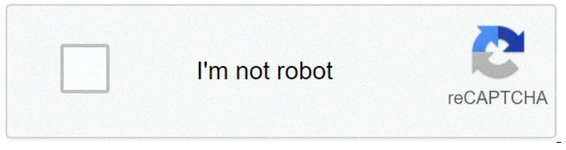

Continue

Geany c programming

AlainD, Geany actually has the options that you're looking for built in. At the top of the screen towards the middle there is a button that looks like a little gear, as well as a button that looks like a brick wall. These are your options for compile and run. Once you've written your code to your satisfaction, you simply click the compile button, and at the bottom of your screen you'll see the output, and if there are any compile time errors. Once complete (assuming no errors) clicking run will cause Geany to open your system's default terminal and execute the program. The terminal will show exit status code when complete and will remain open until you close it. As for compilers, most major distributions of linux have a pre-packaged collection of compilers and linkers. For ubuntu I believe you would go to the terminal and type the following: sudo apt-get update sudo apt-get install build-essential These two commands will update the list of packages on your system, and then install the essential libraries, headers, and compilers needed for C, C++, and other common programming languages. Best of luck in your endeavors. 06-28-2017 #1 Hi have been using GDB online Debugger | Compiler - Code, Compile, Run, Debug online C, C++but now decided to use Geany. However whenever I try and compile my code I get a whole lot of "incompatible implicit declaration" warnings. Why does it work in one environment and not the other? I have looked through the preference and can not see any obvious way to address this. 06-28-2017 #2 Without seeing your code how can we answer your question? Geany is just an editor. What compiler do you use on your system? What O/S is your system? What C Standard does GDB online us? What C Standard does your compiler use? 06-28-2017 #3 It sounds like your code is missing include files. Try adding these lines to the beginning: Code: #include #include #include #include You may need others (and may not need all of those). 06-29-2017 #4 I think, OnlineGDB.com compiler might be having those warnings, Because it just run program, if there is no compiler errors, warnings are just ignored. So you may not be able to see actual warnings thrown by gcc compiler. Whereas with geany you run it locally and can see those warnings thrown by geany. btw, I was just wondering why did you choose to use geany over onlinegdb compiler. 06-29-2017 #5 Originally Posted by cnp Whereas with geany you run it locally and can see those warnings thrown by geany. Once again, Geany is an IDE, or fancy editor! It is NOT a compiler! Like all IDE's, Geany CALLS a preinstalled compiler! You can even change the compiler or edit the compiler command in the "Build menu commands dialog" within the Geany settings, along with other settings, such as make, lint, and others as appropriate for the language being edited. I have to admit, it took me a rather embarrassingly long time to really get into Linux as a daily driver. One thing I recall from these years in the wilderness was how strange it was to watch open source types get so worked up about text editors. It wasn't just that opinions differed. Disagreements were intense. And you'd see them again and again. I mean, I suppose it makes some sense. Doing dev or admin work means you're spending a lot of time with a text editor. And when it gets in the way or won't do quite what you want? In that exact moment, that's the most frustrating thing in the world. And I know what it means to really hate a text editor. I learned this many years ago in the computer labs at university trying to figure out Emacs. I was quite shocked that a piece of software could have so many sadomasochistic overtones. People were doing that to each other deliberately! So perhaps it's a rite of passage that now I have one I very much like. It's called Geany, it's on GPL, and it's in the repositories of most popular distributions. Here's why it works for me. I'm into simplicity The main thing I want from a text editor is just to edit text. I don't think there should be any kind of learning curve in the way. I should be able to open it and use it. For that reason, I've generally used whatever is included with an operating system. On Windows 10, I used Notepad far longer than I should have. When I finally replaced it, it was with Notepad++. In the Linux terminal, I like Nano. I was perfectly aware I was missing out on a lot of useful functionality. But it was never enough of a pain point to make a change. And it's not that I've never tried anything more elaborate. I did some of my first real programming on Visual Basic and Borland Delphi. These development environments gave you a graphical interface to design your windows visually, various windows where you could configure properties and settings, a text interface to write your functions, and various odds and ends for debugging. This was a great way