UNIX tips and tricks for a new user, Part 2: The vi text editor

Skill Level: Intermediate

Tim McIntire (tm@timmcintire.net)
Consultant
Freelance Writer

07 Nov 2006

The vi text editor might seem counterintuitive to new users but, make no mistake, there is a good reason this 30-year old tool is still widely used by many of the best developers in the world. The vi text editor separates operations into insert mode and command mode, which gives you ultrafast access to key commands that can edit, insert, and move text in on-the-fly, user-defined segments.

Section 1. Before you start

Learn what to expect from this tutorial, and how to get the most out of it.

About this series

This four-part tutorial series covers UNIX® basics from a user perspective. This initial tutorial is a good brush-up for users who have been away from UNIX-like operating systems for some time. It's also useful for brand-new UNIX users coming from a Windows® background, because it uses references and comparisons to Windows. Later tutorials in the series will cover specific applications (vi, for instance) in detail and discuss shell tricks and tips.

About this tutorial

The vi editor has been around for 30 years, with only minor changes. It maintains a mouse-free and keyboard-driven interface that lets users keep their fingers in the home positions at all times. Users can switch between two modes, insert and command mode, to either insert text or manipulate and navigate the document, respectively. The command mode provides users with all the actions that are

normally achieved through a point-and-click mouse-driven interface.

Objectives

The objective of this tutorial is to make new vi users comfortable with creating, editing, and navigating text documents. It focuses on common vi commands, and it goes into detail on some of vi's more esoteric features. The most important thing to remember when learning to use vi is that you should expect text editing to feel slow and cumbersome at first. Try to think back to the first time you used a mouse or when you first learned to type. This 30-year-old application forces users into a new mode of thinking, but the learning curve is well worth the end benefits of high-speed, mouse-free text editing.

Prerequisites

You need a basic understanding of the command line for this tutorial. You should understand what files and directories are and be able to log in to your account on a UNIX-like operating system.

System requirements

Access to a user account on any computer running any UNIX-like operating system is all you need to complete this tutorial. UNIX-like operating systems include the IBM® AIX® operating system, Linux®, Berkeley Software Distribution (BSD), Mac OS® X (using Terminal to access the command line), and many others.

Section 2. Introducing vi

The vi text editor uses two main modes: command mode and insert mode. The first part of this tutorial focuses on navigating a file, which is done in command mode. When you're in command mode, normal keypresses are used to execute commands rather than to create text. When you enter insert mode, the keyboard is used to enter text, for instance, on the command line. To exit command mode, press the **Esc** key.

Commands in vi can be either single keypresses, combined keypresses using Shift or Ctrl, or key sequences. Any time a capital letter is referenced for a command, you should use the Shift key combined with that letter. Any time a command is referenced that uses two letters or symbols, you press the keys in sequence rather than simultaneously.

To begin, you'll create a blank file by opening vi from the command line followed by the name of your new file. In this tutorial, you make a document from scratch in vi

and edit the document to learn useful vi commands. By the end of the tutorial, you'll have learned all the major vi commands that are used in day-to-day editing, as well as some powerful commands that you can leverage when the time is right. Throughout this tutorial, please feel free to use the reference sheet from Cal Tech in the Resources section.

Log in to your favorite UNIX-like operating system and open vi with a new file (see Figure 1).

Figure 1. Open vi with a new file

vi opens with a new blank file called tutorial.txt (see Figure 2). You'll immediately notice something odd: The far left column of the text editor is filled with tilde characters. Don't worry -- that's how vi illustrates undefined portions of the document. In other words, these lines don't exist yet, because the file has no content.

Figure 2. A blank file in vi

Before you do anything else, you should learn how to save a file and how to exit vi. These types of commands are entered by pressing the colon key followed by a sequence of letters describing your desired action. To save your new file, press the **colon** key, press the **w** key, and then press the **Enter** key. To quit vi, press the **colon** key, the **q** key, and the **Enter** key. Now, open vi again on the command line. If you try to exit vi without saving your changes, it warns you to press the **exclamation point** key to confirm your action. Any time you want to override a system warning, append your key sequence with an exclamation point; the key sequence to quit is:

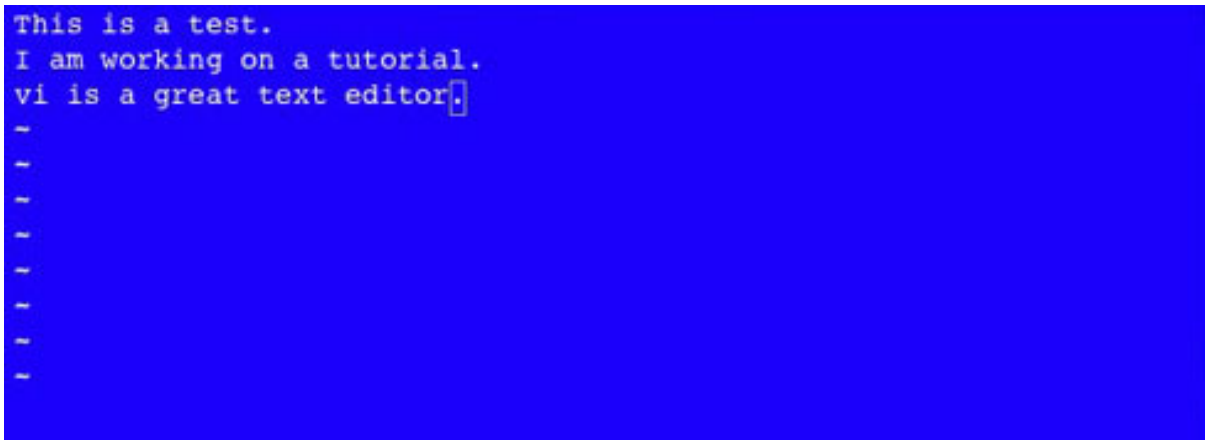
```
key, q key, ! key, Enter key
```

You can combine commands, such as `wq`, to save the file and then exit vi.

The first thing you need to do is insert some text so that you can learn to edit and manipulate your file. By default, you start in command mode. So, if you just start typing, you might be surprised by the results. Each mode is described in detail later; for now, press the **i** key to enter insert mode, and then type a few lines of text as you would in a normal text editor. When you're finished, press the **Esc** key to get back into command mode. See Figures 3 and 4.

Figure 3. When you enter insert mode, the bottom line of the editor indicates you have done so

Figure 4. Type these lines in your example document so that you have text to work with



After you press the **Esc** key, the cursor remains on the last character typed, and you're back in command mode. The next section teaches you how to navigate the file.

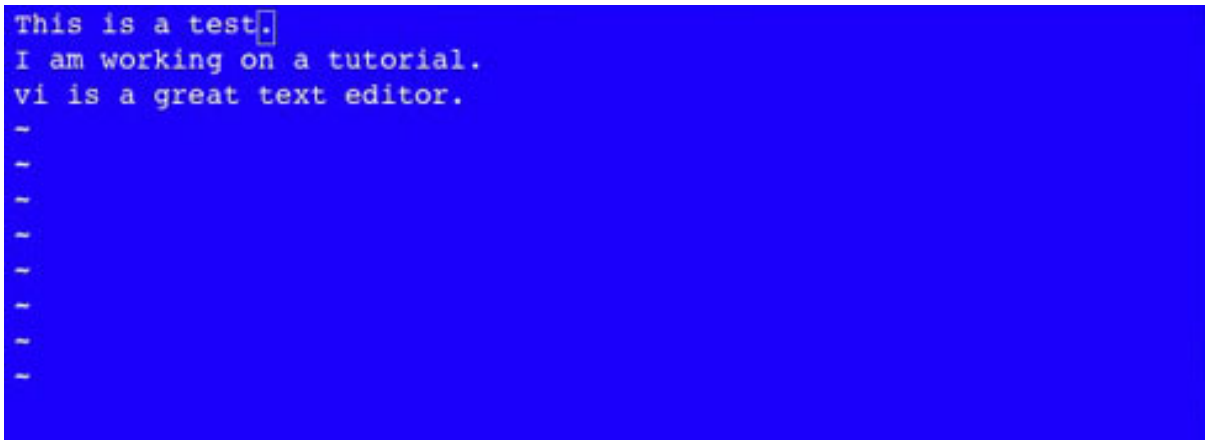
Section 3. Navigating in vi

In command mode, your keyboard becomes an interface tool, as opposed to a text-input tool. vi is designed for users that need access to all common commands while keeping their hands on the home keys (a-s-d-f and j-k-l-;) and reaching to nearby letters. The first basic operation to learn is moving the cursor. Most modern versions of vi allow you to use the arrow keys on the keyboard, but advanced vi users prefer the easily accessible keys under their fingertips, **h-j-k-l**:

- **h** and **l** represent left and right, respectively, which is intuitive because they bound the four navigation keys on the left and on the right.
- **k** moves the cursor up.
- **j** moves the cursor down.

