C++ IDE for Linux?

I want to expand my programming horizons to Linux. A good, dependable basic toolset is important, and what is more basic than an IDE?

I could find these SO topics:

- Lightweight IDE for linux and
- What tools do you use to develop C++ applications on Linux?

I'm not looking for a *lightweight* IDE. If an IDE is worth the money, then I will pay for it, so it need not be free.

My question, then:

> *What good, C++ programming IDE is available for Linux?*

The minimums are fairly standard: syntax highlighting, code completion (like intellisense or its Eclipse counterpart) and integrated debugging (e.g., basic breakpoints).

I have searched for it myself, but there are so many that it is almost impossible to separate the good from the bads by hand, especially for someone like me who has little C++ coding experience in Linux. I know that Eclipse supports C++, and I really like that IDE for Java, but is it any good for C++ and is there something better?

The second post actually has some good suggestions, but what I am missing is what exactly makes the sugested IDE so good for the user, what are its (dis)advantages?

Maybe my question should therefore be:

> *What IDE do you propose (given your experiences), and why?*

c++ linux ide

edited Jan 12 '10 at 5:13

community wiki
6 revs, 3 users 38%
Peter Mortensen

locked by Jeff Atwood ♦ Sep 29 '11 at 10:40

This question exists because it has historical significance, but **it is not considered a good, on-topic question for this site**, so please do not use it as evidence that you can ask similar questions here. This question and its answers are frozen and cannot be changed. More info: help center.

|

45 Answers

1 2 next

Actually, I recently made the switch from Visual Studio (*years* of experience there) to Linux and the first thing I did was try to find a reasonable IDE.

And then I noticed that this simply isn't how you work there*, and I threw everything out, spent a few days reading manuals, set up my shell (bash), set up a GVIM environment, learned the GCC/binutils toolchain, `make` and `gdb` and lived happily ever after.

There are certainly disadvantages but I am sure that I work much more productive since making the switch.

*) It *really* isn't! And that's not an anachronism either because the toolchain described above is incredibly mature and powerful. Any IDE has to replicate hundreds of unique features to get anywhere near the power of these tools that have been honed to the point of perfection for decades. The learning curve might be quite steep, but much less so than I expected.

answered Aug 23 '08 at 10:06

community wiki
Konrad Rudolph

128 i strongly disagree. a decent ide is important no matter what u'r working on. it drastically increases productivity. i use codeblocks and find that going back to VI is almost impossible. i have coded on Mac, Win and Linux, and although i find visual studio to be the best IDE, codeblocks comes close. – DavidG Oct 21 '08 at 17:40

27 David, perhaps you could detail (own answer …) what exactly you're missing from the toolchain I described that an IDE delivers. As I've said, I've come from a strong IDE background and my productivity increase was the exact inverse of yours. – Konrad Rudolph Oct 22 '08 at 9:31

161 UNIX *is* an IDE. All of it. – dsm May 8 '09 at 9:48

29 What refactoring support does Vim offer? In Eclipse (which runs under Linux), I can change the name of any Java method I wish, even if it is called in 300 places. Can you do it easily in Vim? – quant_dev Jun 10 '09 at 9:13

21 quant_dev: refactoring requires parsing the source code in some way. As far as I know, no VIM modules do that so the answer to your question is “none.” That's one of the reasons to prefer an IDE for IDE-centered languages such as Java. Since refactoring support (etc) for C++ is so minimal anyway (even in IDEs), this doesn't apply to C++. – Konrad Rudolph Jun 10 '09 at 13:56

1. Code::Blocks
2. Eclipse CDT

Soon you'll find that IDEs are not enough, and you'll have to learn the GCC toolchain anyway (which isn't hard, at least learning the basic functionality). But no harm in reducing the transitional pain with the IDEs, IMO.

answered Aug 23 '08 at 10:17

community wiki
Imran

8 +1 for eclipse cdt. Code::Blocks isn't good enough. – luiscubal Mar 29 '10 at 18:38

5 -1 for code blocks. -1 for eclipse. They both change their UI's during debugging. To the point where a novice user usually feels lost. They both can't debug fork. They both have nasty problems with SVN/CVS integration. (Latest Eclipse in tandem with Subclipse is broken in Gnome as of 4/28/11 and crashes every 10 minutes). Setting up source control is a nightmare and integration with these IDE's is just as difficult. – bleepzter Apr 28 '11 at 21:28

