2.3.2 Use vi

Click one of the buttons to take you to that part of the video.

Use vi 0:00-0:58

In this demonstration we're going to practice using the vi editor. In order to start the editor, you simply run 'vi' at the shell prompt. At this point, you have three different choices you could make. You could just run vi by itself, which would open up a blank document with no filename associated with it. If you want to create a new file as you run vi, you could then enter the name of the file here, such as 'mynewfile'. Or if you wanted to edit an existing file, you would replace with the name of that file--and you must include path information if that file doesn't reside in the current directory.

For our purposes today, let's go ahead and create a new file, 'mynewfile', within the vi editor. Just as with any other text editor, the changes that we're going to make to this file won't actually be committed to disc until I save the file, which we'll do in just a minute.

Command Mode 0:59-1:21

I'm currently in my text editor, but notice what happens if I try to type text. I'm typing and nothing is happening. And this is a significant point of confusion for new Linux users, because you open up a text editor, you try to type in it, nothing happens. Understand that with the vi editor, you start in command mode by default, and you can't edit the text while you're in command mode.

Insert Mode 1:22-2:17

To do this, we need to switch from command mode into insert mode. In order to switch to insert mode, you can press one of three different keys. One option would be to press the 'i' key, or the 's' key, or you can press the 'Insert' key on your keyboard. Notice that when I do, a little prompt down here appears, telling us that we are now in insert mode.

Once we're in insert mode, we can then actually edit the text in the file. "This is my new file, it is very nice." Because we're in insert mode, anytime we move the cursor around and then put new text in a line, it'll force whatever text that comes after the cursor to be pushed down. For example, I can say, "It is very, very nice." When I insert the second word "very", it pushes "nice" down the appropriate amount of space.

The vi editor actually has two different editing modes. The first one is insert, which we're already in, but there's another one called replace mode.

Replace Mode 2:18-3:13

If I press the 'Insert' key again, notice that the prompt down here changes from insert to replace. In replace mode, whatever characters we type replace whatever text is currently under the cursor. For example, if I wanted to put "very" again like I did before, I type 'very' and it replaces the word "nice". To be honest, I don't use replace mode very often. I've never found a real great use for it. I almost exclusively use insert mode. If I want to get back to insert mode from replace mode, I simply press the 'Insert' key again, like I just did, and then you can type as you normally would in most word processing and text editing programs.

As I said just a minute ago, all the changes we're making to this file are not actually being committed to disc; they're only saved in memory. If we want to preserve these changes, we need to write this information out to the file. You can't actually do that in vi as long as you're in insert mode.

Command Line Mode 3:14-3:43

Before you can do that, you have to enter command line mode. Notice I said "command line mode" not "command mode". Command line mode is another vi mode that you need to be familiar with. You can't go from insert mode directly into command line mode. Instead, we first press 'Escape' once to go into command mode, and then we enter a full colon (:) to enter command line mode. And as you can see down here, a colon (:) is displayed, and a line is provided so we can enter commands. Hence the name command line mode.

Save a File 3:44-5:02

If we want to save this file, we have a couple of different options. One option would be to simply type 'w' and press 'Enter', and when I do, the file is saved. And it tells us down here that mynewfile has been created. It's a new file, it's got one line, 38 characters, and it has been committed to disc. It's been written in the file system.

There are other options. If we go back into command line mode by entering a full colon (:) again, another option would be to enter 'w', 'Space', and then a different filename. This is the equivalent of doing a "save as" in a traditional text editor. If I were to enter 'w mynewfile2', it would save the file using this new filename.

Another option is to save the file and exit the editor in one operation. I do this all the time. I go in, I edit a configuration file and make the changes, and then I am done so I want to save the changes and get out of the editor and continue doing other tasks. There are two ways to do this. One is to enter 'wq', which stands for write quit; or you can just type 'exit', and they both do the exact same thing. They save the file and they exit out of the editor. If you want to just quit without performing a write operation, you enter just 'q'. And you exit out of the vi editor but the changes are not saved.

Let's get back in.

Command Mode Edit Commands 5:03-5:27

Notice now that instead of creating a new file with the vi command, I am editing an existing file. And because the file resides in the current directory, I don't have to specify any path information. I just enter 'vi mynewfile' and it loads the file up again. And just as before, we enter by default into command mode. Within command mode, there are several different commands that we can use to manipulate the text within this file.

