2.13.1 Text Stream Processing

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

Text Stream Processing 0:00-0:43

When you're processing text streams within a script, or maybe when you're piping output at the shell command, there may times when you actually need to 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. In this lesson, we're going to look at a variety of different tools you can use to do this. We're going to look at the cut command. We're going to look at the expand command as well as the un-expand command. We're going to look at the 'fmt' command. We're going to look at join and paste. We're going to look at the nl command, the od command, the pr command, the sort command, the split command, the tr command, the uniq command, and the wc command.

The cut Command 0:44-2:30

Let's begin by looking at the cut command. The cut command is used to print columns or fields that you specify from a file to its standard output. By default, the tab character is the delimiter that's used to determine which field or column from the text stream is going to be included. There are several different options you can use with the cut command. You can use the 'blist' option to select only certain bytes. You can use the 'clist' command to select only certain characters. You can use the '-d' command followed by a delimiter to specify a different character instead of tab to use for the field delimiter. And you can use the 'flist' command to select only specified fields. And you can use the '-s' option to print lines that don't contain any delimiters.

Now, here's an example of how you could do this. It's a fairly simple example, but it brings the point across. Let's supposed we want to use the cut command to display just usernames that are contained in the '/etc/passwd' file. Remember, the name of each user is contained in the first field of each line in the file. However, the password file does not use tab characters between fields. Instead, it uses a colon. So what we have to do is use the '-d' option here to specify that we want to use a colon instead of a tab as the delimiter to identify the individual fields within each record. So in this case, we run 'cut -d:' and then we use the '-f' option that we talked about down here. And we specify which field it is we want to use, which is field number one, and then the file that we want to process. And the output is shown here. Only the usernames, the first field of each record within the '/etc/passwd' file, is actually passed through to the standard out of the command. Everything else that was in that file is ignored. We've filtered it out. Next, let's look at the expand and unexpand commands.

The expand Command 2:31-3:00

The expand command is used to process a text stream and remove all instances of the tab character and replace those instances with a specified number of spaces. The default is eight. You can change that if you want to using the dash 't' (-t) option. And the syntax is to type 'expand -t' followed by the number of spaces that you want to replace tab characters with, followed by the file name that you want to process. You could also just accept input to the standard in of the command from a different command and do the same thing.

The unexpand Command 3:01-3:24

Now, unexpand works in the opposite manner as the expand command. It converts spaces in the text stream into tab characters. And by default, eight contiguous spaces will be converted into tabs. And again, you can customize this using the dash 't' option, and that allows you to specify a different number of spaces that will be required to convert into a tab character. Now let's take a look at reformatting text with the 'fmt' command.

The fmt Command 3:25-4:14

Now, it's commonly used to change the wrapping of long lines within a file to make it more manageable. The syntax for using 'fmt' is to run 'fmt', followed by the option that you want to use, and then the file name that you want to process. You could, likewise, just send text to the standard in of the command and do the same thing. In which case, you don't actually have to specify a file name. An example of using this command is shown here. In this example, we run the 'fmt' command, and we use the dash 'w' (-w) option to specify the width of each line of text within the file. And we want to set that to a maximum of 80. If it's shorter than 80 and it breaks, we don't care. But if it's longer than 80 characters, we're going to insert a carriage return and a line feed character between character 80 and 81, effectively creating a new line. And the output is shown down here. Let's look at the join and the paste commands next.

The join Command 4:15-5:38

Now the join command prints a line from each of two specified input files that have identical joined fields. Now by default, the first field in both files is defined as the joined field. And it's delimited just by white space, either space characters or a tab. Although, you could specify a different joined field using the dash 'j' (-j) option if the joined field was the second field in each record, or third, or whatever it is that you want to use. So the syntax is to enter 'join' file one, file two. In this example, I have two different files. The first one is named firstnames. The second one is named lastnames. And notice that there are two fields in each line. Field one and field two. Well, in both of these files, we have records one, two, and three; and records one, two and three. Because this first field in the first file and this field in the second file are both the same, we can join both files together based on these values. So you see I run the 'join -j 1' to specify we're working with the first field as the joined field. Then the two file names: firstnames and lastnames. And as a result, we join Mike Johnston into one record, and Jenny Doe into the second record, and Joe Jones into the third record. The match was made based on the values here in this first field. For example, Joe Jones was created because both records have a value of three in the first field. So we grab Joe and we grab Jones and we put them together.

