Learn Linux, 101: File editing with vi

The editor that's always there

Skill Level: Intermediate

Ian Shields (ishields@us.ibm.com)
Senior Programmer
IBM

10 Feb 2010

Learn how to use the vi editor, found on almost every UNIX® and Linux® system. You can use the material in this article to study for the LPI 101 exam for Linux system administrator certification, or just to learn for fun.

About this series

This series of articles helps you learn Linux system administration tasks. You can also use the material in these articles to prepare for Linux Professional Institute Certification level 1 (LPIC-1) exams.

See our series roadmap for a description of and link to each article in this series. The roadmap is in progress and reflects the latest (April 2009) objectives for the LPIC-1 exams: as we complete articles, we add them to the roadmap. In the meantime, though, you can find earlier versions of similar material, supporting previous LPIC-1 objectives prior to April 2009, in our LPI certification exam prep tutorials.

Overview

In this article, learn the basic use of the vi editor, which is almost always available on any Linux or UNIX system. Learn to:

- Navigate a document using vi
- Use basic vi modes

- Insert, edit, delete, copy, and find text

This article helps you prepare for Objective 103.8 in Topic 103 of the Linux Professional Institute's Junior Level Administration (LPIC-1) exam 101. The objective has a weight of 3.

Prerequisites

To get the most from the articles in this series, you should have a basic knowledge of Linux and a working Linux system on which you can practice the commands covered in this article. Sometimes different versions of a program will format output differently, so your results may not always look exactly like the listings and figures shown here.

Navigating documents with vi

Connect with Ian
Ian is one of our most popular and prolific authors. Browse all of Ian's articles on developerWorks. Check out Ian's profile and connect with him, other authors, and fellow readers in My developerWorks.

The vi editor is almost certainly on every Linux and UNIX system. In fact, if a system has just one editor, it's probably vi, so it's worth knowing your way around in vi. This article introduces you to some basic vi editing commands, but for a full vi tutorial, check out our tutorial on vi "vi intro -- the cheat sheet method", or consult the man pages or one of the many excellent books that are available.

Starting vi

Most Linux distributions now ship with the vim (for **V**i **IM**proved) editor rather than classic vi. Vim is upward compatible with vi and has a graphical mode available (gvim) as well as the standard vi text mode interface. The `vi` command is usually an alias or symbolic link to the vim program. There are several versions of vim: tiny, small, normal, big, and huge. You can find out what version of vim you are running and what features are included by using the command:

```
vi --version
```

If you recall the section on changing priorities in a previous article "Learn Linux, 101: Process execution priorities," we wanted to change the priority of our running count1.sh shell script. Perhaps you tried this yourself and found that the command ran so fast that you didn't have enough time to accomplish the priority change with `renice`. So let's start by using the vi editor to add a line at the beginning of the file to sleep for 20 seconds so we have some time to change priorities.

If you don't still have the count1.sh program around, open a terminal window in your home directory and paste in the commands from Listing 1. This will create count1.sh in a directory called lpi103-8 and place you in that directory.

Listing 1. CPU-intensive script - count1.sh

```
mkdir -p lpi103-8 && cd lpi103-8 && {
echo 'x="$1"'>count1.sh
echo 'echo "$2" $(date)'>>count1.sh
echo 'while [ $x -gt 0 ]; do x=$(( x-1 ));done'>>count1.sh
echo 'echo "$2" $(date)'>>count1.sh
}
```

To edit an existing file, use the `vi` command with a filename as a parameter. See the man pages or Resources for details on the many options that are available. For now, just use the command without options:

```
vi count1.sh
```

This should open the count1.sh file. You should see a display similar to Listing 2. If you are using vim, some of the words or characters may be in color. Vim has a syntax highlighting mode (which was not part of the original vi editor), and it may be turned on by default on your system.

Listing 2. Editing count1.sh using vi

```
x="$1"
echo "$2" $(date)
while [ $x -gt 0 ]; do x=$(( x-1 ));done
echo "$2" $(date)
~
~
~
~
"count1.sh" 4L, 84C                                          1,1           All
```

The vi editor dates from the time when not all terminal keyboards had cursor movement keys, so everything you can do in vi can be done with the keys typically found on a standard typewriter plus a couple of keys such as **Esc** and **Insert**. However, you can configure vi to use additional keys if they are available; most of the keys on your keyboard do something useful in vi. Because of this legacy and the slow nature of early terminal connections, vi has a well-deserved reputation for using very brief and cryptic commands. Let's start by looking at the keystrokes for navigation around your file.

