USING cURL library with C++ in VS Code (Windows 10)

By: Noah Smith

This tutorial is written from the perspective of a frustrated beginner and aimed at other beginners. Hope its useful!

Curl is a command line tool and library used for transferring data with URL's. It has a lot of uses but can be a pain to get working so I'm writing this tutorial in the hope that it saves someone a few hours of troubleshooting.

The Problem:

As a bit of background, I was using cURL to pull data from a URL and use it elsewhere in some C++ code. This was being done in Visual Studio Code (VS Code) with the g++ compiler. Trying to use the: #include <curl/curl.h> header statement I kept getting the error when compiling:

```
c:\Users\iamno\SeinorSem\Calib.cpp:4:10: fatal error: curl\curl.h: No such file or directory
    4 | #include <curl\curl.h>
      |          ^~~~~~~~~~~~~
compilation terminated.
```

Installation

Using Scoop

I chose to use scoop package manager to install my cURL libraries. It seemed easy enough.

Start by installing scoop from the Scoop.sh homepage. The 2 step instructions are right on the homepage, nice and simple.

Once Scoop is installed, go back to the scoop website, search for CURL and copy the instillation command. It should be:

Scoop install main/curl

With this installed your curl library should be under the file path:

C:\Users\yourName\scoop\apps\curl\version

Git Clone

Another way to install curl is by cloning the git repository.

Navigate to the directory you want by using the “cd C:\file\path\you\want” command. I chose to place it in the same folder as my project to keep things all in the same place.

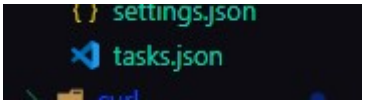
Clone the cURL Github repository using this command in the windows command prompt:

git clone https://github.com/curl/curl.git

VS Code Setup

Now that the cURL library is installed we need to set up VS code. This is assuming that you have C++ already setup in your VS Code. (if not, here’s a tutorial)

Press ctrl+shift+b to access the build task menu. Run the one that matches your compiler, mine’s g++. This should create a tasks.json file.



This file sets up the C++ build for your file. You can add different arguments and such, like you would when compiling from the command line. It makes our life easier.

Add the following pieces of code under the “args” section:

```
"args": [
    "-fdiagnostics-color=always",
    "-g",
    "${file}",
    "-o",
    "${fileDirname}\\${fileBasenameNoExtension}.exe",
    "-IC:\\Users\\iamno\\scoop\\apps\\curl\\8.7.1_9\\include",
    "-LC:\\Users\\iamno\\scoop\\apps\\curl\\8.7.1_9\\lib",
    "-lcurl"
],
```

I’ll explain to the best of my abilities what each line does.

"-IC:\\Users\\iamno\\scoop\\apps\\curl\\8.7.1_9\\include",

This line has the header “-I” which tells the build to include whatever follows it. In this case we are telling the build to include everything from the “include” folder of our cURL installation.

"-LC:\\Users\\iamno\\scoop\\apps\\curl\\8.7.1_9\\lib",

This line has the header “-L” which tells the build that the libraries are in the following path. In our case this is the aptly named “lib” portion of our cURL install.

The final line “-lcurl” tells the build to use the libcurl library. Your final tasks.json file should look something like this:

```
{
    "version": "2.0.0",
    "tasks": [
        {
            "type": "cppbuild",
            "label": "C/C++: g++.exe build active file",
            "command": "C:\\msys64\\ucrt64\\bin\\g++.exe",
            "args": [
                "-fdiagnostics-color=always",
                "-g",
                "${file}",
                "-o",
                "${fileDirname}\\${fileBasenameNoExtension}.exe",
                "-IC:\\Users\\iamno\\scoop\\apps\\curl\\8.7.1_9\\include",
                "-LC:\\Users\\iamno\\scoop\\apps\\curl\\8.7.1_9\\lib",
                "-lcurl"
            ],
            "options": {
                "cwd": "${fileDirname}"
            },
            "problemMatcher": [
                "$gcc"
            ],
            "group": "build",
            "detail": "compiler: C:\\msys64\\ucrt64\\bin\\g++.exe"
        }
    ]
}
```

With this tasks.json file complete, select your .cpp file you are trying to run, press ctrl+shift+b and run a build task using your compliler and the tasks.json file you just edited. This should complete successfully.

If you are getting an error message claiming that ("-lcurl" No such file or directory), double check the file paths you used to include the libraries and the include as this error indicates that it cannot find your install of cURL.

Work-Flow

Now that your build hopefully is successful, there is a slightly different work flow in VS Code to the usual "click run/compile". After running a build task, the executable file, .exe should appear in the output folder.

I found that the terminal in VS Code would not display error messages when running these .exe files. I needed to run them out of the windows command window using .\yourFile.cpp. This should display any runtime error messages in your code.

So the workflow now is:

-Run Build task in VS Code

-Run .exe file in the command window.

Conclusion

While this is far from a complete tutorial, it is how I managed to get the cURL library working for my application. I hope it is helpful to anyone else who is stuck on a similar issue and saves a few hours of googling.