go to the Code::Blocks download page.

- Download the version that does **not** include MinGW. Currently this is `codeblocks-8.02-setup.exe`.
- After the installer downloads, run it and accept all the defaults for installation.
- When it offers to start Code::Blocks, let it.
- When Code::Blocks starts the first time, it looks for C/C++ compilers on your machine. It should detect at least one, If you see more than one, select the Cygwin GCC compiler.
- Code::Blocks will now open its "front page." This is where you can create new projects and re-open old projects.

This is a good time to quit and save your workspace so far. When you exit Code::Blocks, it will ask where you want save the workspace file. Put it in the top level of your projects directory. Give it a short name, like `eecs2 11`. The file will be saved with the extension `workspace`. Later on, just double-click this file to start Code::Blocks.

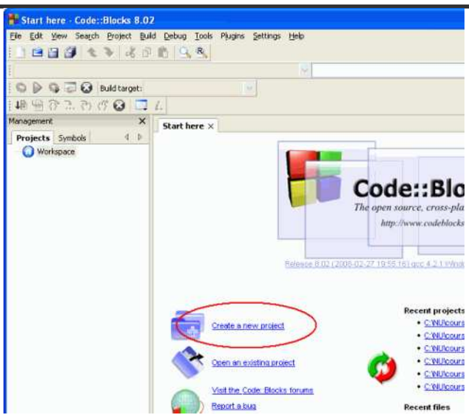
Creating a Console Project

A console project is a program that does all input and output through a console window, such as Window's Command Console or MacOS X's Terminal window. These are the simplest programs to write, but still useful, especially in the Unix world.

To create and test an example project using CppUnit,

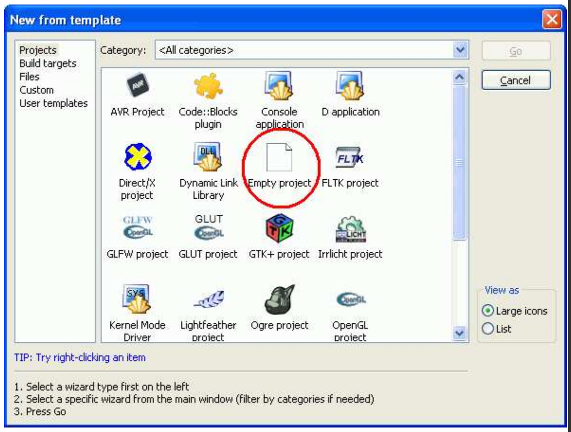
Start Code::Blocks by double-clicking the workspace file you created.

Code::Blocks will open its front page. On that page, click Create a new project.



A dialog box will appear with options for many different kinds of projects.

Click on `Empty project`and click Go.

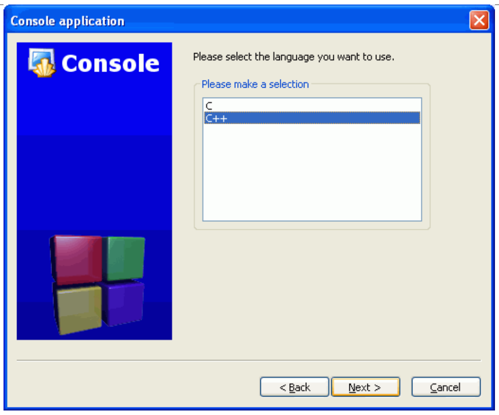


TIP: Try right-clicking an item

1. Select a wizard type first on the left
2. Select a specific wizard from the main window (filter by categories if needed)
3. Press Go

The project wizard will introduce itself. Click Next >.

The wizard will ask which programming language you want to use. Pick **C++** and click Next >.



Please select the language you want to use.

Please make a selection

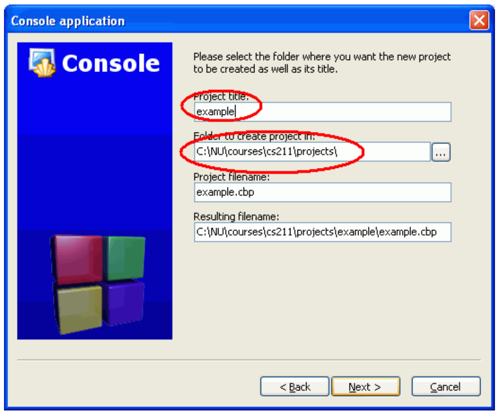
- C
- C++

The wizard will ask what you want to call your project and where you want to put it.

Click ... and use the file dialog box that appears to find and select your course projects directory.

Then enter the project name. **This must match the value of `PROJ` in the Makefile.** For the example project, therefore, use `example`.

Click Next > when done.



Please select the folder where you want the new project to be created as well as its title.