Moving around

These commands help you move around in a file:

h

Move left one character on the current line

j
Move down to the next line

k
Move up to the previous line

l
Move right one character on the current line

w
Move to the next word on the current line

e
Move to the next end of word on the current line

b
Move to the previous beginning of the word on the current line

Ctrl-f
Scroll forward one page

Ctrl-b
Scroll backward one page

If you type a number before any of these commands, then the command will be executed that many times. This number is called a *repetition count* or simply *count*. For example, 5h will move left five characters. You can use repetition counts with many vi commands.

Moving to lines

The following commands help you move to specific lines in your file:

G
Moves to a specific line in your file. For example, 3G moves to line 3. With no parameter, G moves to the last line of the file.

H
Moves relative to the top line on the screen. For example, 3H moves to the line currently 3rd from the top of your screen.

L
Is like H, except that movement is relative to the last line on screen. Thus, 2L moves to the second-to-last line on your screen.

Practice these commands until you are comfortable moving around the file. If you get stuck and things aren't working as expected, read on and learn how to get out of the file.

Getting out of vi

One of the most useful things to know about a new editor is how to get out of it before you do anything you shouldn't do, such as destroying an important configuration file. You can get out of vi by saving or abandoning your changes, or by restarting from the beginning. If these commands don't seem to work for you, you may be in insert mode, which you will learn about in a moment. If in doubt, pressing **Esc** will leave insert mode and return you to command mode where these commands should work.

:q!
: Quit editing the file and abandon all changes. This is a very common idiom for getting out of trouble.

:w!
: Write the file (whether modified or not). Attempt to overwrite existing files or read-only or other unwritable files. You may give a filename as a parameter, and that file will be written instead of the one your started with. It's generally safer to omit the ! unless you know what you're doing here.

ZZ
: Write the file if it has been modified. Then exit. This is a very common idiom for normal vi exit.

:e!
: Edit the current disk copy of the file. This will reload the file, abandoning changes you have made. You may also use this if the disk copy has changed for some other reason and you want the latest version.

:!
: Run a shell command. Type the command and press **Enter**. When the command completes, you will see the output and a prompt to return to vi editing.

Notes:

1. When you type the colon (:), your cursor will move to the bottom line of your screen where you can type in the command and any parameters.
2. If you omit the exclamation point from the above commands, you may receive an error message such as one saying changes have not been saved, or the output file cannot be written (for example, you are editing a read-only file).

3. The : commands have longer forms (:quit, :write, :edit), but the longer forms are seldom used.

vi modes

The vi editor has two modes of operation:

Command mode
: In command mode, you move around the file and perform editing operations such as searching for text, deleting text, changing text, and so on. You usually start in command mode.

Insert mode
: In insert mode, you type new text into the file at the insertion point. To return to command mode, press the **Esc** (Escape) key.

These two modes determine the way the editor behaves. Anything you type in insert mode is considered text to be inserted into the file. If you are trying to type a command and nothing happens, or the character appears under the cursor, then you probably forgot to press **Esc** to escape from insert mode.

Editing text

Now that you can open a file in vi, move around it and get out, it's time to learn how to edit the text in the file.

Modifying text

Use the following commands when you need to insert, delete, or modify text. Note that some of these commands have an uppercase form that is similar to the lowercase form; see the descriptions below.

i
: Enter insert mode before the character at the current position. Type your text and press **Esc** to return to command mode. Use I to insert at the beginning of the current line.

a
: Enter insert mode after the character at the current position. Type your text and press **Esc** to return to command mode. Use A to insert at the end of the current line.

c
: Use c to change the current character and enter insert mode to type

replacement characters.

o
: Open a new line for text insertion below the current line. Use O to open a line above the current line.

cw
: Delete the remainder of the current word and enter insert mode to replace it. Use a repetition count to replace multiple words. Use c$ to replace to end of line.

dw
: Same as for cw (and c$) above, except that insert mode is not entered.

dd
: Delete the current line. Use a repetition count to delete multiple lines.

x
: Delete the character at the cursor position. Use a repetition count to delete multiple characters.

p
: Put the last deleted text after the current character. Use P to put it before the current character.

xp
: This combination of x and p is a useful idiom. This swaps the character at the cursor position with the one on its right.