You'll learn these keys quickly with muscle memory. To move the cursor to the first line of the three-line file you created, press **k** two times. The cursor now sits at the end of first line. Go ahead and use **h-j-k-l** to move the cursor to other locations in the file, but then bring it back to the end of line one. See Figure 5.

Figure 5. Move the cursor back to the first line of the file with the h-j-k-l directional keys



Moving the cursor with shortcuts

Moving within a line

Now that your cursor is at the end of first line, you might want to move it to the beginning of the line, but you don't want to click **h** over and over again to get there. In command mode, vi has keyboard-based shortcuts that let you navigate rapidly to various locations in the file more quickly than you could by moving your hand over to the mouse and pointing to the location in the file you want to go to, or using arrow keys to traverse spaces one by one. The first such shortcut is 0 (zero):

- To go to the beginning of a line, press **0**; your cursor jumps to that location.
- To go back to the end of the line, press **$**.

Go ahead and try it.

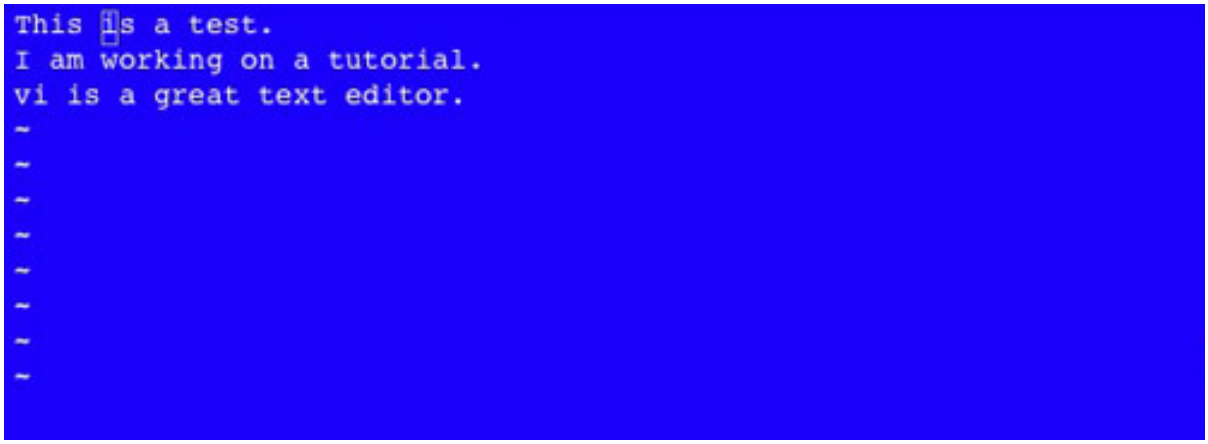
Now you can move one character at a time. You can also move to the beginning and end of each line, but those are two widely varying granularities. Another option is to navigate word by word. You can do so with the w and b keys:

- Pressing **w** moves forward one word.
- Pressing **b** moves back one word.

Try it by moving to the beginning of the first line (by pressing 0) and then move to the beginning of the word *test* by pressing **w** three times. Then, press **b** twice to get back to the word *is*.

You probably noticed that the w key and the b key set the cursor to the beginning of each word. You can also navigate to the end of each word by using the e key to move forward, or by pressing **g**. Press **e** to go backward. See Figure 6.

Figure 6. Moving the cursor word by word with w and b is a good way to quickly navigate to a word you misspelled or would like to change



Moving from line to line

Now, you can quickly move within a line but, as you deal with larger files, it's also important to rapidly navigate from line to line. There are a few ways to do this in vi. You can use the up and down commands (`k` and `j`), or you can use the `page-up` and `page-down` commands. Most commands in vi don't require you to press the **Ctrl** key, but the `page-up` and `page-down` commands are a couple of exceptions to this loose rule:

- Press **Ctrl-u** to go up a page.
- Press **Ctrl-d** to go down a page.

To quickly navigate to the beginning or end of a file, you can use gg or G:

- Pressing **gg** puts the cursor on the first line of the document.
- Pressing **G** puts the cursor on the last line of the document.

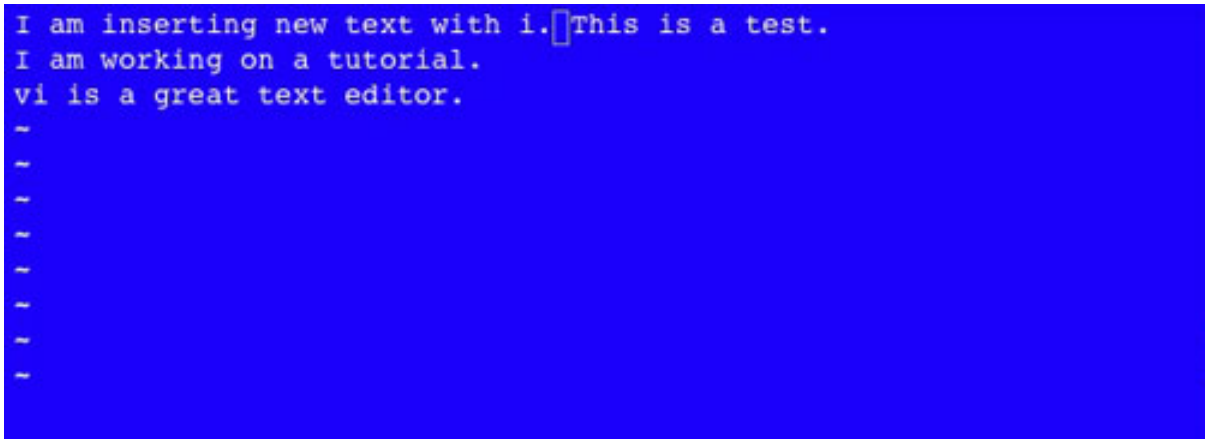
Additional methods specify line numbers; these are discussed in the Using numbers to preface vi commands section.

Section 4. Inserting and editing in vi

While navigating the document, you remain in command mode and use the keyboard as an interface tool to move the cursor. The next set of command mode keys provide different ways to enter insert mode, which is used to input new text into your file. You used the most basic way to enter insert mode at the beginning of this tutorial when you entered your initial text: You pressed the **i** key, which stands for *insert*. Pressing **i** puts you into insert mode in front of the current cursor location.

