



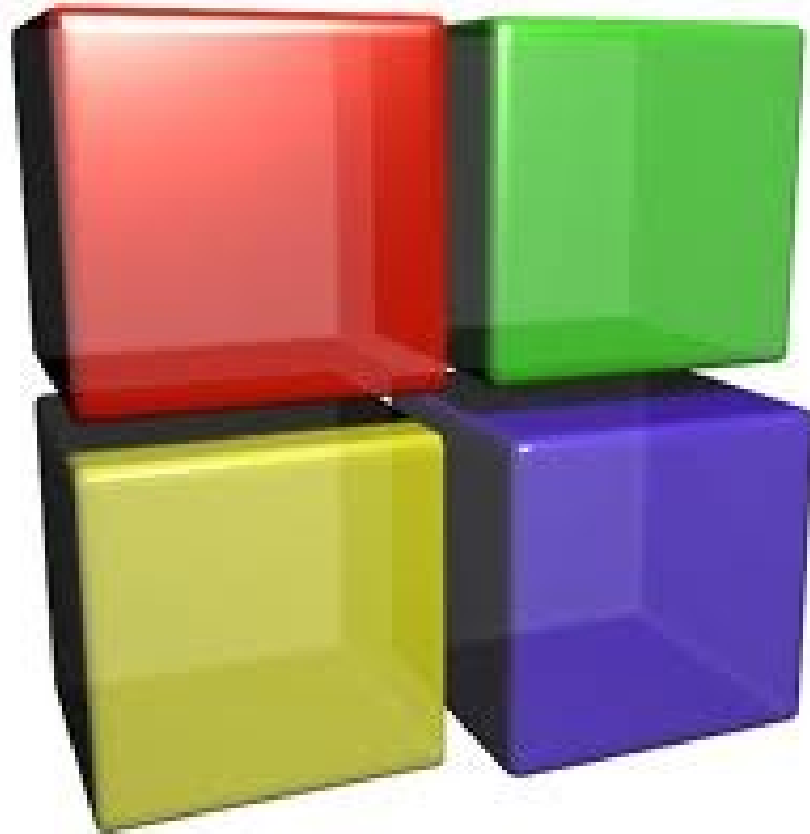
Welcome back!

144 days to competition!

A few changes from the fall...

- Code::Blocks!
- Unit testing
- Stricter architecture
- ACTUAL CODE!