1 @bleepzter -1 for using SVN/CVS to start with ;) – OneOfOne Nov 2 '11 at 8:05

A quick answer, just to add a little more knowledge to this topic:
You must definitely check out NetBeans. Netbeans 6.7 has the following features:

- C/C++ Projects and Templates: Supports syntax highlighting, automatic code completion, automatic indentation.
- It has a C/C++ Debugger
- Supports Compiler Configurations, Configuration Manager and Makefile Support (with a Wizard).
- It has a Classes Window, a Usages Window and a File Navigation Window (or panel).
- A Macro expansion view, and also tooltips.
- Support for QT development.

I think it's a perfect (and far better) Visual Studio substitution, and a very good tool to learn C/C++.

Good Luck!

edited Oct 7 '09 at 12:12

community wiki
2 revs, 2 users 75%
ramayac

1 It's also really good for c++. It's got the best language parser ever. – Johan Boule Jul 20 '10 at 15:41

3 NetBeans is so much more enjoyable then Eclipse, I really wish more people start to realize that. – didibus Feb 24 '11 at 10:53

My personal favorite is the **CodeLite 2.x** IDE.

see: http://www.codelite.org

The decision to use CodeLite was based on a research regarding the following C++ IDE for Linux:

- Eclipse Galileo with CDT Plugin
- NetBeans 6.7 (which is also the base for the SunStudio IDE)
- KDevelop4
- CodeBlocks 8.02
- CodeLite 2.x

After all I have decided to use *CodeLite 2.x*.

Below I have listed some Pros and Cons regarding the mentioned C++ IDEs. Please note, that this reflects my personal opinion only!

EDIT: what a pity that SOF doesn't support tables, so I have to write in paragraphs ...

Eclipse Galileo with CDT Plugin

Pros:

- reasonable fast
- also supports Java, Perl(with E.P.I.C plugin)
- commonly used and well maintained
- also available for other OS flavours (Windows, MacOS, Solaris, AIX(?))

Cons:

- GUI is very confusing and somewhat inconsistent - not very intuitive at all
- heavy weight
- Only supports CVS (AFAIK)

NetBeans 6.7 (note this is also the base for the SunStudio IDE)

Pros:

- one of the most intuitive GUI I have ever seen
- also supports Java, Python, Ruby
- integrates CVS, SVN, Mercurial
- commonly used and well maintained
- also available for other OS flavours (Windows, MacOS, Solaris)

Cons:

- extremly slow
- heavy weight
- uses Spaces for indentation, which is not the policy at my work. I'm sure this is configurable, but I couldn't find out how to to that

KDevelop4 (note: I did not much testing on it)

Pros:

- commonly used on Linux
- integrates CVS, SVN, Mercurial

Cons:

- the GUI looks somewhat old fashioned
- heavy weight
- very specific to the KDE environment

CodeBlocks 8.02 (note: I did not much testing on it)

Pros:

- reasonable fast

Cons:

- the GUI looks somewhat old fashioned (although it has a nice startup screen)
- the fonts in the editor are very small
- some icons (e.g. the debugger related icons starting/stepping) are very small
- no source control integration

CodeLite 2.x (note: this is my personal favorite)

Pros:

- the best, modern looking and intuitive GUI I have seen on Linux
- lightweight
- reasonable fast
- integrates SVN
- also available on other OS flavours(Windows, MacOS, Solaris(?))

Cons:

- no CVS integration (that's important for me because I have to use it at work)
- no support for Java, Perl, Python (would be nice to have)

edited Dec 24 '09 at 10:26 — community wiki, 6 revs, anon

11 Eclispe has support for Hg, Git, SVN and others via plugins. And startup/splash screens suck huge balls. They suck down resources and offer very little in benefit. And they usually pop up in front of whatever I'm working on while waiting for the app to load. PortableApps and Eclipse need to get rid of them. – Chris Kaminski Mar 29 '10 at 19:28

4 Codelite got the same keyboard shortcut as Visual Studio for debugging, making it very user-friendly for Visual addicts. – Raoul Supercopter Apr 2 '10 at 15:19

KDevelop is pretty nice.