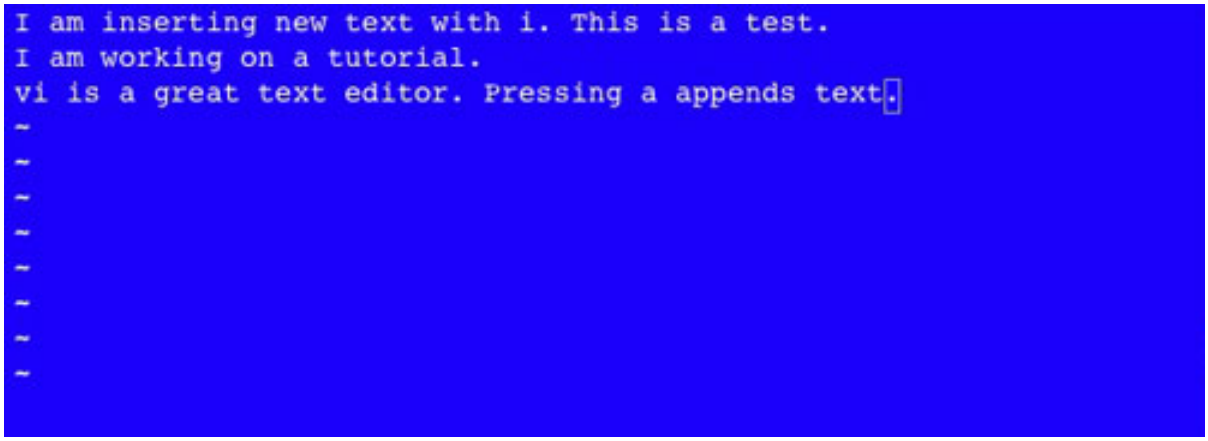
For instance, if you navigate to the beginning of the file by pressing **gg,** you can press **i**, which makes any text you enter appear prior to the text on the current line. Press **gg**, **i**, and then type `I am inserting new text with i`. Press the **Esc** key when you're finished to go back to command mode. Remember, you must go back into command mode after entering new text, or there is no way to navigate the document. See Figure 7.

Figure 7. The simplest way to add new next is to use the i key to enter insert mode



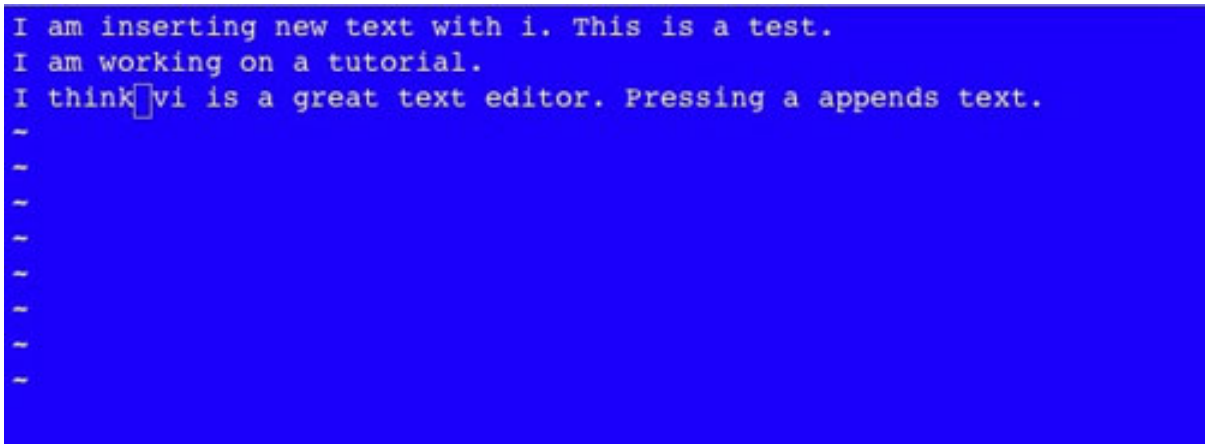
The other basic method of entering new text is to use the a key, which stands for *append*. Using the a key puts you into insert mode, but it adds text after the current location of the cursor instead of before the current location of the cursor. To test this, go to the last line of your document and press the **G** and **$** key to go to the end of the line. Then press the **a** key, type `Pressing a appends text`, and press the **Esc** key to return to command mode. See Figure 8.

Figure 8. Another way to insert text is with the a key, which stands for append



Now your cursor is at the period on the last line of your file. If you press the **i** key now, you'd insert text just before the period. If you press the **a** key, you'd insert text just after the period. By pressing the **I** key (the capital letter), you can start your input at the beginning of the line, even though your cursor is at the end of the line. Similarly, if you press the capital **A** key, you can input text at the end of the line regardless of the cursor position. To test this, press **I**, type `I think` , and then press the **Esc** key. See Figure 9.

Figure 9. To insert text at the beginning of a line (regardless of cursor position), press the I key

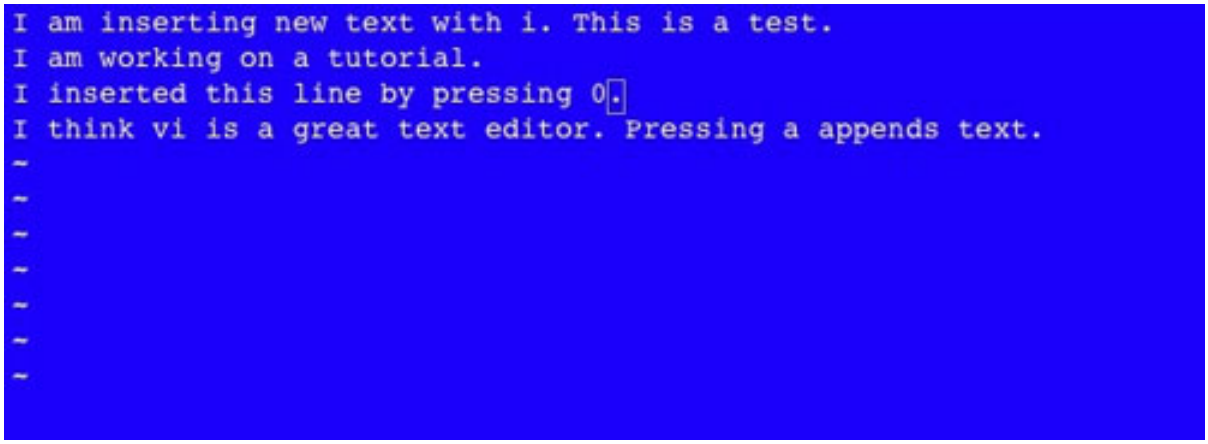


Another useful way of inserting new text is to simultaneously enter insert mode and add a new line to your text file. Just like the normal text insertion, this can be done before or after the cursor location:

- To insert a new line above the current cursor location, press the **O** key.
- To insert a new line below the current cursor location, press the **o** key.

To try this command, press **O**, type `I inserted this line by pressing O`, and then press the **Esc** key to return to command mode. See Figure 10.

Figure 10. To insert text on a new line before the cursor, press the O key



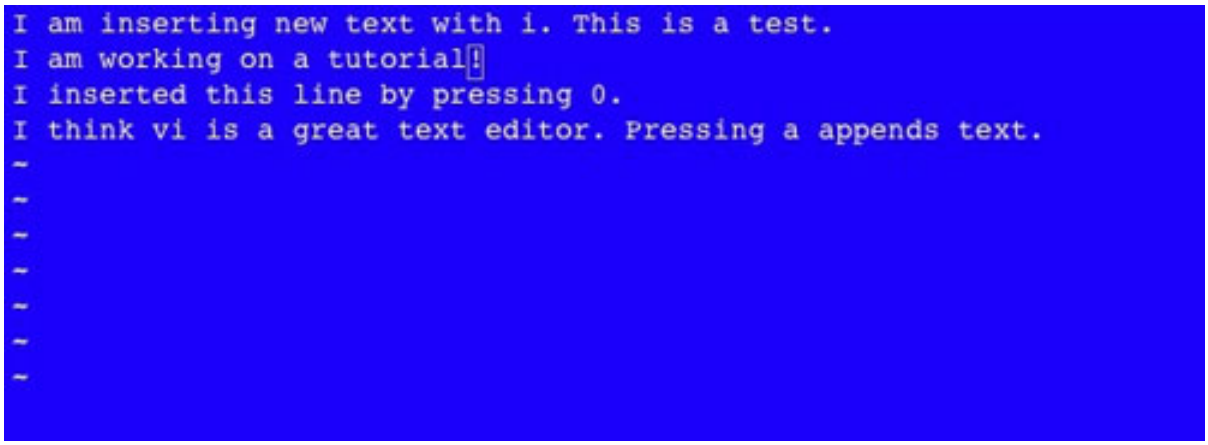
You've now used the major methods of inserting new text. To recap, the keys are a, i, A, I, o, and O. Can you remember what each one does? Don't worry, even if you don't, they will come natural to you after a few days of use.