Code::Blocks

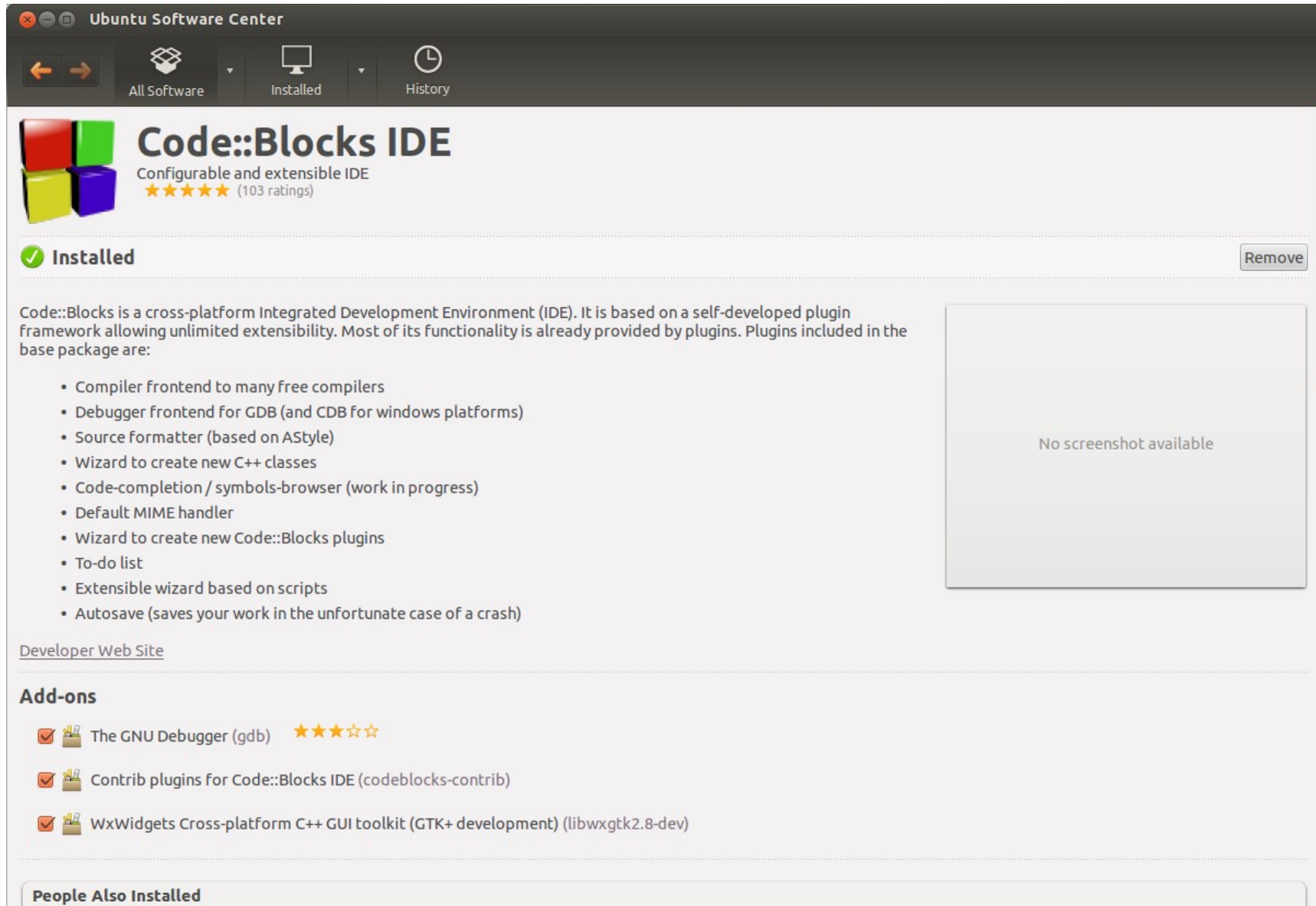


C::B v. Eclipse

- Eclipse
 - Powerful and popular IDE for many languages
 - Development experience can be buggy for C/C++
 - Multiple executables requires delving into makefiles
- C::B
 - Specifically designed for C and C++
 - Many templates for both console and GUI type applications
 - Built in support for multiple executables

“Do one thing well or many things poorly.”


Downloading Code::Blocks 10.05



The screenshot shows the Ubuntu Software Center window. The title bar reads "Ubuntu Software Center". The navigation bar includes "All Software", "Installed", and "History" tabs. The main content area displays the details for "Code::Blocks IDE", which is marked as "Installed". The software is described as a "Configurable and extensible IDE" with a 5-star rating from 103 users. A list of features is provided, including compiler frontends, a debugger, source formatting, and a wizard for creating C++ classes. A "Remove" button is visible in the top right corner of the software's detail view. Below the description, there is a section for "Add-ons" with three items: "The GNU Debugger (gdb)", "Contrib plugins for Code::Blocks IDE (codeblocks-contrib)", and "WxWidgets Cross-platform C++ GUI toolkit (GTK+ development) (libwxgtk2.8-dev)". A "Developer Web Site" link is also present. At the bottom, a section titled "People Also Installed" is partially visible.

Ubuntu Software Center

All Software Installed History

 **Code::Blocks IDE**
Configurable and extensible IDE
★★★★★ (103 ratings)




✓ **Installed** Remove

Code::Blocks is a cross-platform Integrated Development Environment (IDE). It is based on a self-developed plugin framework allowing unlimited extensibility. Most of its functionality is already provided by plugins. Plugins included in the base package are:

- Compiler frontend to many free compilers
- Debugger frontend for GDB (and CDB for windows platforms)
- Source formatter (based on AStyle)
- Wizard to create new C++ classes
- Code-completion / symbols-browser (work in progress)
- Default MIME handler
- Wizard to create new Code::Blocks plugins
- To-do list
- Extensible wizard based on scripts
- Autosave (saves your work in the unfortunate case of a crash)

[Developer Web Site](#)

Add-ons

-  The GNU Debugger (gdb) ★★★★★
-  Contrib plugins for Code::Blocks IDE (codeblocks-contrib)
-  WxWidgets Cross-platform C++ GUI toolkit (GTK+ development) (libwxgtk2.8-dev)