answered Aug 23 '08 at 10:53 — community wiki, fulmicoton

2 KDevelop4 is a really, really great C++ IDE. Unfortunately it hasn't yet been released, so I'm hesitant to recommend it, especially to someone new to Linux. I build it from SVN weekly and am nothing but impressed with the direction it's taking. – Parker Coates May 8 '09 at 18:56

At least for Qt specific projects, the Qt Creator (from Nokia/Trolltech) shows great promise.

edited Nov 22 '09 at 23:58 — community wiki, 2 revs, 2 users 67%, Henrik Hartz

5 I use it for non-Qt projects as well. – Chance Jul 21 '11 at 14:43

> could you clarify a little bit more how it was for you, what you had to change. Maybe you could point me in the right direction by providing some links to the information you used.

My first source were actually the tools' `man` pages. Just type

```
$ man toolname
```

on the command line (`$` here is part of the prompt, not the input).

Depending on the platform, they're quite well-written and can also be found on the internet. In the case of `make`, I actually read the complete documentation which took a few hours. Actually, I don't think this is necessary or helpful in most cases but I had a few special requirements in my first assignments under Linux that required a sophisticated makefile. After writing the makefile I gave it to an experienced colleague who did some minor tweaks and corrections. After that, I pretty much knew `make`.

I used GVIM because I had some (but not much) prior experience there, I can't say anything at all about Emacs or alternatives. I find it really helps to read other peoples' `.gvimrc` config file. Many people put it on the web. Here's mine.

Don't try to master all binutils at once, there are too many functions. But get a general overview so you'll know where to search when needing something in the future. You *should*, however, know all the important parameters for `g++` and `ld` (the GCC linker tool that's invoked automatically except when explicitly prevented).

> Also I'm curious, do you have code completion and syntax highlighting when you code?

Syntax highlighting: yes, and a much better one than Visual Studio. Code completion: yes-*ish*. First, I have to admit that I didn't use C++ code completion even in Visual Studio because (compared to VB and C#) it wasn't good enough. I don't use it often now but nevertheless, GVIM *has* native code completion support for C++. Combined with the ctags library and a plug-in like taglist this is almost an IDE.

Actually, what got me started was an article by Armin Ronacher. Before reading the text, look at the screenshots at the end of it!

> do you have to compile first before getting (syntax) errors?

Yes. But this is the same for Visual Studio, isn't it (I've never used Whole Tomato)? Of course, the syntax highlighting will show you non-matching brackets but that's about all.

> and how do you debug (again think breakpoints etc)?

I use `gdb` which is a command-line tool. There's also a graphical frontend called `DDD`. `gdb` is a modern debugging tool and can do everything you can do in an IDE. The only thing that really annoys me is reading a stack trace because lines aren't indented or formatted so it's really hard to scan the information when you're using a lot of templates (which I do). But those also clutter the stack trace in IDEs.

> Like I said, I had the 'pleasure' to set my first steps in the Java programming language using windows notepad and the command line java compiler in high school, and it was, .. wel a nightmare! certainly when I could compare it with other programming courses I had back then where we had decent IDE's

You shouldn't even try to compare a modern, full-feature editor like Emacs or GVIM to Notepad. Notepad is an embellished `TextBox` control, and this really makes all the difference. Additionally, working on the command line is a very different experience in Linux and Windows. The Windows `cmd.exe` is severely crippled. PowerShell is much better.

/EDIT: I should mention explicitly that **GVIM has tabbed editing** (as in tabbed browsing, not tabs-vs-spaces)! It took me ages to find them although they're not hidden at all. Just type `:tabe` instead of plain `:e` when opening a file or creating a new one, and GVIM will create a new tab. Switching between tabs can be done using the cursor or several different shortcuts (depending on the platform). The key `gt` (type g, then t in command mode) should work everywhere, and jumps to the next tab, or tab no. *n* if a number was given. Type `:help gt` to get more help.

edited Jan 18 '12 at 10:51

community wiki
6 revs, 3 users 86%
Konrad Rudolph

As an old-time UNIX guy, I always use Emacs. But that has a pretty steep and long learning curve, so I'm not sure I can recommend it to newcomers.

There really isn't a "good" IDE for Linux. Eclipse is not very good for C/C++ (CDT is improving, but is not very useful yet). The others are missing all the features you are going to be looking for.