Replacing text

Now that you've typed a few items into your tutorial.txt file, you might find that you've made some errors, or you might want to change your choice of words. Before learning how to delete text, you should learn how to replace text. Why? Because if you learn to add new text and delete old text, you might fall into the habit of using the delete commands when a `replace` command would be more efficient. It's quicker to replace a word in a one-step process than it is to delete a word and then add a new word in its place.

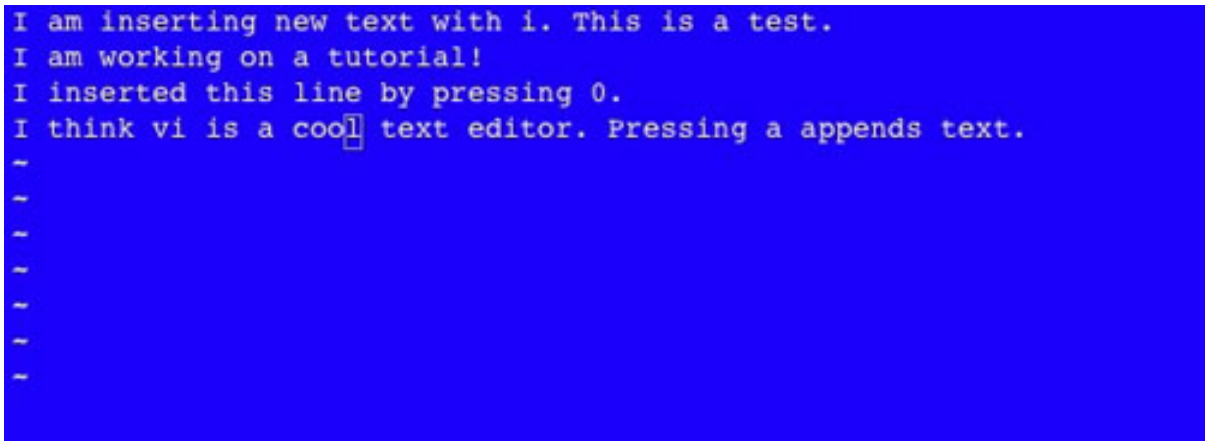
vi uses two important replace commands. The first is the r key, which removes the character the cursor is focused on and puts you in insert mode for a single replacement character. In other words, you can make one keypress after pressing **r**. As a result, vi automatically returns to command mode (without pressing the **Esc** key). To try this, use the k key and the l key to navigate up to the end of second line. Your cursor should be focused on a period. To change the period to an exclamation point, press the **r** key and then press the **!** key. See Figure 11.

Figure 11. To replace a single character, press the r key and then type the character you want in the document



An even more useful `replacement` command is executed by pressing the **c** key and then the **w** key, which together stand for *change word*. This command deletes the current word and puts you in insert mode so that you can immediately start typing a replacement word. In this case, you need to press the **Esc** key when you're finished typing the new word to let vi know you're done. Move down to the word *great*, press the **c** and **w** key, type `cool`, and then press the **Esc** key. See Figure 12.

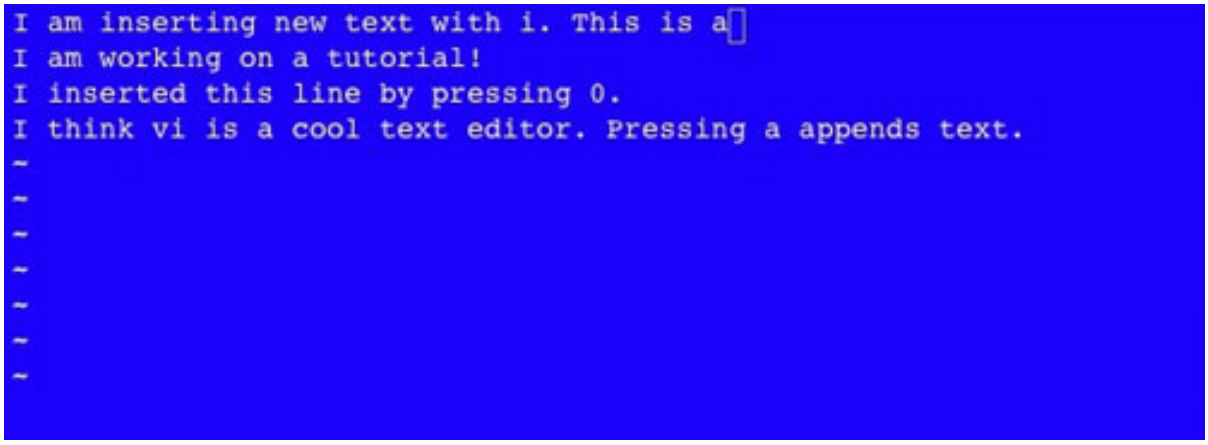
Figure 12. To replace a word, press the c and w key and then type the new word



If you want to remove text altogether rather than replacing it, you need to use a `delete` command. As with many other things in vi, you have multiple choices, depending on how much data you want to delete at once. The most basic `delete` command is the x key, which deletes one character at a time. To try it, go back to the beginning of your text file and press the **Ctrl-u (page-up)** key. Press the **$** key to go to the end of the line and then press the **x** key five times to delete the *test.* See Figure 13.

Deleting text

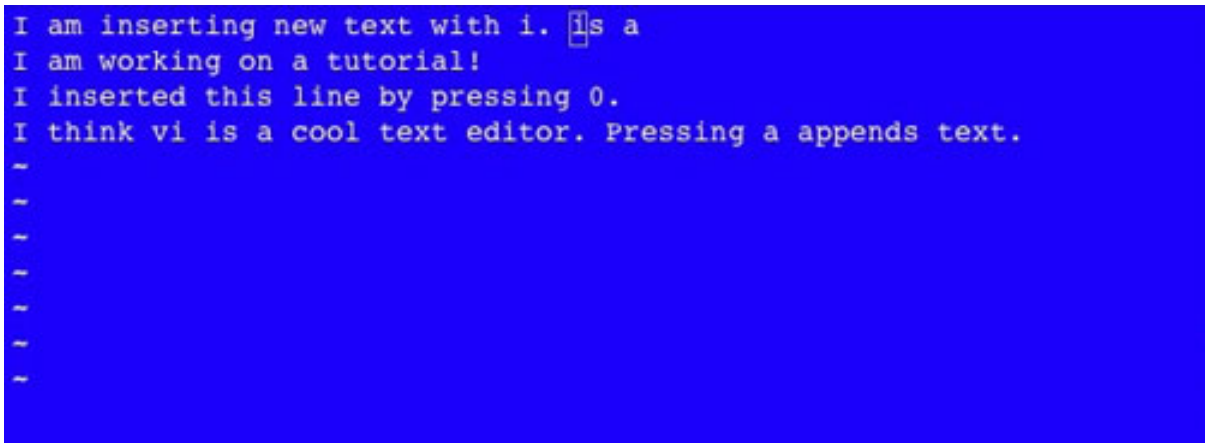
Figure 13. To delete individual characters, use the x key