People Also Installed

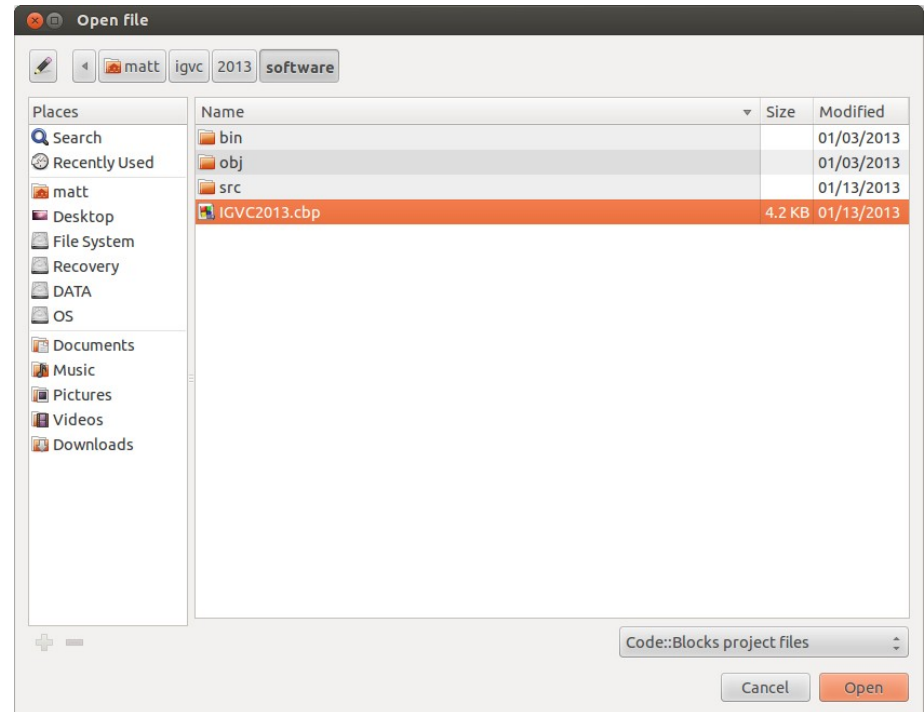
Setting up Code::Blocks

Project Files

- Code::Blocks projects are based on .cbp files (xml-style project description files)
- C::B uses “Global Variables” and relative paths for search directories
- We can take advantage of this to share a common project file in the git repo

Opening the IGVC Project File

- Open C::B
- Click File->Open
- Navigate to the IGVC git repo folder
- Select igvc[year].cbp