Project title: example

Folder to create project in: C:\NU\courses\cs211\projects\

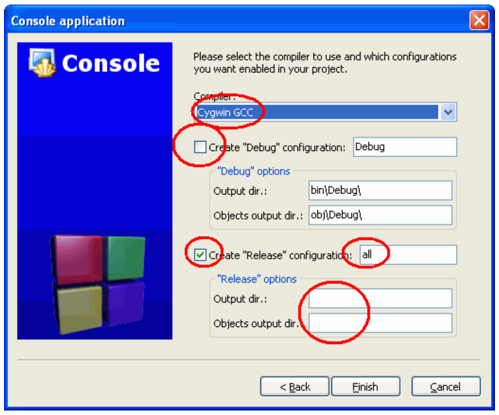
Project filename: example.cbp

Resulting filename: C:\NU\courses\cs211\projects\example\example.cbp

Code::Blocks will ask what targets you want to build. Since you'll be using a Makefile with a target called `all`,

- Uncheck `Create "Debug" Configuration`.
- Change the name for `Create "Release" Configuration` to `all`.
- Optionally, clear the output directory fields.

Click Finish when done.



Please select the compiler to use and which configurations you want enabled in your project.

Compiler: Cygwin GCC

☐ Create "Debug" configuration: Debug

"Debug" options

Output dir.: bin\Debug\

Objects output dir.: obj\Debug\

☑ Create "Release" configuration: all

"Release" options

Output dir.:

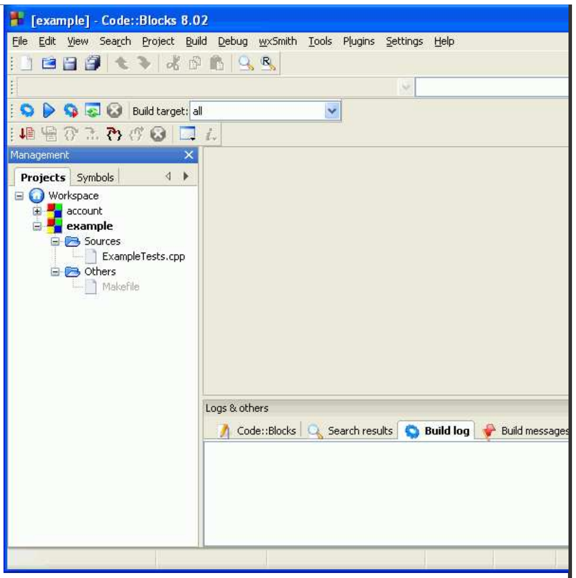
Objects output dir.:

Now tell Code::Blocks which files are part of your project.

Select **Add files...** from the **Project** menu.

Select the two source files in the `example` directory.

Afterward, click on the plus-signs in the project window, to see the files grouped under `Sources`, `Headers` and `Others`.

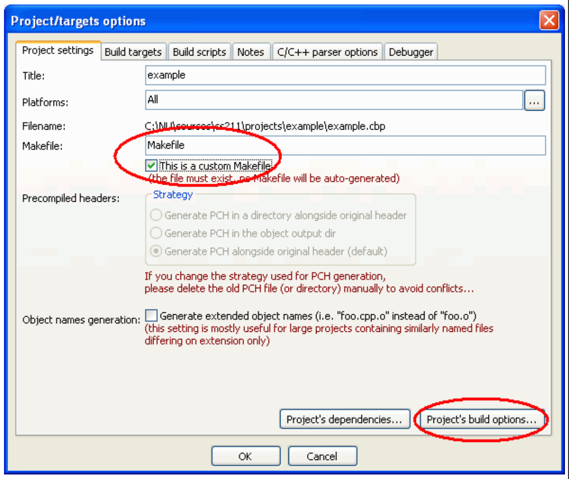

[example] - Code::Blocks 8.02
File Edit View Search Project Build Debug wxSmith Tools Plugins Settings Help
Build target: all
Management
Projects Symbols
Workspace
account
example
Sources
ExampleTests.cpp
Others
Makefile
Logs & others
Code::Blocks Search results Build log Build messages

Now tell Code::Blocks how to call the Makefile

Select **Properties...** from the **Project** menu.

Check the box for `This is a custom Makefile.`.

Click OK to save this setting.


Project/targets options
Project settings Build targets Build scripts Notes C/C++ parser options Debugger