The dw and p Commands 5:28-6:21

For example, a very common text editing task is to cut and paste text. You're probably very familiar with how to do that in a word processor or graphical text editor. You can do the same thing in vi; it's just done a little differently. For example, while in command mode, we can cut and paste text using the 'dw' command. The 'dw' command cuts the word that comes immediately after the cursor, along with the space that follows that word and puts it in a memory buffer. It's basically like doing an Edit+Cut in a graphical editor.

Let's go down here to the word "very," and I press 'dw' and it takes the word "very" and the space that followed very and puts it in a memory buffer. Once done, I can then move the cursor around to wherever it is I want the word to go. Then in order to paste the word that's in the buffer, I press 'p', and the word "very" is pasted from the memory buffer into the line of text.

The de Command 6:22-6:48

In addition to 'dw', you can also use the 'de' command. It basically does the same thing, except that it just cuts the word that comes after the cursor and not the space that follows the word. Just like before, it takes the word, it puts it in the memory buffer--but unlike before, it does not cut the space that came after the word. And at this point we could paste it again, just like we did before, using the 'p' command. It puts the word back in the line where the cursor is.

The u Command 6:49-7:11

With a graphical text editor, you have the option to undo the last action. You have that same option here in command mode in vi. You just simply press the 'u' key and then whatever change you just made is undone. In this case, we pasted "very," and now the paste operation is undone and we go back to the state we were in before we performed that operation. I'm going to put the word back in by pasting it.

The dd Command 7:12-8:01

In addition to 'dw' and 'de', we can also use the 'dd' command. The 'dd' command works in much the same way as 'de' and 'dw', except that it cuts the entire line of text. For example here, I'll put my cursor at the beginning of the line and do a 'dd', and the entire line is cut, and then I can paste it just as before using the 'p' key. There's a variation on 'dd' called 'd$'. It works in the same way as 'dd', except that it cuts from the current cursor position to the end of the line. For example, if I wanted to cut this last sentence and preserve the first half of the sentence over here, I would enter 'd$' and it cuts just from the cursor position to the end of the line. And as before, we can paste it using the 'p' key.

Search for Text 8:02-9:31

Another useful feature of the vi editor is the fact that it allows you to search for text within the file. To do this, in command mode you enter forward slash (/), followed by the text that you want to search for. Let's search for the term 'nice'. When I do, it automatically jumps to the first instance of the word "nice" in the file. This is only a one line file, so I only have one instance of the word nice, but if I had multiple instances of the word nice, I could use the 'n' key at this point to jump to the next version.

It's important to note that when you use forward slash (/), it searches from the cursor position forward in the file. There may be situations when you need to search from the cursor position backwards in the file. Notice here that I have the cursor on the period at the end of the line. We can perform the same search we did before for the word nice, but this time we want to search backwards through the file. To do this, we use a question mark (?) instead of a forward slash (/). We use '? nice', press 'Enter', and again it finds the next previous instance of the word specified.

Just as you can cut and paste text within the vi editor, you can also copy and paste text, just like you can in a graphical editor. For example, you can use the 'yy' command to copy an entire line of text to the memory buffer. Notice that the text didn't change, because we copied--we didn't cut--and then we can go up here and use the 'p' key to paste once again.

Before we end this demonstration, there's one more thing I want to show you.

Quit without Saving 9:32-10:33

There may be instances when you've opened a configuration file in vi and you've made some changes and then you realize that you've made a lot of mistakes and you don't want to try to go back and fix all of the mistakes you made; you want to get out of the editor without saving your changes to the file so you can come back in the second time and do it the right way. The default way to exit out of vi is to go into command line mode and then press 'q', but we have made many changes to this file since we opened it. Watch what happens if I try to use the 'q' option. It says, "You can't do that because you made lots of changes since the last time this was written to a file." Notice here that it gives us a prompt as to how to do this. It says, "Add ! to override." Once again let's go into command line mode, and this time we enter 'q' with an '!'. Press 'Enter' and it does exit out of the file. And if we do a 'cat' of 'mynewfile', we should see that none of the changes we made were actually saved.

Summary 10:33-10:46

That's it for this demonstration. In this demo we talked about how to use the vi editor to edit files. We talked about how to use insert mode, we talked about how to use replace mode, we talked about how to use command line mode, and then we reviewed several of the commands that you can use in command mode.