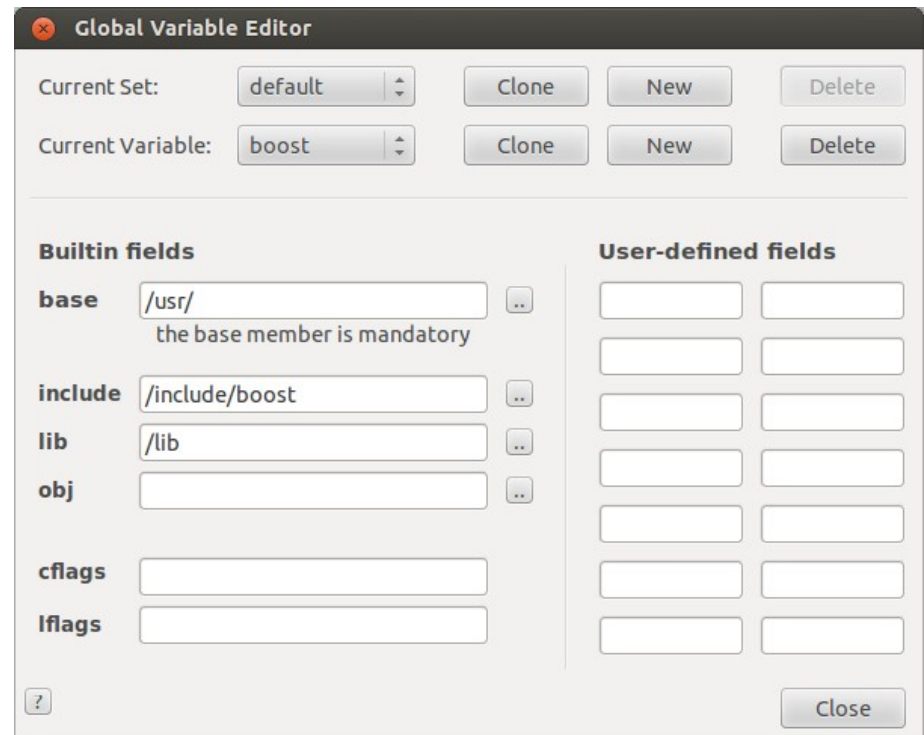


Global Variables

- When you open the IGVC project, you will probably see the Global Variables window pop up asking you to define global variables for libraries used in our project, like Boost!
- To make the project work, you must install the needed libraries and complete the fields...

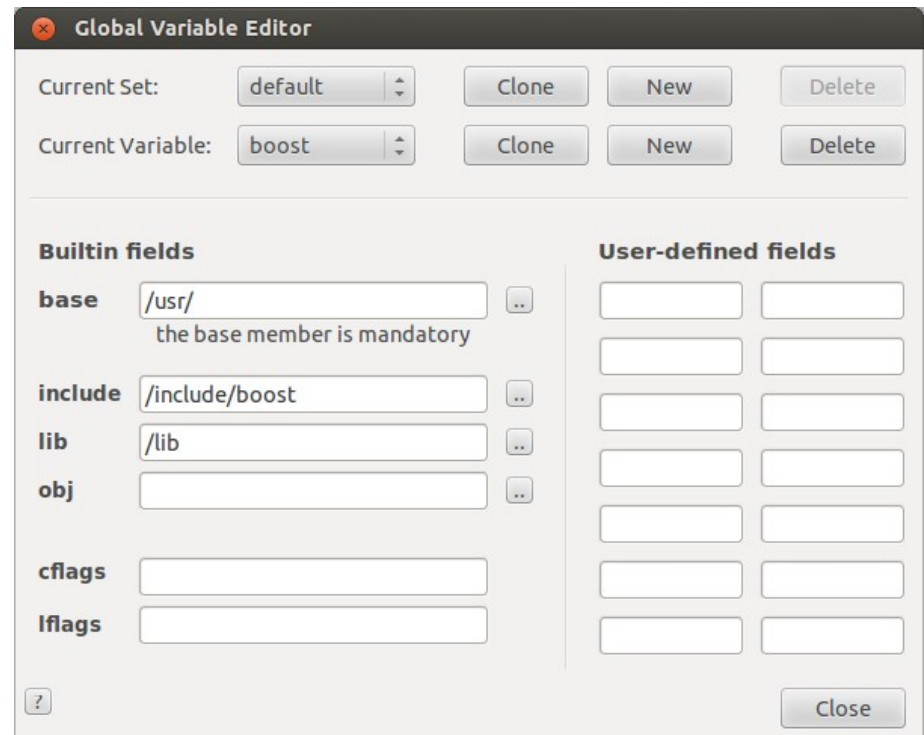
Global Variables

- base
 - This field must be filled
 - If your library installs to a single directory with `/lib/` and `/include/` subfolders, put that path here
 - Otherwise, as in this example, use the closest common parent folder



Global Variables

- include
 - This is the directory to find the library's header (.h/.hpp) files
- lib
 - This is the directory to find the library's lib (.o/.so/etc) files



Global Variables

- To reference Global Vars in C::B settings, use this format:

\$(#boost)