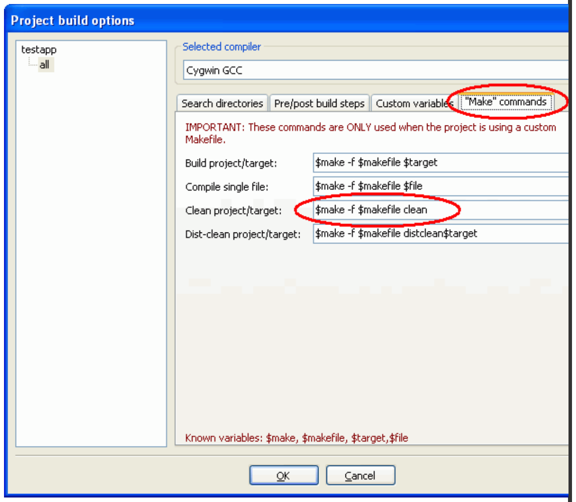
Title: example
Platforms: All
Filename: C:\NU\courses\cs211\projects\example\example.cbp
Makefile: Makefile
☑ This is a custom Makefile
(the file must exist, no Makefile will be auto-generated)
Precompiled headers: Strategy
○ Generate PCH in a directory alongside original header
○ Generate PCH in the object output dir
● Generate PCH alongside original header (default)
If you change the strategy used for PCH generation, please delete the old PCH file (or directory) manually to avoid conflicts...
Object names generation: ☐ Generate extended object names (i.e. "foo.cpp.o" instead of "foo.o")
(this setting is mostly useful for large projects containing similarly named files differing on extension only)

Select **Build Options...** from the **Project** menu.

Use the arrow buttons to scroll the tabs until you see the `"Make" commands`.

Select `"Make" commands` and change the field for `Clean project/target` to be `$make -f $makefile clean`.

Click `OK` twice.


Project build options
testapp
all
Search directories Pre/post build steps Custom variables "Make" commands
OK
Cancel

Selected compiler: Cygwin GCC
IMPORTANT: These commands are ONLY used when the project is using a custom Makefile.
Build project/target: $make -f $makefile $target
Compile single file: $make -f $makefile $file
Clean project/target: $make -f $makefile clean
Dist-clean project/target: $make -f $makefile distclean$target
Known variables: $make, $makefile, $target,$file

Now test your project.

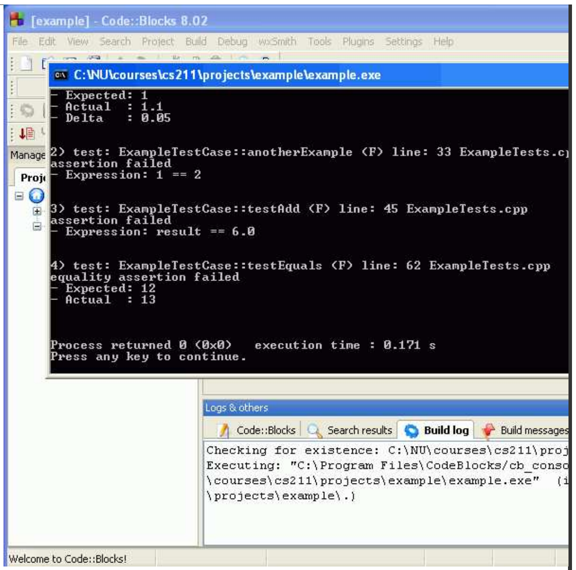
Select **Build and Run** from the **Build** menu.

If all is set up correctly, the project should compile your project, creating the application `example.exe`. It then runs the application in a console window.

The console output should look like the output you saw when you ran the executable directly in a console window.

To close the console window and return to your project, press any key while the console window is active

Exit Code::Blocks to save your work.


[example] - Code::Blocks 8.02
File Edit View Search Project Build Debug wxSmith Tools Plugins Settings Help
C:\NU\courses\cs211\projects\example\example.exe
Manage
Proj
Logs & others
Code::Blocks
Search results
Build log
Build messages
Welcome to Code::Blocks!

```
- Expected: 1
- Actual   : 1.1
- Delta    : 0.05


2) test: ExampleTestCase::anotherExample (F) line: 33 ExampleTests.c
assertion failed
- Expression: 1 == 2


3) test: ExampleTestCase::testAdd (F) line: 45 ExampleTests.cpp
assertion failed
- Expression: result == 6.0


4) test: ExampleTestCase::testEquals (F) line: 62 ExampleTests.cpp
equality assertion failed
- Expected: 12
- Actual   : 13


Process returned 0 (0x0)   execution time : 0.171 s
Press any key to continue.
```

```
Checking for existence: C:\NU\courses\cs211\proj
Executing: "C:\Program Files\CodeBlocks/cb_conso
\courses\cs211\projects\example\example.exe"  (i
\projects\example\.)
```

Tips and Traps

Learn the shorcuts. This is a general tip for any development tool. After you've used your IDE a bit, learn to use the keyboard shortcuts for compiling and running and so on. This will speed up coding and testing a lot.

Don't rename code files in the Finder. This is a general tip for any development tool. Don't rename or delete files behind its back. The IDE can get very confused. Most IDEs provide their own commands for renaming and removing files, that rename or remove the file and also update the IDE's internal records.

Start Code::Blocks by double-clicking your workspace file. If you keep one workspace file with all your class projects, you'll have easy access to old code, and Code::Blocks will remember many (though not all) of your settings when you create a new project.

Comments? Contact the Prof!