It really is important to learn how all the individual tools (gcc, make, gdb, etc.) work. After you do so, you may find the Visual Studio way of doing things to be very limiting.

edited May 8 '09 at 9:39

community wiki
2 revs
Kristopher Johnson

Not to repeat an answer, but I think I can add a bit more.

Slickedit is an excellent IDE.

It supports large code-bases well without slowing down or spending all its time indexing. (This is a problem I had with eclipse's cdt). Slickedit's speed is probably the nicest thing about it, actually.
The code completion works well and there are a large amount of options for things like automatic formatting, beautification and refactoring.
It does have integrated debugging.
It has plug-in support and fairly active community creating them.
In theory, you should be able to integrate well with people doing the traditional makefile stuff, as it allows you to create a project directly from one, but that didn't work as smoothly as I would have liked when I tried it.
In addition to Linux, there are Mac and Windows versions of it, should you need them.

answered Sep 8 '08 at 17:31

community wiki
rck

Just a quick follow up for this question...

It's been a month since I started using Vim as my main 'GUI' tool for programming C++ in Linux. At first the learning curve was indeed a bit steep but after a while and with the right options turned on and scripts running I really got the hang of it!

I love the way how you can shape Vim to suite your needs; just add/change key mappings and Vim is turned into a highly productive 'IDE'.

The toolchain to build and compile a C++ program on Linux is also really intuitive. make and g++ are *the* tools you'll use.

The debugger ddd is however not really that good, but maybe that's because I haven't had the time to master it properly.

So to anyone who is, or was looking for a good C++ IDE in Linux, just like I was, your best bet lays with the standard available tools in Linux itself (Vim, g++, ddd) and you should really at least try to use them, before looking for sonething else...

Last but not least, I really want to thank konrad for his answer here, It really helped me find my way in the Linux development environment, thank you!

I'm also *not* closing this question, so people can still react or maybe even add new suggestions or additions to the already really nice answers...

edited Oct 7 '09 at 13:23

community wiki
2 revs, 2 users 64%
Sven

Checkout Netbeans, it's written in Java so you'll have the same environment regardless of your OS, and it supports a lot more than just C++.

I'm not going to try to convince you, because I think IDEs can be a very personal choice. For me it improves my productivity by being fast, supporting the languages I code in and has the standard features you'd expect from an IDE.

answered Aug 23 '08 at 9:55

community wiki
Steve M

4 I just can add my +1 to this. netbeans has the best c++ language parser i've seen in an ide ; beats eclipse's cdt. – Johan Boule Jul 20 '10 at 15:36

I recommend you read The Art Of UNIX Progranmming. It will frame your mind into using the environment as your IDE.

answered May 8 '09 at 9:59 community wiki dsm

Netbeans is in my experience the most heavyweight IDE there is. I'm using Eclipse with the CDT plugin, its fast and it works pretty well.

http://www.eclipse.org/cdt/

answered Aug 23 '08 at 9:57 community wiki exander Stolz

2 I just can add my +1 to this. netbeans is really a remarkable piece of software. Java, Ruby, C/C++ all is supported and suprise it works ;-) – Friedrich Mar 15 '09 at 6:43

2 You meant +1 for netbeans, not for eclipse's cdt, right? – Johan Boule Jul 20 '10 at 15:37

If you like Eclipse for Java, I suggest Eclipse CDT. Despite C/C++ support isn't so powerful as is for Java, it still offers most of the features. It has a nice feature named Managed Project that makes working with C/C++ projects easier if you don't have experience with Makefiles. But you can still use Makefiles. I do C and Java coding and I'm really happy with CDT. I'm developing the firmware for a embedded device in C and a application in Java that talks to this device, and is really nice to use the same environment for both. I guess it probably makes me more productive.

answered Aug 23 '08 at 14:32 community wiki jassuncao

Shorter answer is: choosing whatever "editor" you like, then use GDB console or a simple GDB front end to debug your application. The debuggers come with fancy IDEs such as Netbeans sucks for C/C++. I use Netbeans as my editor, and Insight and GDB console as my debugger.

With insight, you have a nice GUI and the raw power of GDB.

As soon as you get used to GDB commands, you will start to love it since you can do things you will never be able to do using an GUI. You can use even use **Python** as your script language if you are using GDB 7 or newer version.