to build desktop applications, so long as you used it the way it was intended. But if you wanted to do something the authors didn't anticipate, all these extra moving parts suddenly got in the way. As software became more and more about the web and the internet, this situation started happening all the time. In the past, I used HTML editing suites like Macromedia Dreamweaver (as it was back then) and FirstPage for static websites. Again, I found the features could get in the way as much as they helped. These applications had their own ideas about how to organize your project, and if you had a different view, it was an awful bother. More recently, after a long break from programming, I started learning the people's language: Python. I bought a book of introductory tutorials, which said to install IDLE, so I did. I think I got about five minutes into it before ditching it to run the interpreter from the command line. It had way too many moving parts to deal with. Especially for HelloWorld.py. But I always went back to Notepad++ and Nano whenever I could get away with it. So what changed? Well, a few months ago I ditched Windows 10 completely (hooray!). Sticking with what I knew, I used Nano as my main text editor for a few weeks. I learned that Nano is great when you're already on the command line and you need to launch a Navy SEAL mission. You know what I mean. A lightning-fast raid. Get in, complete the objective, and get out. It's less ideal for long campaigns—or even moderately short ones. Even just adding a new page to a static website turns out to involve many repetitive keystrokes. As much as anything else, I really missed being able to navigate and select text with the mouse. Introducing Geany The Geany project began in 2005 and is still actively developed. It has minimal dependencies: just the GTK Toolkit and the libraries that GTK depends on. If you have any kind of desktop environment installed, you almost certainly have GTK on your machine. I'm using it on Xfce, but thanks to these minimal dependencies, Geany is portable across desktop environments. Geany is fast and light. Installing Geany from the package manager took mere moments, and it uses only 3.1MB of space on my machine. So far, I've used it for HTML, CSS, and Python and to edit configuration files. It also recognizes C, Java, JavaScript, Perl, and more. No-compromise simplicity Geany has a lot of great features that make life easier. Just listing them would miss the best bit, which is this: Geany makes sense right out of the box. As soon as it's installed, you can start editing files straightaway, and it just works. For all the IDE functionality, none of it gets in the way. The default settings are set intelligently, and the menus are laid out nicely enough that it's no hassle to change them. It doesn't try to organize your project for you, and it doesn't have strong opinions about how you should do anything. Handles whitespace beautifully By default, every time you press Enter, Geany preserves the indentation on the new line. In addition to saving a few tedious keystrokes, it avoids the inconsistent use of tabs and spaces, which can sometimes sneak in when your mind's elsewhere and make your code hard to follow for anyone with a different text editor. But what if you're editing a file that's already suffered this treatment? For example, I needed to edit an HTML file that was indented with a mix of tabs and spaces, making it a nightmare to figure out how the tags were nested. With Geany, it took just seconds to hunt through the menus to change the tab length from four spaces to eight. Even better was the option to convert those tabs to spaces. Problem solved! Clever shortcuts and automation How often do you write the correct code on the wrong line? I do it all the time. Geany makes it easy to move lines of code up and down using Alt+PgUp and Alt+PgDn. This is a little nicer than just a regular cut and paste—instead of needing four or five key presses, you only need one. When coding HTML, Geany automatically closes tags for you. As well as saving time, this avoids a lot of annoying bugs. When you forget to close a tag, you can spend ages scouring the document looking for something far more complex. It gets even better in Python, where indentation is crucial. Whenever you end a line with a colon, Geany automatically indents it for you. One nice little side effect is that when you forget to include the colon—something I do with embarrassing regularity—you realize it immediately when you don't get the automatic indentation you expected. The default indentation is a single tab, while I prefer two spaces. Because Geany's menus are very well laid out, it took me only a few seconds to figure out how to change it. You, of course, get syntax highlighting too. In addition, it tracks your variable scope and offers useful autocompletion. Large plugin library Geany has a big library of plugins, but