2.3.3 Use nano

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

Use nano 0:00-0:17

In this demonstration, we're going to use the nano text editor. Before we start, it might be helpful to talk about the general differences between between the vi editor and why you might want to use one over the other.

The vi Editor versus the nano Editor 0:18-1:20

First of all, the vi editor is the defacto text editor for working in the Linux shell. The vi editor is included in virtually every Linux distribution. But the nano editor is also included in most Linux distributions. If nano is not included, it can readily be installed. In fact, your likely to find that when you type nano at a shell prompt and nano is not installed, you will be prompted to install it. Both editors are licensed as essentially, free software, so there's not much difference in the availability in the two editors

The major difference between them is that the nano interface is more intuitive, especially for new users who are used to a GUI-based text editor. The learning curve for nano is not as steep as vi, mostly because nano displays a list of shortcuts at the bottom of the interface.

Bottom line, if you're just starting out in Linux and need a simple editor, choose nano. If you're already proficient using the vi editor or you need its more powerful features, then choose vi.

The nano Interface 1:21-2:26

Let's start by looking at the nano interface. We've opened an ssh session and logged into a Linux server. We've also increased the size of the terminal from the normal 80 columns by 24 lines to about 120 columns by 40 lines. This server wasn't installed with a graphical interface, which is quite common for Linux servers. As administrators, we may need to change configuration files. For that, we'll need a text editor like nano.

At the shell prompt, we'll type ‘nano'. At the top of the nano interface is the name of the program and the version number. Notice that New Buffer is shown. Just like vi and other GUI editors, you don't directly edit a file. Instead, your edits are made in memory and then saved to a file. We haven't saved to a file yet. Otherwise the file name would be displayed instead of New Buffer.

At the bottom, we see two other parts of the interface. The third line from the bottom is a system message that displays relevant information. The two bottom lines are shortcuts which will dynamically change depending on what actions you take.

Basic Help 2:27-3:12

The system message tells us, "For basic help, type Ctrl+G." When we do that, we see the main nano help text. We can use this information to decipher the shortcuts. We read that the caret symbol (^) shown in the shortcuts means press the Ctrl key. For example, farther on down, we see that caret G (^G) means, "While holding down the Ctrl key, press the (G) key." The help text tells us that this will display this help text, which is exactly what we have done.

Notice the next two shortcuts. Ctrl+X closes the current buffer or exits from nano. Ctrl+O writes the current buffer to disk. Also, notice that the keys within parentheses are alternate keystrokes that accomplish the same task.

Meta and Alt Keys 3:13-3:51

As we look down the list of shortcuts, we see shortcuts that begin with M-. Reviewing the help text, we learn that these are Meta-key sequences. Older keyboards had a Meta key. On newer keyboards, you use the Alt, Command, or Escape keys instead. Let's use the Alt key for the Meta key. For example, Alt+U undoes the last operation.

Looking at the shortcuts, we'll use Ctrl+V to view the next page and continue until we reach the end of the help text. Here, you can see that Alt+L configures nano to automatically hard wrap lines that are too long. Let's see how that works.

Automatic Text Wrapping 3:52-4:25

We'll close the help text with Ctrl+X. As we do so, notice that the shortcuts change. For example, Ctrl+X will now exit nano instead of closing the help text buffer. We don't see Alt+L because it's not important enough to be listed, but when we use it, the system message shows us that overlong lines will be wrapped. Let's type a few lines of text to see how that works. When we turn off wrapping and type in the same text, a dollar sign shows us there is more text that is off the screen to the left or right.

Cursor Movement 4:26-4:59

While we're here, let's try to discover how to move the cursor around in the text. What happens when we use the arrow keys? The curser moves up, down, left, and right, as expected. What do the Home and End keys do? Again, as expected, Home moves the cursor to the beginning of the line, and End, to the end of the line.

Here's a fun one. Ctl+Space Bar moves the cursor forward one word, and Alt+Space Bar moves the cursor back one word. You can look in the help text to find other key sequences that move the cursor to a particular line or column.

Cut, Copy and Paste 5:00-5:37

How about cutting, copying, and pasting text? The shortcuts show us to mark text, use Alt+A. We'll press Alt+A and then move the cursor. That will highlight some text. To copy the text, use Alt+6.

When we look at our shortcuts, we don't see Paste Text, but we do see Uncut Text, which is Ctrl+U. Let's put our cursor on a new line and use Ctrl+U. Yes, that means paste. Let's try Alt+A, Ctrl+K ,and Ctrl+U. Yes, that does a cut and paste.

Save to a File 5:38-6:14

We'll leave you to explore the help text for other shortcuts and experiment with them to see how nano operates. But we'll want to see how to save our buffer to a file and how to edit other files. Let's try Ctrl+O. The shortcut says this performs a write out. Since we haven't specified a file, this prompts us for a file name. Let's type ‘temp.txt'. The system message tells us the number of lines that were written, and the file name is listed at the top.

When we change the buffer and use Ctrl+X, we're prompted to save the buffer. We'll answer Yes.

Save to a New File 6:15-6:38

At our shell prompt, we can see that the file has been created, and here is its contents. To edit a file, type nano and the file name. Within nano, we can save to a new file using Ctrl+O and then typing a new file name. We'll need to confirm this is a different name.

Edit a File Outside the Current Directory 6:39-7:29

Let's exit nano and see what happens when we specify a file that is not in our current working directory. How about a network script? The nano editor will put the full path to the file at the top. It also tells us that we don't have permission to write our changes to the file.

To do that, let's use the root account. Now we can edit and save our changes. But let's just exit without saving.

Summary 7:30-7:55

That's all for this demonstration. We talked about how the nano editor is more intuitive, but less powerful, than the vi editor. We saw how the nano interface displays system messages and shortcut hints, and we saw how the help text lists all the shortcuts. We determined how to move the cursor, and how to cut, copy, and paste text. Finally, we saved our changes to a file and saw how nano worked with files that are not in the current working directory, and files that are read only.