The paste Command 5:39-6:29

Now the paste command works in much the same manner as the join command. It pastes together corresponding lines from one or more files into columns. By default, the tab character is used to seperate the columns. Now you can use the dash 'd' (-d) option followed by a character to specify a different delimiter character if you don't want to use tab. You can also use the dash 's' (-s) option to put the contents of each file into a single line. So the syntax is 'paste', followed by file one and file two. We're gonna work on the same files that we worked with last time. The firstnames file and the lastnames file. So essentially what this command does is say, "Ok. First line. First line. Let's glue them together" to make this new first line, which contains the contents of both." And we didn't specify a different delimiter character, so a tab was added in between Mike and Johnston.

The nl Command 6:30-7:10

Now let's look at the nl command. Now the nl command is a pretty simple command. All it does is determine the number of lines in the file. And when you run the command, the output consists of each line in the file that you specified, but with a number added to the beginning. Just like all the other commands we've been working with, you can use nl in one of two ways. One option is to run 'nl' and specify a file name that you want to add line numbers to. Or, you could run some other command and pipe the standard out from that command to the standard in of the nl command, and it'll do the same thing. In this example, we just did it with a file in the file system. We said 'nl boot.log'. nl command read the boot dot log file, and added a line number at the beginning of each line.

The od Command 7:11-8:58

Next, let's look at the od command. Now the od command is actually pretty useful. You know that if you try to view the contents of a text file at the shell prompt, let's say the cat command, no problem. The text displays and you can read it. But what if you need to view the contents of a file that is not a text file? For example, maybe you created a file using a word processing program, like LibreOffice or OpenOffice. Well, if you tried to view the contents of those files with the cat command, you're just gonna see a bunch of gibberish because there's a lot of binary code included within the files itself. The od command is really useful because it can take any kind of file, including a file that contains binary data, and dump it into many different formats so that you can read it. You can dump it into octal format, decimal format, floating point, hexadecimal, and just plain old character string formats. And the great thing is that the output from the od command is just plain text, so you can use any text manipulation utility, like cat, less, more, whatever it is you want to use. Or, you can also further filter it through one of the other commands we've talked about in this lesson.

Now, just as with all the other commands, we can either run 'od' and open a particular file with it and process it. Or, we can send it to the input of the od command from some other command. There are several different options we can use. We can use dash 'b' (-b) to create an octal dump, 'd' to create a decimal dump, 'x' to create a hexadecimal dump, and 'c' to create a character dump, which is what I usually use. In this example right here, I have an OpenOffice file called 'zombie.c.odt'. If tried to view this file natively with the cat command, I can't--it's just a bunch of junk. But by using the 'od -c zombie.c.odt' command, all the various characters within that file are translated into simple text. And I am actually able to view it in the output of the command. I could also redirected the output to a file, and I could view it with the cat command.

The pr Command 8:59-10:02

Next, let's look at the pr command. The pr command is used to format a text stream for printing. Basically, it formats the data with pagination, with headers, and arranges the data into columns. Now the header contains the date and time, the name of the file, and also adds a page number. The syntax is to type 'pr', followed by options, and then the file name that you want to reformat. And likewise, as with all the other commands, we could just accept a text stream from a different command, and we won't have to use a file name in that case. We'll just process that text stream itself. There are several different options you can use with pr command. You can use dash 'd' (-d) to double space the output. You can use the dash 'l' (-l) command to set the page length to a specified number of lines-- I believe the default is 66 lines per page. And you can use the dash 'o' (-o) option to add a margin. Basically, it'll offset each line with a particular number of spaces. Which is a good idea, because I think that by default it doesn't do any. The default margin is zero, which puts the text right up against the edge, and you probably don't want to do that. So, you can add a certain number of spaces to push the text in, away from the edge of the paper.