Pressing the **x** key five times did the job, but you've probably gathered that vi has a quick way of doing just about everything. You can delete a word with one command by pressing the **d** key and **w** key, which together stand for *delete word*. Navigate

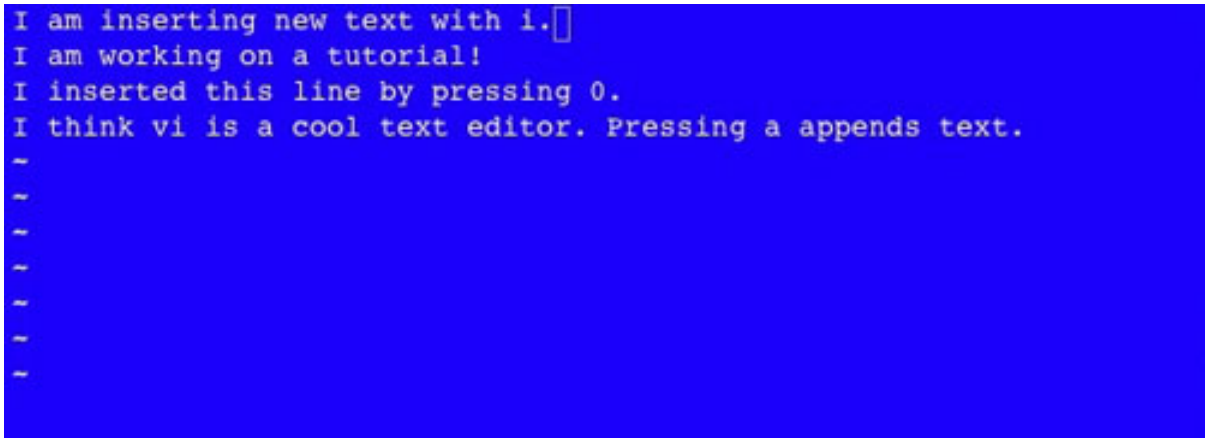
back to the word *This* by pressing the **b** key three times. To delete the word, press the **d** key and **w** key. See Figure 14.

Figure 14. To delete individual words, use the dw command



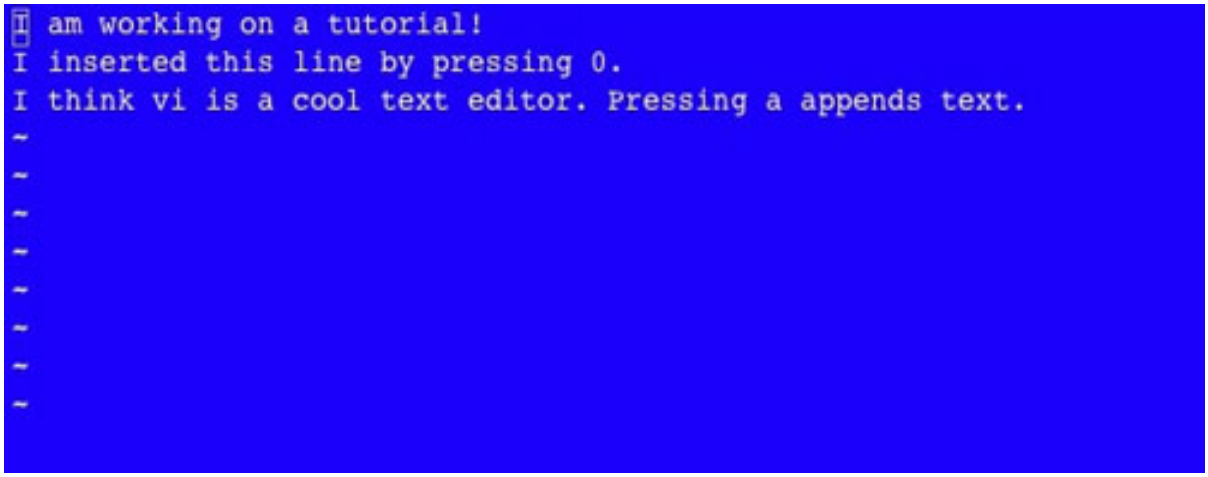
You're left with a hanging, partial sentence at the end of the first line. To delete everything on a line that follows your cursor position, you can use the d key with the $ key, which you've learned is used for end-of-line operations. Press the **d** key and the **$** key to remove *is a* from the end of the line. See Figure 15.

Figure 15. To delete from the cursor to the end of a line, use d$



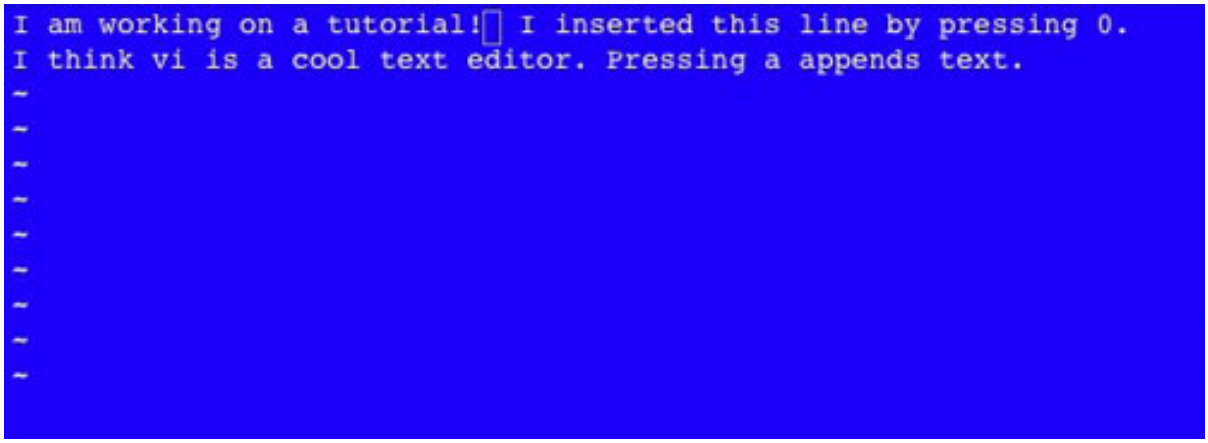
The final commonly used deletion method in vi is the `delete-line` command, which is accomplished by pressing the **d** key twice. It deletes the text on the line and brings the next lower line up, so you don't have an empty line in the document. To remove the first line of your file, press the **d** key and then press the key again. See Figure 16.

Figure 16. To delete an entire line, press dd.



Speaking of bringing up a line, you can use the `join` command by pressing the **J** key, which brings the line below the cursor up to the line the cursor is on, without deleting any text. Your cursor should be on the new line (line 1) of your document. Press the **J** key to bring line 2 up to line 1. See Figure 17.

Figure 17. To bring two lines together in vi, press the J key



Section 5. Getting fancy

You should now have the basic skills to create, navigate, and edit a text file in vi. After you get familiar with the basic commands, you can create and edit files about as quickly as you can in a more conventional text editor. A few things are missing from your repertoire, though. This section teaches you how to cut, copy, and paste. You can do multiple iterations of a command at one time, repeat commands, search the document, and use undo and redo. This editor meets the major functionality found in other text editors and uses the fast-access keyboard command style.