Most people here paid more attentions to the "Editors" of the IDEs. However, if you are developing a large project in C/C++, you could easily spend more than 70% of your time on the "debuggers". The debuggers of the fancy IDEs are at least 10 years behind Visual Studio. For instance, Netbenas has very similar interfaces with Visual Studio. But its debugger has a number of disadvantages compared to Visual Studio.

1. Very slow to display even a array with only a few hundreds of elements
2. No highlighting for changed value (By default, visual studio shows changed values in the watch windows in red)
3. Very limited ability to show memory.
4. You cannot modify the source code then continue to run. If a bug takes a long time to hit, you would like to change the source and apply the changes live and continue to run your application.
5. You cannot change the "next statement" to run. In Visual Studio, you can use "Set Next Statement" to change how your application runs. Although this feature could crash your application if not used properly, but it will save you a lot of time. For instance, if you found the state of your application is not correct, but you do not know what caused the problems, you might want to rerun a certain region of the your source codes without restarting your application.
6. No built-in support for STL such as vector, list, deque and map etc.
7. No watch points. You need to have this feature, when you need to stop your application right at the point a variable is changed. Intel based computers have hardware watch points so that the watch points will not slow down your system. It might takes many hours to find some

hard-to-find bugs without using watch points. "Visual Studio" calls "watch pointer" as "Data BreakPoint".

The list can be a lot longer.

I was so frustrated by the disadvantages of the Netbeans or other similar IDEs, so that I started to learn GDB itself. I found GDB itself are very powerful. GDB does not have all the "disadvantages" mentioned above. Actually, GDB is very powerful, it is even better than Visual Studio in many ways. Here I just show you a very simple example.

For instance, you have a array like:

```
struct IdAndValue
{
  int ID;
  int value;
};


IdAndValue IdAndValues[1000];
```

When your application stops, and you want to examine the data in IdAndValues. For instance, if you want to find the ordinals and values in the array for a particular "ID", you can create a script like the following:

```
define PrintVal
set $i=0
printf "ID = %d\n", $arg0
while $i<1000
  if IdAndValues[$i].ID == $arg0
    printf "ordinal = %d, value = %d\n", $i, IdAndValues[$i].vaue
    set $i++
  end
end
end
```

You can use all variables in your application in the current context, your own variables (in our example, it is $i), arguments passed (in our example, it is $arg0) and all GDB commands (built-in or user defined).

Use **PrintVal 1** from GDB prompt to print out values for ID "1"

By the way, NetBeans does come with a GDB console, but by using the console, you could crash Netbeans. And I believe that is why the console is hidden by default in NetBeans

edited Apr 4 '10 at 23:29 — community wiki, 9 revs, Charles Zhang

---

make + gedit + gdb = one great IDE

answered Dec 11 '09 at 19:34 — community wiki, Matt Fichman

I am using "Geany" found good so far, its fast and light weight IDE.

Among Geany's features are:

- Code folding
- Session saving
- Basic IDE features such as syntax highlighting, tabs, automatic indentation and code completion
- Simple project management
- Build system
- Color picker (surprisingly handy during web development)
- Embedded terminal emulation
- Call tips
- Symbol lists
- Auto-completion of common constructs (such as if, else, while, etc.)

edited Jan 6 '11 at 6:28

community wiki
2 revs
TheCottonSilk

I quite like Ultimate++'s IDE. It has some features that were designed to use with their own library (which, BTW, is quite a nice toolkit if you don't want to buy on either GTK+ or QT) but it works perfectly well with general C++ projects. It provides decent code completion, good syntax colouring, integrated debugging, and all other features most modern IDEs support.

edited Oct 11 '08 at 17:50

community wiki
2 revs, 2 users 86%
dguaraglia

I really suggest codeblocks. It's not as heavy as Eclipse and it's got Visual Studio project support.

edited Oct 7 '09 at 12:03

community wiki
2 revs, 2 users 75%
DavidG

> And then I noticed that this simply isn't how you work there*, and I threw everything out, spent a few days reading manuals, set up my shell (bash), set up a GVIM environment, learned the GCC/binutils toolchain, make and gdb and lived happily ever after.

I'd mostly agree, but the problem is also one of perception: we forget how difficult it was to become productive in any chose IDE (or other environment). I find IDE's (Visual Studio, NetBeans, Eclipse) amazingly cumbersome in so many ways.

> As an old-time UNIX guy, I always use Emacs. But that has a pretty steep and long learning curve, so I'm not sure I can recommend it to newcomers.