The sort Command 10:03-11:12

Now we're going to look at the sort command. The sort command sorts the lines of text within a file or within a text stream alphabetically. And the output is sent to the standard output, just like all of our other commands. The syntax is to enter 'sort', followed by the options that you want to use, and then the file name you want to process. And, again, you could just accept the input from another command-- the output from another command-- as the standard in, and then run the sort command on the text stream itself instead of on the file name. That's just fine. There are several different options you can use with the sort command. The dash 'f' (-f) option folds lowercase characters to uppercase characters. Dash 'M' (-M)--capital 'M'--sorts by month. Dash 'n' (-n)--lowercase--sorts numerically. And dash 'r' (-r) specifies that it sort in reverse order. An example of that is shown over here. We have our firstnames file. We're going to run the sort command against it. We're going to sort numerically because, remember, the first field of this file-- the first field of each line of this file-- starts with a number. One Mike, Two Jenny, Three Joe. So, we specify 'n' to sort numerically. The default is alphabetically, which won't work in this file. And we also specify '-r' to sort the records in reverse order. So we'll start at the highest numbered line--three-- and go down to the lowest numbered line-- one.

The split Command 11:13-11:37

Now let's look at the split command. The split command takes one large input file, or a really large text stream, and breaks it up into a series of separate files. It basically just takes one big file and chops it up into a bunch of smaller ones. Now, by default, it's going to split the input file into separate 1,000 line files. Although you can use the dash 'n' (-n) option to specify a different number of lines to use.

The tr Command 11:38-13:11

Now let's look at the tr command. tr stands for translate. It's used to translate or delete characters from the text stream. And, unlike the other commands that we've been looking at, you can't actually use tr on a file. The only option for using tr is to generate a text stream with some other command and send the standard out of that command to the standard in of the tr command. The syntax for using tr is to run 'tr', followed by the option that you want to use. And then you specify what you want to translate to what. So, X specifies what is to be translated, and Y specifies what it is going to be translated to.

There are different options you can use with this command. Dash 'c' (-c) specifies that we use all characters that are not in the X part of the command. Dash 'd' (-d) specifies that we delete characters that are in the X part of the command. Dash 's' (-s) specifies that we replace each input sequence of a repeated character that's in the X part of the command with a single occurrence of that character in the Y part of the command. And dash 't' (-t) specifies that we truncate the X part of the command to the length of the Y part of the command. A simple example of using tr is shown here. We run 'cat' to create a text stream from the lastnames file, and we send that to the standard in of the tr command. And we specify for X: All characters 'a-z' lowercased. And we want the output to be the same characters translated to uppercase, 'A-Z'. So remember, lastnames used capital letters for just the first letter of the name. By running it through tr, we now have all uppercase characters.

The uniq Command 13:12-13:33

Let's look at the uniq command. The uniq command reports or omits repeated lines. The syntax shown here is to run 'uniq', followed by several options, and then you specify your input file; and then you specify your output file. You can use a couple of options with the uniq command. Dash 'd' (-d) will only print duplicate lines. And dash 'u' (-u) will do the opposite and only print uniq lines.

The wc Command 13:34-14:26

Finally, let's look at the wc command. Now, the wc command prints the number of new lines, words, and/or bytes in a file. The syntax is to enter 'wc', followed by the options that you want to use, and then the name of the file to process. You can use the option shown here. Dash 'c' (-c) prints the byte count of the file. Dash 'm' (-m) prints the character count, the number of characters in the file. Dash lowercase 'l' (-l) prints the number of lines in the file. Dash capital 'L' (-L) prints the length of the longest line in the file. And dash 'w' (-w) prints the number of words in the file. So, in this example, I run the 'wc' command against the firstnames file. So the first number in the output is the number of lines in the file. The second number is the number of words in the file. And then the last parameter is the total size of the file in bytes: 21 bytes.

Summary 14:27-14:41

That's it for this lesson. In this lesson, we looked at the cut command. We looked at expand and unexpand. We looked at the 'fmt' command. We looked at join and paste. We looked at the nl command. We looked at od, pr, sort, split, tr, uniq, and then we ended by looking at the wc command.