Searching text

You can search for text in your file using regular expressions:

/
: Use / followed by a regular expression to search forward in your file.

?
: Use ? followed by a regular expression to search backward in your file.

n
: Use n to repeat the last search in either direction.

You may precede any of the above search commands with a number indicating a repetition count. So 3/x will find the third occurrence of x from the current point, as will /x followed by 2n. Similarly, 2/^e will find the second line from the current position that starts with e.

Note that search will wrap around to the top once the bottom of file is reached.

Getting help

Another useful command in vi is the help command, which you invoke by typing `:help`. Help will open inside vi; use the `:q` command to leave help and go back to your work. If you want help on some particular topic, say wrapping of lines, try adding a word after the :help command, for example: `:help wrap`.

Putting it together

We began by wanting to add a line to our count1.sh file. To keep the original and save the modified version as count2.sh, we could use these vi commands once we open the file with `vi`. Note that <Esc> means to press the **Esc** key.

Listing 3. Editor commands to add a line to count1.sh

```
1G
O
sleep 20<Esc>
:w! count2.sh
:q
```

These commands do the following:

1G
: Move to the first line of the file

O
: Open a new line above it and enter insert mode

sleep 20
: The new text that you want to add

<Esc>
: Press the **Esc** key to return to command mode

:w! count2.sh
: Write the file to disk

:q
: Close vi

Simple when you know how.

This is the last article for Exam 101 - Topic 103: GNU and UNIX commands. See our series roadmap for a description of and link to other articles in this series.

Resources

Learn

- Use the developerWorks roadmap for LPIC-1 to find the developerWorks articles to help you study for LPIC-1 certification based on the April 2009 objectives.
- At the LPIC Program site, find detailed objectives, task lists, and sample questions for the three levels of the Linux Professional Institute's Linux system administration certification. In particular, see their April 2009 objectives for LPI exam 101 and LPI exam 102. Always refer to the LPIC Program site for the latest objectives.
- Review the entire LPI exam prep series on developerWorks to learn Linux fundamentals and prepare for system administrator certification based on earlier LPI exam objectives prior to April 2009.
- In our tutorial on using vi, "vi intro -- the cheat sheet method" (developerWorks, December 2000), Daniel Robbins shows you how to use the vi editor to edit text, use insert mode, copy and paste text, and use important vim extensions like visual mode and multi-window editing.
- Learn more about scripting the vim editor by starting with "Scripting the Vim editor, Part 1: Variables, values, and expressions" (developerWorks, May 2009).
- In "Basic tasks for new Linux developers" (developerWorks, March 2005), learn how to open a terminal window or shell prompt and much more.
- The Linux Documentation Project has a variety of useful documents, especially its HOWTOs.
- In the developerWorks Linux zone, find more resources for Linux developers, and scan our most popular articles and tutorials.
- See all Linux tutorials and Linux tips on developerWorks.
- Stay current with developerWorks technical events and Webcasts.
- Follow developerWorks on Twitter.

Get products and technologies

- With IBM trial software, available for download directly from developerWorks, build your next development project on Linux.

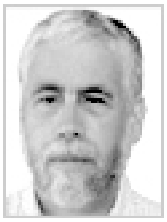
Discuss

- Participate in the discussion forum for this content.
- Get involved in the My developerWorks community. Connect with other

developerWorks users while exploring the developer-driven blogs, forums, groups, and wikis.

About the author

Ian Shields



Ian Shields works on a multitude of Linux projects for the developerWorks Linux zone. He is a Senior Programmer at IBM at the Research Triangle Park, NC. He joined IBM in Canberra, Australia, as a Systems Engineer in 1973, and has since worked on communications systems and pervasive computing in Montreal, Canada, and RTP, NC. He has several patents and has published several papers. His undergraduate degree is in pure mathematics and philosophy from the Australian National University. He has an M.S. and Ph.D. in computer science from North Carolina State University.