Cut, copy, and paste

In vi, any time a piece of text is deleted, it's automatically stored in a buffer (like the

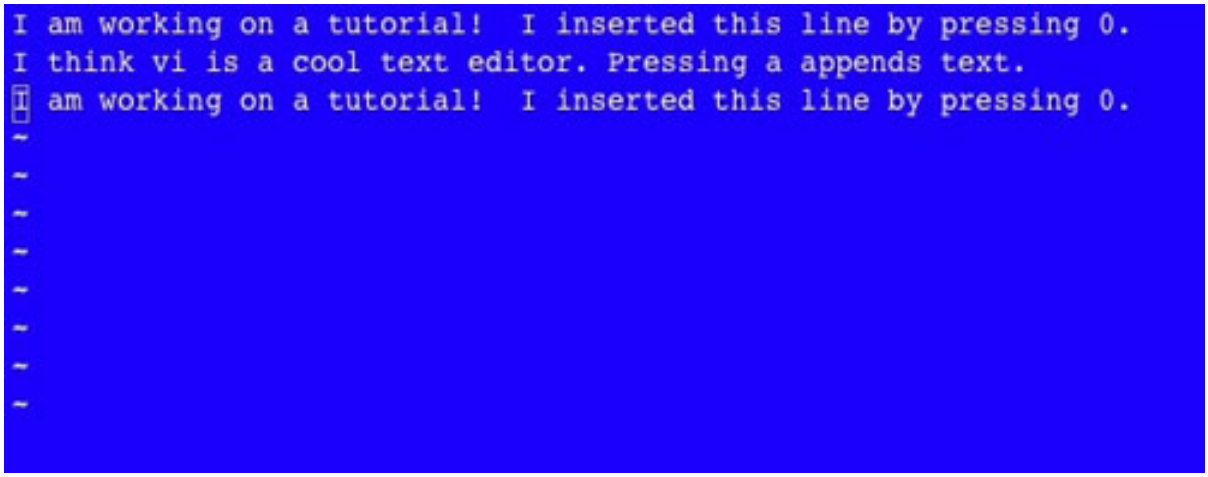
clipboard in Windows). You already know how to do a cut command by using `x`, `dd`, `dw`, and `d$`. Similar commands are available to copy data without deleting it -- this process is called *yank* in vi -- :

- Press the **y** key twice to copy a whole line.
- Press the **y** key and **w** key to copy an individual word.
- Press the **y** key and **$** key to copy a line starting at the current cursor location.

Copying data isn't much use without knowing how to paste it. So, before testing these commands, you should learn the `paste` command, which is enacted with the p key. Like many other commands in vi, a lowercase p key pastes data after the cursor location, whereas an uppercase P key pastes data before the cursor position.

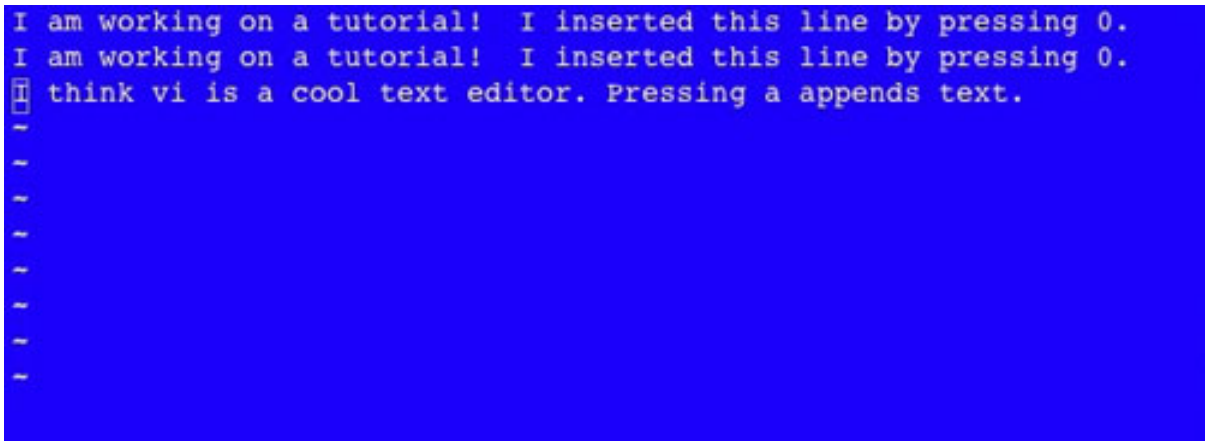
To copy and paste, navigate to the first line of your text file and press the **y** key twice. Then, move the cursor down to line 2 and press the **p** key once. Doing so creates a copy of line 1 on line 3. See Figure 18.

Figure 18. To copy a line, use the yy command and paste using the p command.



Try doing a cut and paste by moving the cursor to the second line and press the **d** key twice. Then, press the **p** key to paste the line below line 2. See Figure 19.

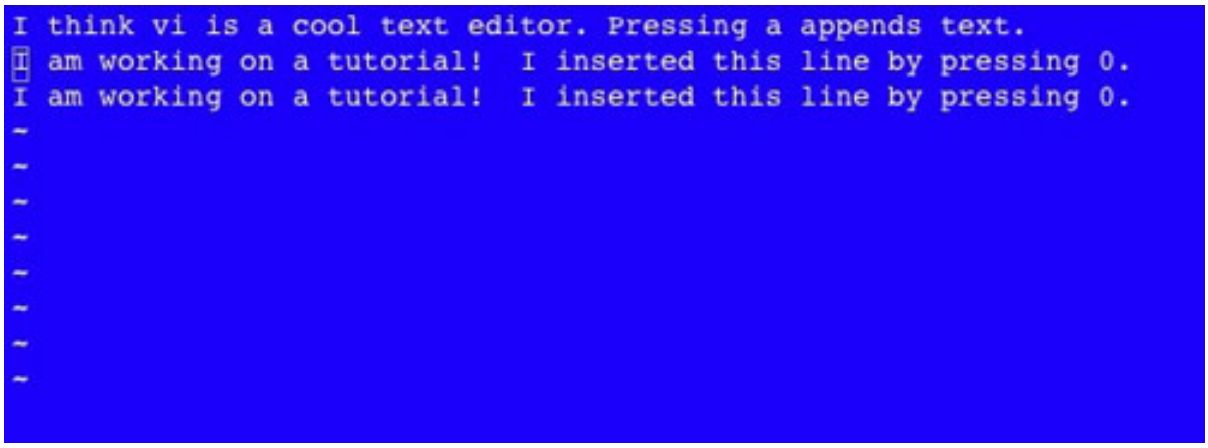
Figure 19. To cut and paste a line, use the dd command and the p command



Using numbers to preface vi commands

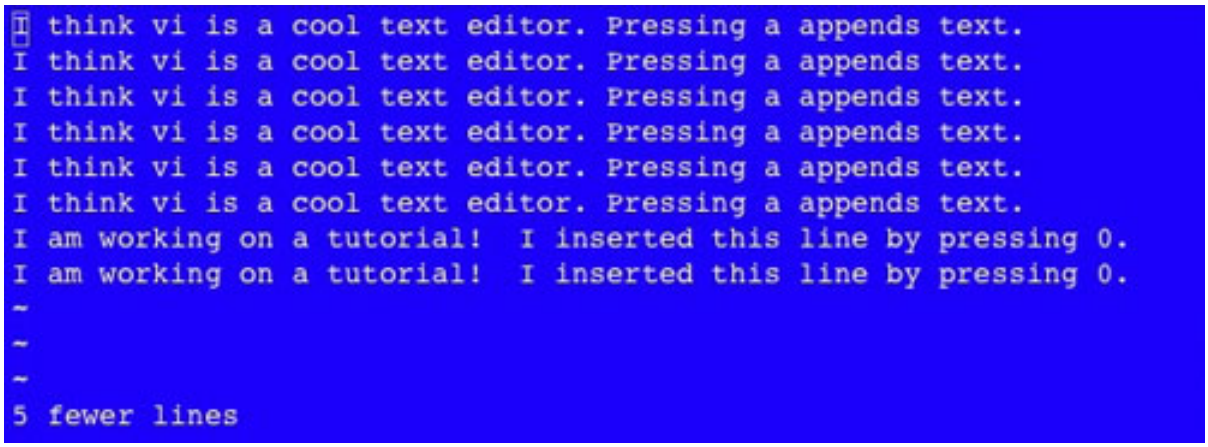
At this point, you might wonder how to do some of these commands on more than one piece of data at a time. For instance, you probably often copy and paste entire paragraphs, as opposed to single lines. vi lets you preface just about every command in the application with a number, which causes the command to operate multiple times at once. This is an extremely powerful, important part of what makes vi a great editor for power users. To try a simple example of cutting and pasting two lines at a time, navigate to line 1 of the file, press the **2** key, press the **d** key twice, and then press the **p** key. See Figure 20.

