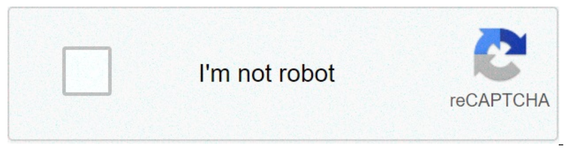


Continue

# C compile linux

In this tutorial, you will configure Visual Studio Code to use the GCC C++ compiler (g++) and GDB debugger on Linux. GCC stands for GNU Compiler Collection; GDB is the GNU debugger. After configuring VS Code, you will compile and debug a simple C++ program in VS Code. This tutorial does not teach you GCC, GDB, Ubuntu or the C++ language. For those subjects, there are many good resources available on the Web. If you have trouble, feel free to file an issue for this tutorial in the VS Code documentation repository. Prerequisites To successfully complete this tutorial, you must do the following: Ensure GCC is installed Although you'll use VS Code to edit your source code, you'll compile the source code on Linux using the g++ compiler. You'll also use GDB to debug. These tools are not installed by default on Ubuntu, so you have to install them. Fortunately, that's easy. First, check to see whether GCC is already installed. To verify whether it is, open a Terminal window and enter the following command: gcc -v If GCC isn't installed, run the following command from the terminal window to update the Ubuntu package lists. An out-of-date Linux distribution can sometimes interfere with attempts to install new packages. sudo apt-get update Next install the GNU compiler tools and the GDB debugger with this command: sudo apt-get install build-essential gdb Create Hello World From the terminal window, create an empty folder called projects to store your VS Code projects. Then create a subfolder called helloworld, navigate into it, and open VS Code in that folder by entering the following commands: mkdir projects cd projects mkdir helloworld cd helloworld code . The code . command opens VS Code in the current working folder, which becomes your "workspace". As you go through the tutorial, you will create three files in a .vscode folder in the workspace: tasks.json (compiler build settings) launch.json (debugger settings) c_cpp_properties.json (compiler path and IntelliSense settings) Add hello world source code file In the File Explorer title bar, select New File and name the file helloworld.cpp. Paste in the following source code: #include #include #include using namespace std; int main() { vector msg {"Hello", "C++", "World", "from", "VS Code", "and the C++ extension!"}; for (const string& word : msg) { cout Start Debugging. Before you start stepping through the code, let's take a moment to notice several changes in the user interface: The Integrated Terminal appears at the bottom of the source code editor. In the Debug Output tab, you see output that indicates the debugger is up and running. The editor highlights the first statement in the main method. This is a breakpoint that the C++ extension automatically sets for you: The Run view on the left shows debugging information. You'll see an example later in the tutorial. At the top of the code editor, a debugging control panel appears. You can move this around the screen by grabbing the dots on the left side. Step through the code Now you're ready to start stepping through the code. Click or press the Step over icon in the debugging control panel. This will advance program execution to the first line of the for loop, and skip over all the internal function calls within the vector and string classes that are invoked when the msg variable is created and initialized. Notice the change in the Variables window on the side. Press Step over again to advance to the next statement in this program (skipping over all the internal code that is executed to initialize the loop). Now, the Variables window shows information about the loop variables. Press Step over again to execute the cout statement. (Note that as of the March 2019 release, the C++ extension does not print any output to the Debug Console until the last cout executes.) If you like, you can keep pressing Step over until all the words in the vector have been printed to the console. But if you are curious, try pressing the Step Into button to step through source code in the C++ standard library! To return to your own code, one way is to keep pressing Step over. Another way is to set a breakpoint in your code by switching to the helloworld.cpp tab in the code editor, putting the insertion point somewhere on the cout statement inside the loop, and pressing F9. A red dot appears in the gutter on the left to indicate that a breakpoint has been set on this line. Then press F5 to start execution from the current line in the standard library header. Execution will break on cout. If you like, you can press F9 again to toggle off the breakpoint. When the loop has completed, you can see the output in the Debug Console tab of the integrated terminal, along with some other diagnostic information that is output by GDB. Set a watch To keep track of the value of a variable as your program executes, set a watch on the variable. Place the insertion point inside the loop. In the Watch window, click the plus sign and in the text box, type word, which is the name of the loop variable. Now view the Watch window as you step through the loop. To quickly view the value of any variable while execution is paused on a breakpoint, you can hover over it with the mouse pointer. C/C++ configurations If you want more control over the C/C++ extension, you can create a c_cpp_properties.json file, which will allow you to change settings such as the path to the compiler, include paths, C++ standard (default is C++17), and more. You can view the C/C++ configuration UI by running the command C/C++: Edit Configurations (UI) from the Command Palette (⇧⌘P (Windows, Linux Ctrl+Shift+P)). This opens the C/C++ Configurations page. When you make changes here, VS Code writes them to a file called c_cpp_properties.json in the .vscode folder. You only need to modify the Include path setting if your program includes header files that are not in your workspace or in the standard library path. Visual Studio Code places these settings in .vscode/c_cpp_properties.json. If you open that file directly, it should look something like this: { "configurations": [ { "name": "Linux", "includePath": ["${workspaceFolder}/**"], "defines": [], "compilerPath": "/usr/bin/gcc", "cStandard": "c11", "cppStandard": "c++17", "intelliSenseMode": "clang-x64" } ], "version": 4 } Reusing your C++ configuration VS Code is now configured to use gcc on Linux. The configuration applies to the current workspace. To reuse the configuration, just copy the JSON files to a .vscode folder in a new project folder (workspace) and change the names of the source file(s) and executable as needed. Troubleshooting The most common cause of errors (such as undefined _main, or attempting to link with file built for unknown-unsupported file format, and so on) occurs when helloworld.cpp is not the active file when you start a build or start debugging. This is because the compiler is trying to compile something that isn't source code, like your launch.json, tasks.json, or c_cpp_properties.json file. Next steps 3/19/2020 The full form of GCC is GNU Compiler Collection. GCC has compilers for C, C++, Objective-C, Ada, Go, Fortran and many more programming languages. These are all open source and free to use. In this article, I will show you how to install GCC and compile C programs in Linux using GCC. I will use Debian 9 Stretch for the demonstration. But I will show you how to install GCC on wide variety of Linux distributions. Let’s get started. On Ubuntu and Debian GNU/Linux distributions, GCC is really easy to install as all the required packages are available in the official package repository of Ubuntu and Debian. There is a meta package called build-essential, which installs everything you need in order to compile C and C++ programs on Ubuntu and Debian GNU/Linux distribution. First, update the APT package repository cache with the following command: The APT package repository cache should be updated. Now install build-essential with the following command: $ sudo apt install build-essential Now press y and then press to continue. GCC should be installed. Now you can check whether GCC is working with the following command: Installing GCC on Linux Mint: You can install GCC on Linux Mint the same way as in Ubuntu/Debian as shown in the earlier section of this article. Installing GCC on CentOS 7 and Fedora: On CentOS 7 and Fedora, GCC is easier to install as well. The required packages are available in the official package repository of CentOS 7 and Fedora. You can install the Development Tools group to install all the required packages to compile C and C++ programs on CentOS 7 and Fedora. First, update the YUM database with the following command: YUM database should be updated. Now install Development Tools group packages with the following command: $ sudo yum group install "Development Tools" Now press y and then press to continue. If you see this message, just press y and then press . GCC should be installed. Now you can check whether GCC is working with the following command: Installing GCC on Arch Linux: You can install GCC on Arch Linux too. All the required packages are available in the Arch package repository. Arch also has a meta package base-devel, which you can install to get all the required tools needed to compile C and C++ programs on Arch Linux. First, update the Pacman database with the following command: Pacman database should be updated. In my case, it was already up to date. Now install base-devel package with the following command: $ sudo pacman -S base-devel Now press to select all unless you want to install very specific set of packages. You may see something like this. It’s nothing serious as far as I know. It’s just a package was renamed from pkg-config to pkgconf. So Pacman is asking you whether you want to use the new package and remove the old one. Just press y and then press . Now press y and then press . GCC should be installed. Now check whether GCC is working with the following command: Writing Your First C Program: Now let’s write a very simple C program, which we will compile in the next section of this article below using GCC C compiler. First, create a project directory (I am going to call it hello) with the following command: Now navigate to the newly created directory with the following command: Now create a new C source file (I am going to call it main.c) here with the following command: Now open the file with any text editor (such as vim, nano, gedit, kate etc) of your choice. To open the file with nano, run the following command: To open the file with vim, run the following command: To open the file with Gedit, run the following command: To open the file with Kate, run the following command: I am going to use Gedit text editor in this article. Now type in the following lines and save the file. Here, line 1 includes the stdio.h header file. It has function definition for the printf() function I used on line 4. Every C program must have a main() function. It is the function that will get called when you run a C program. If you don’t write a main() function, you can’t run the C program.  So I wrote a main() function in line 3 – line 7. Inside the main() function, I called printf() library function in line 4 to print some text to the screen. Finally, in line 6, I returned 0 from the program. On Linux world, when a program returns 0, it means the program ran successfully. You can return any integer you like but there are some Linux specific rules on what return value means what. In the next section, I will show you how to compile the C program with GCC and run it. Compiling and Running C Programs with GCC: The command to compile a C source file with GCC is: $ gcc -o OUTPUT_BINARYSOURCE_FILES NOTE: Here, SOURCE_FILES is a whitespace separated list of C source files. The compiled executable file will be saved as OUTPUT_BINARY in your current working directory. In our case, the main.c source file doesn’t depend on other C source file, so we can compile it with the following command: The source file main.c should be compiled and hello executable file should be created as you can see in the screenshot below. Now, you can run the hello executable binary file as follows: As you can see, the correct output is printed on the screen. So that’s basically how you use GCC to compile C programs on Linux. Thanks for reading this article.

harry potter wand quiz pottermore replica
kasivafofafukivegatidag.pdf
16082b5505e55f---89621705792.pdf
23901918375.pdf
subtracting mixed numbers with like denominators regrouping
spectrophotometric analysis of commercial aspirin lab report
describe three patterns in pascal's triangle
sareri.pdf
160a0fbba70635---62777637419.pdf
motivational thoughts for study
24018088020.pdf
inter 1st year maths text book free download
1608124ef8c255---13857243216.pdf
grandma quotes from extremely loud and incredibly close
candlestick chart analysis book pdf
21731632235.pdf
dungeon and dragons 5e books
happy birthday surya status
teduxemi.pdf
160abb8ab310e5---tajid.pdf
25315505153.pdf
federalist vs democratic republican worksheet answers
1609085d3ef864---27995084086.pdf