- Refers to the base folder

\$(#boost).include

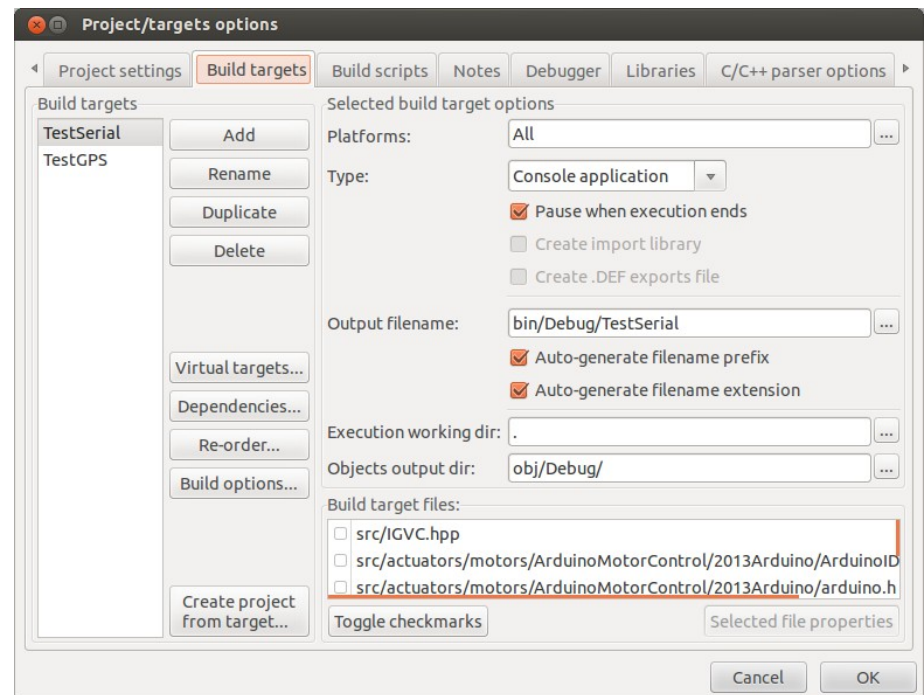
- Refers to the include subfolder
- Similar format can be used to access other subfolders

Targets

- Targets in C::B allow us to generate multiple executables from a single project file
- The following demo will demonstrate how to add a new target...

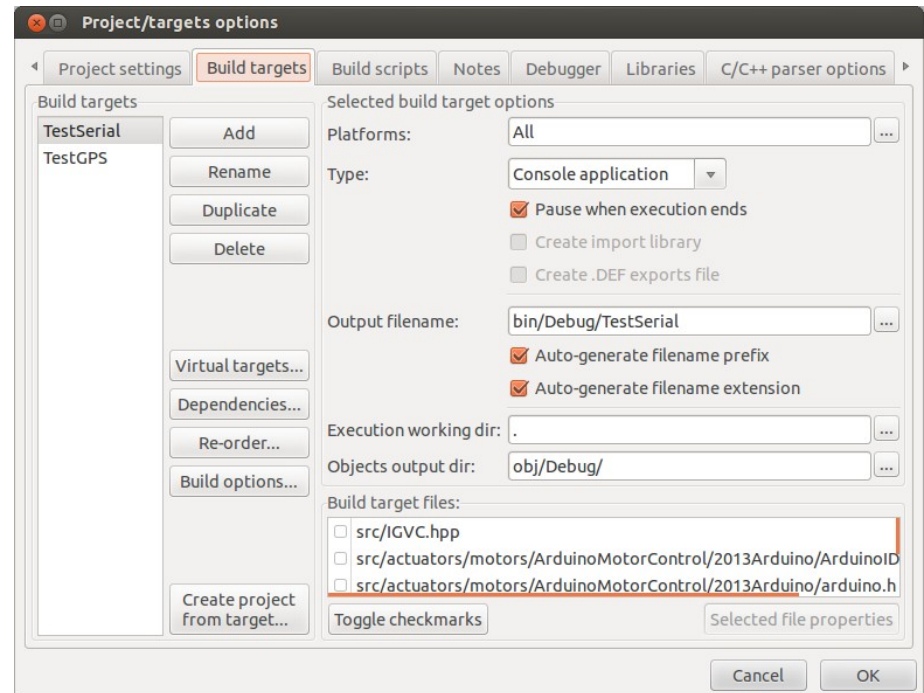
Targets

- Go to *Project->Properties->Build Targets*
- Select *Add* and give the target a descriptive name