Figure 20. Cut and paste two lines at a time by using a number 2 preceding the dd command and the p command



The same concept can be used during the `paste` command to replicate a line multiple times. To do so, go back to line 1, copy the line with the `yy` command, and then press **10** before pressing the **p** key. Now you have 10 more lines of comments about vi. Before moving on, press the **5** key and then press the **d** key twice to remove some of the extra lines. See Figure 21.

Figure 21. Paste multiple lines by preceding the p command with the number 10; then delete some of them with the number 5 preceding dd



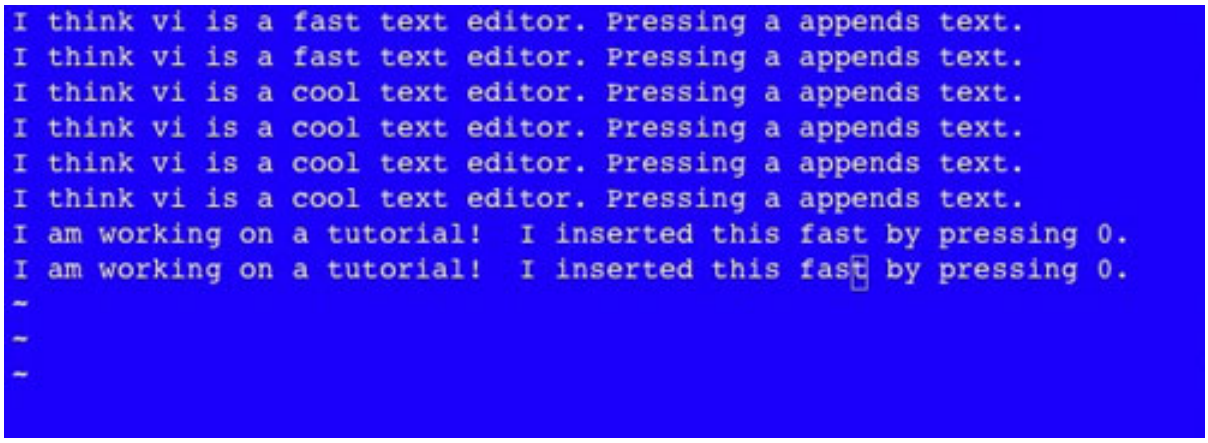
Experiment with numbers in front of navigation commands as well. For instance, pressing the **30** key and then the **l** key moves the cursor 30 spaces to the right. Pressing the **7** key and then the **G** key moves the cursor to line 7 in the file. Pressing

the **5** key and then the **w** key moves the cursor to the fifth word. When you're finished experimenting, you can move on to the next step.

Repeating vi commands

Another useful command in vi is executed with the **.** (period) key. The **.** key repeats the last command, which is an important feature for getting work done quickly. For instance, navigate the cursor to the word *cool* on the first line, and then use the `cw` command to change the word to *fast*. Don't forget to press the **Esc** key when you're finished typing the word *fast*. Move down to another instance of the word *cool*, and then press the **.** key to change that word to *fast*. You can also move to the word *line* on the last two lines of your file and replace those with the **.** key. See Figure 22.

Figure 22. Repeat commands with the period key, which lets you produce document edits quickly



Searching text in vi

Searching text in vi is also fast and efficient. To start searching for a string, press the ***/*** key (slash key) followed by the string you want to search for and then press the **Enter** key. To combine some of your vi skills, press the ***/*** key, type `think`, press **Enter**, and then use the `cw` command to change the word to *know*. Don't forget to press the **Esc** key when you're finished.

If you want to do the same thing on line 2, press the **n** key to find the next instance of *think*, and then press the **.** key to change the word to *know*. The ? key does a search just like the / key, but it searches the document backward instead of forward. After you've replaced *think* with *know*, press the **?** key followed by the word *fast* to search back to it. See Figures 23 and 24.

Figure 23. Search for strings by using the slash key followed by the string you're searching for

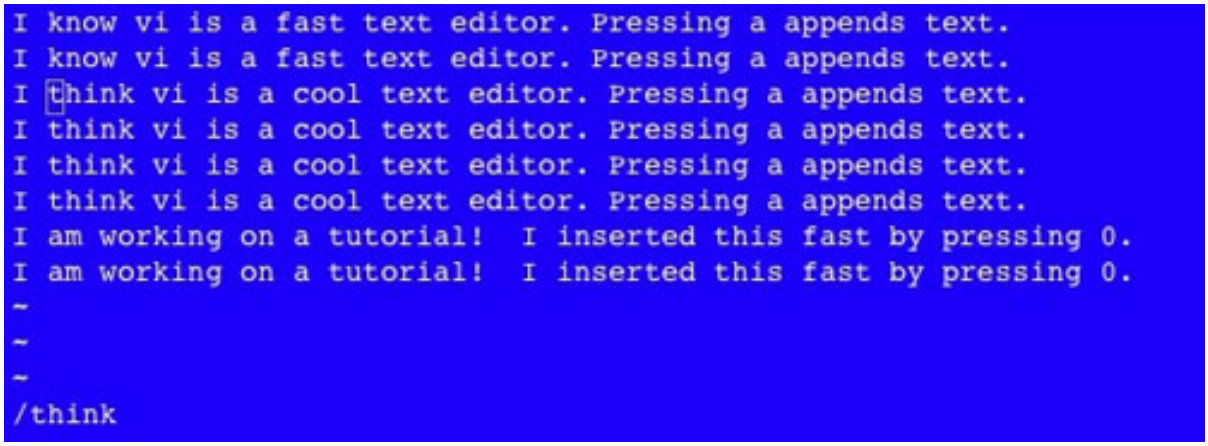
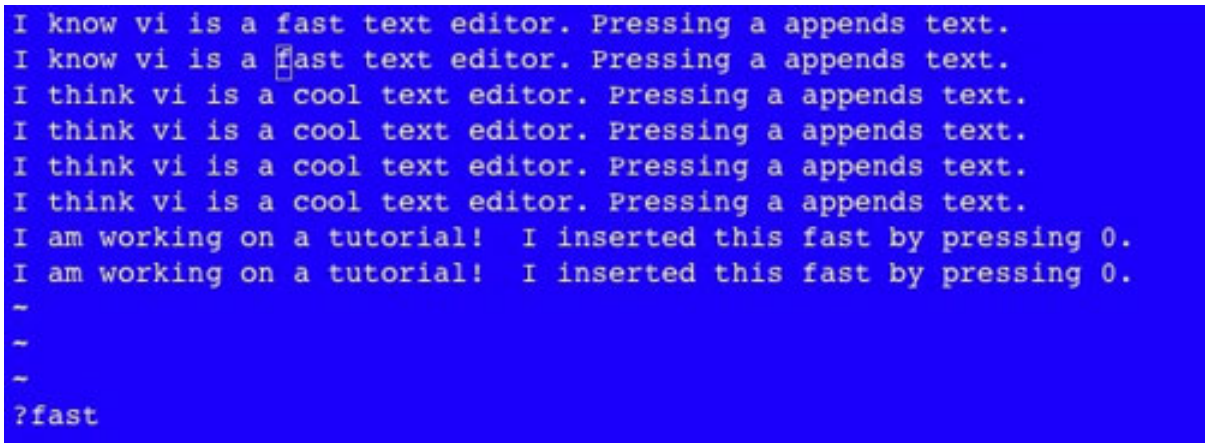