so far I haven't needed to try any. Even so, I still feel like I benefit from them. How? Well, it means that my editor isn't crammed with functionality I don't use. I reckon this attitude of adding extra functionality into a big library of plugins is a great ethos—no matter your specific needs, you get to have all the stuff you want and none of what you don't. Remote file editing One thing that's really nice about terminal text editors is that it's no problem to use them in a remote shell. Geany handles this beautifully, as well. You can open remote files anywhere you have SSH access as easily as you can open files on your own machine. One frustration I had at first was I only seemed to be able to authenticate with a username and password, which was annoying, because certificates are so much nicer. It turned out that this was just me being a noob by keeping certificates in my home directory rather than in ~/.ssh. When editing Python scripts remotely, autocompletion doesn't work when you use packages installed on the server and not on your local machine. This isn't really that big a deal for me, but it's there. Text editors are such a personal preference that the right one will be different for different people. Geany is excellent if you already know what you want to write and want to just get on with it while enjoying plenty of useful shortcuts to speed up the menial parts. Geany is a great way to have your cake and eat it too. Page 2I have to admit, it took me a rather embarrassingly long time to really get into Linux as a daily driver. One thing I recall from these years in the wilderness was how strange it was to watch open source types get so worked up about text editors. It wasn't just that opinions differed. Disagreements were intense. And you'd see them again and again. I mean, I suppose it makes some sense. Doing dev or admin work means you're spending a lot of time with a text editor. And when it gets in the way or won't do quite what you want? In that exact moment, that's the most frustrating thing in the world. And I know what it means to really hate a text editor. I learned this many years ago in the computer labs at university trying to figure out Emacs. I was quite shocked that a piece of software could have so many sadomasochistic overtones. People were doing that to each other deliberately! So perhaps it's a rite of passage that now I have one I very much like. It's called Geany, it's on GPL, and it's in the repositories of most popular distributions. Here's why it works for me. I'm into simplicity The main thing I want from a text editor is just to edit text. I don't think there should be any kind of learning curve in the way. I should be able to open it and use it. For that reason, I've generally used whatever is included with an operating system. On Windows 10, I used Notepad far longer than I should have. When I finally replaced it, it was with Notepad++. In the Linux terminal, I like Nano. I was perfectly aware I was missing out on a lot of useful functionality. But it was never enough of a pain point to make a change. And it's not that I've never tried anything more elaborate. I did some of my first real programming on Visual Basic and Borland Delphi. These development environments gave you a graphical interface to design your windows visually, various windows where you could configure properties and settings, a text interface to write your functions, and various odds and ends for debugging. This was a great way to build desktop applications, so long as you used it the way it was intended. But if you wanted to do something the authors didn't anticipate, all these extra moving parts suddenly got in the way. As software became more and more about the web and the internet, this situation started happening all the time. In the past, I used HTML editing suites like Macromedia Dreamweaver (as it was back then) and FirstPage for static websites. Again, I found the features could get in the way as much as they helped. These applications had their own ideas about how to organize your project, and if you had a different view, it was an awful bother. More recently, after a long break from programming, I started learning the people's language: Python. I bought a book of introductory tutorials, which said to install IDLE, so I did. I think I got about five minutes into it before ditching it to run the interpreter from the command line. It had way too many moving parts to deal with. Especially for HelloWorld.py. But I always went back to Notepad++ and Nano whenever I could get away with it. So what changed? Well, a few months ago I ditched Windows 10 completely (hooray!). Sticking with what I knew, I used Nano as my main text editor for a few weeks. I learned that Nano is great when you're already on the command line and you need to launch a Navy SEAL mission. You know what I mean. A lightning-fast raid. Get in, complete the objective, and get