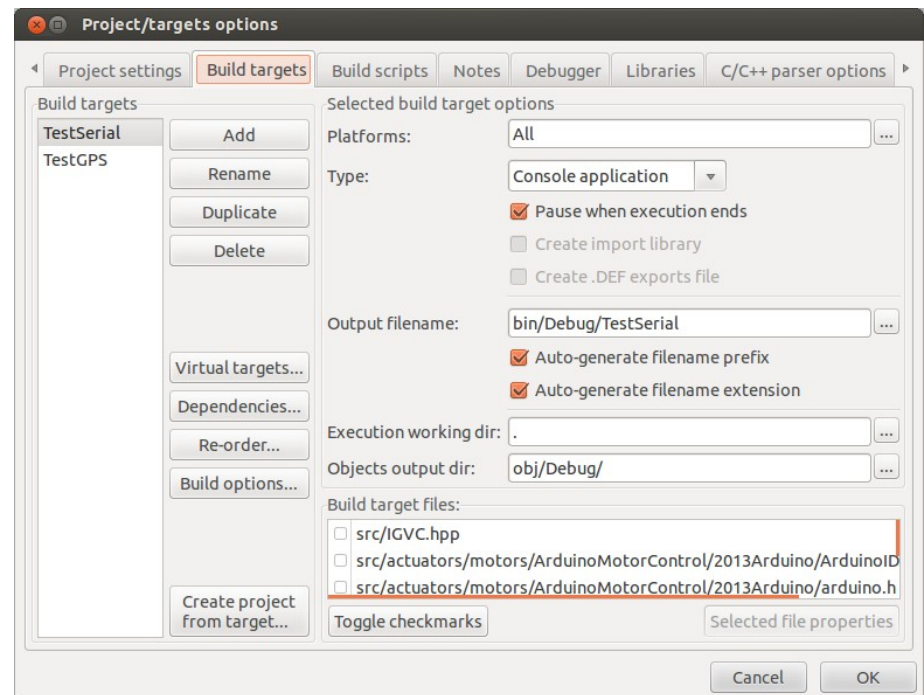
Targets

- Type: should be Console app
- Check “Pause when execution ends”
- Select all classes in the list that need to be compiled for your target



Targets

- *Output Filename*
 - This is the executable file that will be generated when the target is built
 - This file should be somewhere in the `/bin/` subdirectory and should probably match the name of your target so it's easy to know what's what when looking through `/bin/`

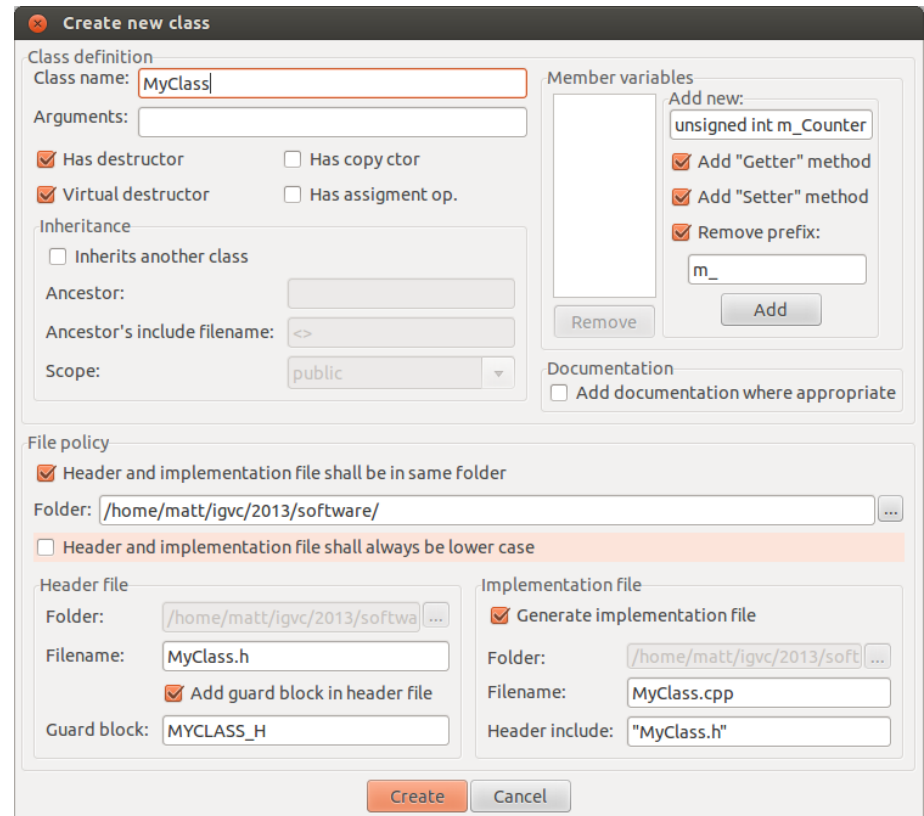


Class Wizard

- Lastly, let's look at one of the many wizards
C::B provides to make our C++ lives magically
delicious