Figure 24. Search for strings backward by using the ? key followed by the string you're searching for

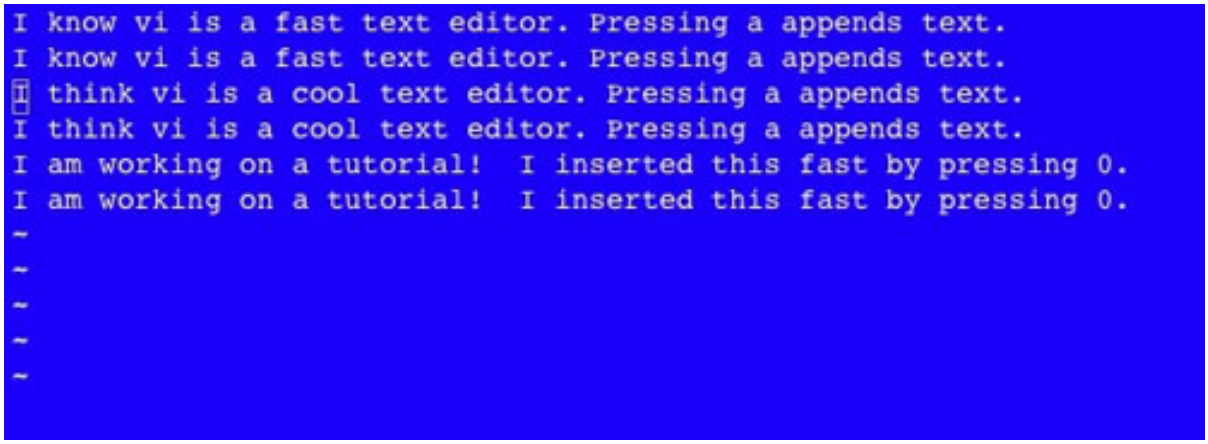


Undo and redo in vi

If you make any mistakes, vi has the modern convenience of undo and redo to make sure you can restore your document to the proper state. Undo is accomplished by pressing the **u** key in command mode, and redo is executed by pressing **Ctrl-r** in command mode. Try undo and redo, as follows (see Figure 2):

1. Go to line 3 in your text file and remove a few lines.
2. Press the **3** key and the **G** key to go to line 3.
3. Press the **2** key and the dd command to delete two lines.
4. Oops! That was a mistake, and you want to get the two lines back. To do so, press the **u** key to undo the previous command.
5. If you change your mind and want the two lines removed, press **Ctrl-r** to redo the command.

Figure 25. To undo a command, use the u key; to redo a command, press Ctrl-r



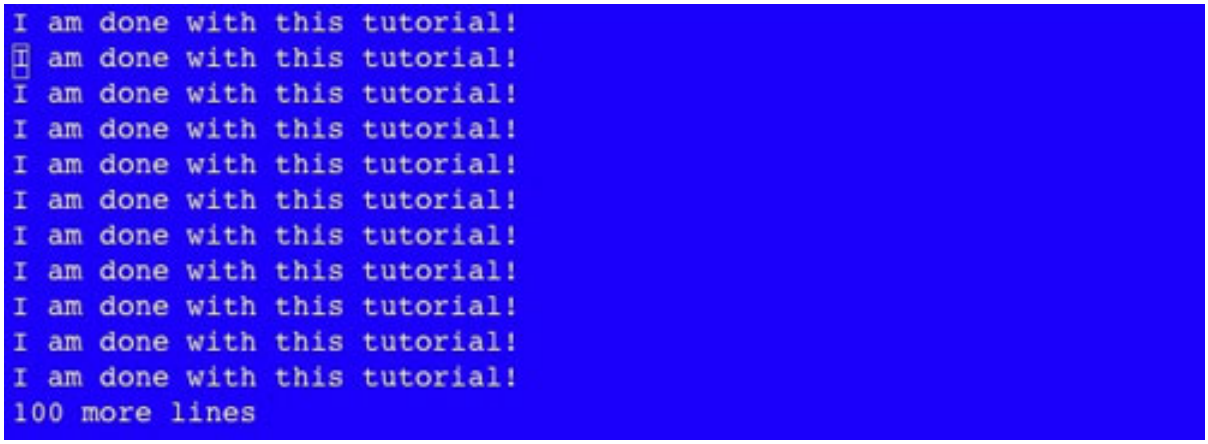
Section 6. Wrap-up

To wrap up, run though a last sequence of commands to combine some of the things you've learned (see Figure 26):

1. To go to the beginning of your document, press the **g** key twice.
2. To delete everything in your file (because you have less than 100 lines), type `100` followed by the `dd` command.
3. Press the **i** key to go into insert mode.
4. Type `I am done with this tutorial!.`
5. Press the **Esc** key.
6. Press the **y** key twice, type **100**, and then press the **p** key.

You've now yelled to your computer 100 times that you're done! Good job; time for a break and a cup of coffee.

Figure 26. You're finished with this tutorial



After working through this tutorial, you should have the knowledge to create and edit files with vi. Experiment with the commands listed throughout this tutorial, and practice them by beginning to use vi as your day-to-day text editor. It will slow you down at first but, in a short time as you memorize the commands and learn when and where to use them, vi can significantly increase your productivity.

Future tutorials in the series will cover shell tricks and tips. In the meantime, keep working with the command line and practicing with vi -- you'll be a power UNIX user in no time!

Resources

Learn

- UNIX tips and tricks for a new user: Check out other parts in this series.
- vi reference sheet: Explore this reference sheet and keep it handy during the tutorial.
- Paper for freebsd.org: Bill Joy, the creator of the original version of vi, wrote this paper, which includes a complete reference of vi commands.
- vim.org: Learn more about vi from this site. vim is a modern version of vi, which is included in many Linux and UNIX-like operating systems.
- Bram Moolenaar: Learn more about vi from one of vim's developers.
- vi Lover's homepage: This site has extensive version information and links to complete vi references.
- Unix.org: The Open Group's site can teach you more about UNIX and UNIX certifications.
- Linux.org: Learn more about Linux, an open source UNIX-like operating system.
- AIX and UNIX: Visit the developerWorks AIX and UNIX zone to expand your UNIX skills.
- New to AIX and UNIX: Visit the New to AIX and UNIX page to learn more about AIX and UNIX.
- developerWorks technical events and webcasts: Stay current with developerWorks technical events and webcasts.
- AIX 5L Wiki: A collaborative environment for technical information related to AIX.
- Podcasts: Tune in and catch up with IBM technical experts.

Get products and technologies

- IBM trial software: Build your next development project with software for download directly from developerWorks.

Discuss

- Participate in the AIX and UNIX forums:
  - AIX 5L -- technical
  - AIX for Developers Forum
  - Cluster Systems Management
  - IBM Support Assistant
  - Performance Tools -- technical

- Virtualization -- technical
- More AIX and UNIX forums

- Participate in the developerWorks blogs and get involved in the developerWorks community.

About the author

Tim McIntire
Tim McIntire works as a consultant and co-founder of Cluster Corporation, a market leader in HPCC software, support, and consulting. He also contributes periodically to IBM developerWorks and Apple Developer Connection. Tim's research, conducted while leading the computer science effort at Scripps Institution of Oceanography's Digital Image Analysis Lab, has been published in a variety of journals, including *Concurrency and Computation* and *IEEE Transactions on Geoscience and Remote Sensing.* You can visit TimMcIntire.net to learn more.