Nigh7Sh4de

Setting up SublimeGDB

March 28, 2014 Codings C++, debug, GDB, GNU, Sublime-Text, SublimeGDB Nigh7Sh4de

Sublime Text is by far one of the best text editors in the world. It can be gotten for free through illegitimate methods or by downloading it and simply never paying for it and just continuously clicking OK to the notification that says “Hey, maybe you should just give us your money now?”. Regardless of how moral and ethical you are, this is a great piece of software, and one of the things that truly set it apart from other editors is the community behind. Sublime Text currently has a metric duckload (not to be confused with the imperial duckload) of plugins and extensions, including one that is used to more easily install plugins and extensions from with sublime. One of these great plugins is called SublimeGDB (https://github.com/quarnster/SublimeGDB) which is a plugin that is used to debug C/C++ application. It is based off of GDB (http://www.sourceware.org/gdb/)which was created by the GNU Project (http://www.gnu.org/gnu/thegnuproject.html) (makers of g++).

It was kind of a pain to get started so to save you some pain, here are the steps I ended up taking:

1. Download and install SublimeGDB (preferably through the Package Manager (https://sublime.wbond.net/installation)).
2. Open (or create) a “sublime project” (by using the Ctrl+Shift+P palette or going to Project>Save Project As.
3. Open the generated .sublime-project file and add the the “settings” line and the 4 following it:

```
{
 "folders":
 [
 {
 "path": "NONE OF YOUR BUSINESS WORDPRESS"
 }
 ],
 "settings":
 {
 "sublimegdb_commandline": "gdb -nx -readnow -fullname --
interpreter=mi -args ./test1.exe",
 "sublimegdb_workingdir": "${folder:${project_path:bin/test1.exe}}"
 }
}
```

4. Replace all the "/test1.exe" with "/YOU_EXE_NAME_HERE.exe"
5. In your build statements make sure that you pass the compiler the '-g' flag (you basically need the executable to have explicit something or other for GDB to work…. Google it….)
6. Reference the Github repo for all the shortcuts, but basically its F9 to toggle a breakpoint and F5 to start debugging.

And that's that!

I faced two huge issues: one was figuring out exactly what the "sublimegdb_commandline" thing needed to be and then spent 2-3 hours searching for the '-g' solution.

Errors I noticed:

```
1) The given directory does not exist
```

This basically means you screwed up your "sublimegdb_workingdir" line

```
2) Source file named ******.cpp not found
```

'-g' into the compiler should fix this

```
3) You have not configured the plugin correctly
```

You need to modify you .sublime-project file as mentioned in steps 3 and 4

5 comments

1. **nope** says:
   March 6, 2015 at 5:48 am
   Thank you!!

2. **raashid** says:
   January 6, 2017 at 12:57 am
   Hi, I followed this blog of yours and instantly got SublimeGDB working. However, I wanted to have some kind of universal setting instead of project specific. A setting that allows me to debug any single C++ file without even thinking about making changes to SublimeGDB settings file.

   Thanks in advance.

   1. **() Nigh7Sh4de** says:
      January 12, 2017 at 10:28 pm
      This might be possible, honestly I haven't touched this in ages so I'm not sure. But I'm pretty sure that sublime expects that file to be present for any action that happen local to some folder as all the paths had to be absolute (or at least relative to `$project_path` which you wouldn't have if you didn't create a project). So if you want a global solution to debugging C++ in Sublime, I don't think this will help 😐

      1. **raashid** says:
         January 13, 2017 at 7:46 am
         Totally understandable. The problem was with my project structure. I had one parent folder and multiple sub-folders containing their own programs. I created the .sublime-project file at this parent folder, and was trying to find some setting that would allow me to run any of the executables in the subfolders without always changing the settings for SublimeGDB. It took me a while to understand the format SublimeGDB is looking for. Finally got it working using the following, inspired by your solution:

         "`json
         "settings":
         {
         "sublimegdb_executables":
         {
         "first_executable_name":
         {
         "workingdir": "${folder:${project_path:/path/to/first_executable_name}}",
         "commandline": "gdb –interpreter=mi ./first_executable_name"
         },
         "second_executable_name":
         {
         "workingdir": "${folder:${project_path:/path/to/second_executable_name}}",
         "commandline": "gdb –interpreter=mi ./second_executable"
         }
         }

}
“`

Regarding global settings, SublimeGDB’s documentation says the following:
> When you start debugging, you will be prompted to choose from one of your executables. Any settings not specified for that project will be searched in your project settings (with a sublimegdb_ prefix), then in your user settings, then in the default settings.

And THAT is what confused me as to why it was not pulling up the settings from user settings. Anyways, the above solution works and is simple enough so I am happy for now. Thanks!

3. **raashid** says:
January 13, 2017 at 7:48 am
aah damn! I thought markdown is enabled for comments. For anyone looking at the above comment, just ignore the “`json and “` right before “Regarding global settings…”.