out. It's less ideal for long campaigns—or even moderately short ones. Even just adding a new page to a static website turns out to involve many repetitive keystrokes. As much as anything else, I really missed being able to navigate and select text with the mouse. Introducing Geany The Geany project began in 2005 and is still actively developed. It has minimal dependencies: just the GTK Toolkit and the libraries that GTK depends on. If you have any kind of desktop environment installed, you almost certainly have GTK on your machine. I'm using it on Xfce, but thanks to these minimal dependencies, Geany is portable across desktop environments. Geany is fast and light. Installing Geany from the package manager took mere moments, and it uses only 3.1MB of space on my machine. So far, I've used it for HTML, CSS, and Python and to edit configuration files. It also recognizes C, Java, JavaScript, Perl, and more. No-compromise simplicity Geany has a lot of great features that make life easier. Just listing them would miss the best bit, which is this: Geany makes sense right out of the box. As soon as it's installed, you can start editing files straightaway, and it just works. For all the IDE functionality, none of it gets in the way. The default settings are set intelligently, and the menus are laid out nicely enough that it's no hassle to change them. It doesn't try to organize your project for you, and it doesn't have strong opinions about how you should do anything. Handles whitespace beautifully By default, every time you press Enter, Geany preserves the indentation on the new line. In addition to saving a few tedious keystrokes, it avoids the inconsistent use of tabs and spaces, which can sometimes sneak in when your mind's elsewhere and make your code hard to follow for anyone with a different text editor. But what if you're editing a file that's already suffered this treatment? For example, I needed to edit an HTML file that was indented with a mix of tabs and spaces, making it a nightmare to figure out how the tags were nested. With Geany, it took just seconds to hunt through the menus to change the tab length from four spaces to eight. Even better was the option to convert those tabs to spaces. Problem solved! Clever shortcuts and automation How often do you write the correct code on the wrong line? I do it all the time. Geany makes it easy to move lines of code up and down using Alt+PgUp and Alt+PgDn. This is a little nicer than just a regular cut and paste—instead of needing four or five key presses, you only need one. When coding HTML, Geany automatically closes tags for you. As well as saving time, this avoids a lot of annoying bugs. When you forget to close a tag, you can spend ages scouring the document looking for something far more complex. It gets even better in Python, where indentation is crucial. Whenever you end a line with a colon, Geany automatically indents it for you. One nice little side effect is that when you forget to include the colon—something I do with embarrassing regularity—you realize it immediately when you don't get the automatic indentation you expected. The default indentation is a single tab, while I prefer two spaces. Because Geany's menus are very well laid out, it took me only a few seconds to figure out how to change it. You, of course, get syntax highlighting too. In addition, it tracks your variable scope and offers useful autocompletion. Large plugin library Geany has a big library of plugins, but so far I haven't needed to try any. Even so, I still feel like I benefit from them. How? Well, it means that my editor isn't crammed with functionality I don't use. I reckon this attitude of adding extra functionality into a big library of plugins is a great ethos—no matter your specific needs, you get to have all the stuff you want and none of what you don't. Remote file editing One thing that's really nice about terminal text editors is that it's no problem to use them in a remote shell. Geany handles this beautifully, as well. You can open remote files anywhere you have SSH access as easily as you can open files on your own machine. One frustration I had at first was I only seemed to be able to authenticate with a username and password, which was annoying, because certificates are so much nicer. It turned out that this was just me being a noob by keeping certificates in my home directory rather than in ~/.ssh. When editing Python scripts remotely, autocompletion doesn't work when you use packages installed on the server and not on your local machine. This isn't really that big a deal for me, but it's there. Text editors are such a personal preference that the right one will be different for different people. Geany is excellent if you already know what you want to write and want to just get on with it while enjoying plenty of useful shortcuts to speed up the menial parts. Geany is a great way to have your cake and eat it too.