I'd second that; use Emacs as my primary editor on both Linux and on MSW (XP2,W2K). I would disagree that it has a *steep* learning curve, but would say that because of the huge number of features it has a *long* learning curve. You can be productive within a short time, but if you want you can learn new features of it for years to come.

However -- don't expect all the features of Emacs to be available on drop-down menus, there is just too much functionality to find it there.

As I metioned, I've used GNU Emacs on MSW for years. And it's always worked well with Visual Studio until I "upgraded" to 2008; now it *sometimes* delays many seconds before refreshing files from disk. The main reason for editing in the VS window is the "Intellisense" code completion feature.

answered Feb 5 '09 at 20:21

community wiki
NVRAM

Although I use Vim, some of my co-workers use SlickEdit which looks pretty good. I'm not certain about integrated debugging because we wouldn't be able to do that on our particular project anyway.

SlickEdit does have good support for navigating large code bases, with cross referencing and tag jumping. Of course it has the basic stuff like syntax highlighting and code completion too.

edited Oct 7 '09 at 13:59

community wiki
2 revs, 2 users 57%
Greg Hewgill

Perhaps the Linux Tools Project for Eclipse could fill your needs?

> The Linux Tools project aims to bring a full-featured C and C++ IDE to Linux developers. We

> build on the source editing and debugging features of the CDT and integrate popular native development tools such as the GNU Autotools, Valgrind, OProfile, RPM, SystemTap, GCov, GProf, LTTng, etc. Current projects include LTTng trace viewers and analyzers, an RPM .spec editor, Autotools build integration, a Valgrind heap usage analysis tool, and OProfile call profiling tools.

edited Sep 14 '10 at 14:45 — community wiki
2 revs, 2 users 56%
Jean Hominal

I hear Anjuta is pretty slick for Gnome users. I played a bit with KDevelop and it's nice, but sort of lacking featurewise. Code::Blocks is also very promising, and I like that one best.

answered Aug 23 '08 at 12:14 — community wiki
wvdschel

Sun Studio version 12 is a free download(FREE and paid support available) -- http://developers.sun.com/sunstudio/downloads/thankyou.jsp?submit=%A0FREE+Download%A0%BB%A0.

I'm sure you have code completion and debugging support including plugin support in this IDE.

Sun Studio is available for Linux as well as Solaris. forums : http://developers.sun.com/sunstudio/community/forums/index.jsp. Sun Studio Linux forums : http://forum.sun.com/forum.jspa?forumID=855

I'll be eager to hear your feedback on this tool.

BR,
~A

answered Sep 17 '08 at 19:55 — community wiki
anjanb

I've previously used Ultimate++ IDE and it's rather good.

answered Dec 21 '08 at 19:48 — community wiki
Hernán

---

geany I recommend

edited Mar 24 '09 at 12:01 — community wiki
2 revs, 2 users 67%
Sven

I use Eclipse CDT and Qt Creator (for Qt applications).

That's my preferences. It's a very suggestive question and there is as many answers as there is developers. :)

edited Jan 30 '10 at 15:16 — community wiki
2 revs
Etienne Savard

SlickEdit. I have used and loved SlickEdit since 2005, both on Windows and on Linux. I also have experience working in Visual Studio (5, 6, 2003, 2005) and just with Emacs and command line. I use SlickEdit with external makefiles, some of my teammates use SlickEdit, others use Emacs/vi. I do not use the integrated debugger, integrated version control, integrated build

system: I generally find too much integration to be real pain. SlickEdit is robust (very few bugs), fast and intuitive. It is like a German car, a driver's car.

The newest versions of SlickEdit seem to offer many features that do not interest me, I am a little worried that the product will become bloated and diluted in the future. For now (I use V13.0) it is great.

answered Nov 9 '09 at 2:35 community wiki Radim Cernej

For me Ultimate++ seems to be the best solution to write cross-os program

answered Feb 17 '10 at 17:13 community wiki IonLupascu

If you were using vim for a long time, then you should actually make that as your IDE. There are a lot of addons available. I found several of those as pretty useful, and compiled it here, have a look at it.

- C/C++ IDE
- Source code browser

And a lot more in the vi / vim tips & tricks series over there.

answered Mar 30 '10 at 11:04 community wiki thegeek

---

1 2 next