2.13.2 The awk and sed Commands

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

awk and sed 0:00-1:24

When you're processing text screens within a script or perhaps when you're piping the output of one command at the shell prompt to the input of another command, there will probably be times when you want to actually filter the output of one command so that only certain portions of the text stream are actually passed along to the standard input of the next command. There's actually a variety of tools you can use on Linux to accomplish this.

In this lesson we are going to look at using the awk and sed commands. Now before we begin, please understand that the sed and awk commands are actually quite extensive, and they can do a lot of different things. We don't have time here to cover everything. Instead we are going to look at some basics. What you need to do is review the man page for both of these commands and then use that information to explore what all possibilities are available for these two commands.

Let's begin by looking at sed. The sed is a text stream editor. Unlike other interactive text editors that you are already familiar with like vi and gedit, a stream editor doesn't provide any kind of user interface. Instead what it does is take a stream of text as a standard input and then will perform whatever operations you tell it to edit that text stream in the manner that you specify and then it sends the results to its standard out.

---

sed 1:21-6:29

There are several different commands you can use with sed to edit text. Two of them are listed here. The first one is the 's' command. 's' is used to replace certain instances of a specified text stream with new text. The syntax is shown here. We run 'sed s/' followed by the original text that we want to search for, forward slash, then we want to specify what text we want to replace that text with. Then another forward slash.

An example of doing this is shown right here. Notice that we've used the cat command to display a text file on the screen. The name of the file is lipsum.txt, and we're going to pipe the output of the cat command to the input of the sed command, and then using the 's' command we're going to tell sed to parse the text stream that's coming in and look for any instance of the word ipsum, lowercase. For every instance of that word that we find we want to uppercase it. This is the replacement text. The result are written to the screen because we didn't tell sed to redirect or pipe its output. Notice here that one instance of ipsum was found and the text, and it was lowercased, so using the sed tool, we transformed it from lowercase to uppercase.

In addition to the 's' command, you can also use 'd' command which is also used to delete specified text. The syntax for deleting text with sed is shown here. We run 'sed /' followed by the text that we want to search for then we tell what to do. If it finds that text which is D for delete. Your first inclination might be to think that it will delete just the text that you specified here, but that is not the case. What it actually does instead is delete the entire line that contains the term that we specify right here. An example of that is shown right here. We run the cat command again, we pipe the output of the cat command to the input of the sed command. We tell it to look for all lines that contain the word elit in it. If it contains the word elit, we want to delete that line. Notice in the example above, that the first line of text does contain the word elit and as a result that entire line is removed from the output of the sed command.

You will notice in both of these examples that because we didn't tell sed what to do with its output it just wrote it to the default standard out which is the screen. If we wanted to capture that text and save it, what we would have to do is either pipe it to another command or we could use the greater than sign as shown right here to redirect the output of sed to a file. In this case we redirect it to lipsum_out.txt. Then the transformation we made to the text with this command would be saved as a file in the file system.

In addition to sed, you can also use the awk command to edit a text stream. Frankly, awk is quite a bit more powerful than sed in what it can do. Now like sed, awk can be used to receive input from another command through a pipe to a standard in, and then manipulate it, edit it in a manner that you specify but the way awk does this is quite a bit different than the way sed does. What you need to understand is that the awk command treats each line of text that it receives from a text stream as a record. Much like a record in the database. Each word in each line, words that are separated by either a space or a tab character will be treated as a separate field within the record. For example, take a look at this text stream right here. As far as awk is concerned, this file has 7 records, because it has one, two, three, four, five, six, seven separate lines of text. Each line of text has a carriage return line feed character at the end that creates a new line. This is the character combination that awk uses to define the end of a record.

Within each record are separate fields. Remember they are identified by space or tabs. This first line of text has one, two, three, four, five, six, seven, eight different fields. The second record actually has more fields. It has one, two, three, four, five, six, seven, eight, nine, ten fields. The key thing to remember is that with awk, the white space, tabs are spaces, not the punctuation delimits fields. It doesn't care about periods or commas or anything like that. Each field within each record is referenced a variable number. The first field is one, the second field is two, second field is three and so on because they're variables; we use a dollar sign ($). The first field in the first record would be referenced as dollar sign 1 ($1); the second one is dollar sign 2 ($2); the third one is dollar sign 3 ($3), and so on.

awk 6:28-8:26

Using awk, we can then specify a particular field in a specific record then manipulate it in some manner. We can edit it. The syntax for doing that is shown here. We run 'awk', and then we specify a pattern followed by the manipulation we want to perform on that pattern. Just like sed we have to take the output of some other command and pipe it to the input of the awk command to provide it with the text stream that it needs to perform its manipulation. In this case, we're running the cat command. We're piping the output of cat to the input of awk and what we're telling it to do is from within that text stream, print the first three fields of every single record on the screen. Because we did not specify a pattern to match on in this example, it matches everything. Every single record in the text stream. As a result, it prints out the first three words of every single line of the text file.

If you wanted to, you could also include a pattern to specify exactly which records to match on. For example, let's suppose that we only wanted to display the first three fields of any record in this text stream that contains the term lorem, somewhere in a given line. If a line of text contains the word lorem then we are going to perform the manipulation. If it doesn't then we are going to skip it and not process it. We could use the command shown here to do this. In this case, we use the cat command to output the text stream of the input of the awk command. We provide a pattern to match on of lorem. If the line of text contains the word lorem, then we are going to print fields 1, 2, and 3. In this example, only one line of text matched the pattern that we specified so only those three words are specified in the output of the awk command.

Summary 8:27-8:32

That's it for this lesson. In this lesson we reviewed how you can use the sed and the awk commands as text stream editors.