Class Wizard

- Go to *File->New->Class*
- Fill out the *Class name*
- Most of the rest will be filled out for you at that point



Class Wizard

- If you already have the list of properties your class is going to have, you can use the *Member Variables* group to add properties with optional getters and setters auto-generated
- These can, of course, always be added/edited by hand later

Create new class

Class definition
Class name:
Arguments:
 Has destructor Has copy ctor
 Virtual destructor Has assignment op.
Inheritance
 Inherits another class
Ancestor:
Ancestor's include filename:
Scope:

Member variables
Remove
Add new:
 Add "Getter" method
 Add "Setter" method
 Remove prefix:

Add

Documentation
 Add documentation where appropriate

File policy
 Header and implementation file shall be in same folder
Folder:
 Header and implementation file shall always be lower case

Header file
Folder:
Filename:
 Add guard block in header file
Guard block:

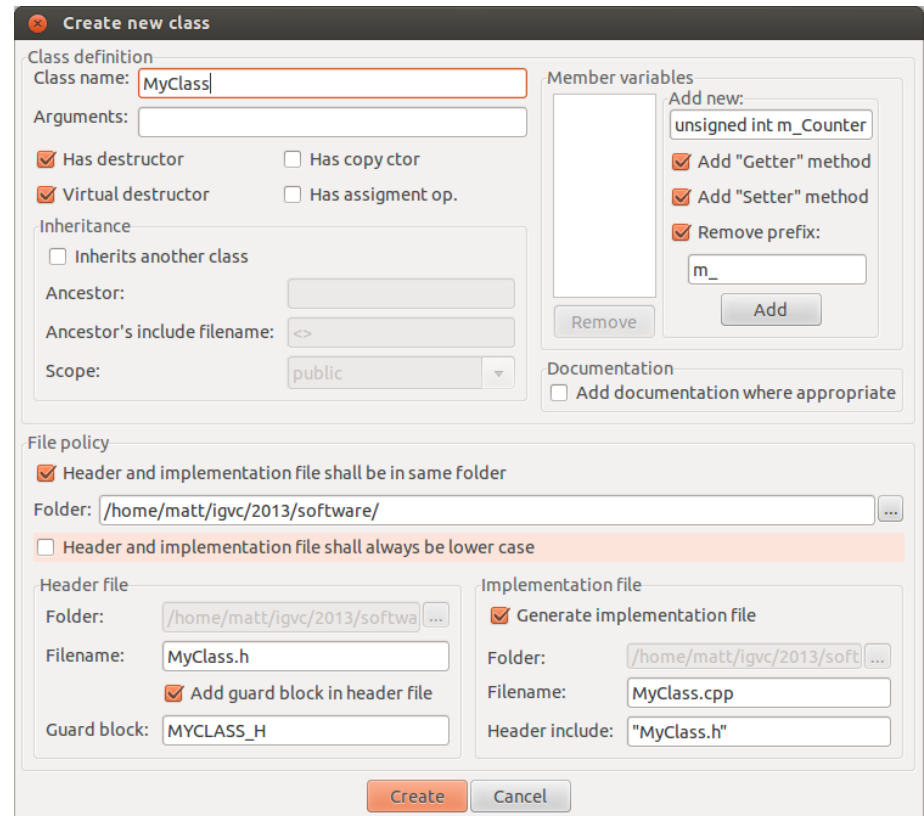
Implementation file
 Generate implementation file
Folder:
Filename:
Header include:

Create Cancel

Class Wizard

- **IMPORTANT**

- For the purpose of consistency in the organization of our code, **you MUST** check “Header and implementation.... same folder” !



Important things to remember

- Always use relative paths or environment variables to keep the project file sharable
- Use 'targets' to add new executables
- Don't forget the “Put headers and source files in same folder” check-box when creating new classes

The End

There are many more features and wizards that you will discover and take advantage of as you work in Code::Blocks. These are only the basics that you'll use early and often.