2.13.4 Text Stream Processing Facts

When you are processing a text stream within a script or piping output at the shell prompt, you many need to alter the output of one command, allowing only certain portions of the text stream to pass along to the stdin of the next command.

This lesson covers the following topic:

- Text stream processing commands

Text Stream Processing Commands

A text stream is any information redirected from the standard output of a command to the standard input of another command. The following commands can be used to intercept and process the text stream.

Command	Function	Example
cut	Prints just the columns or fields that you specify to the standard output. By default, the tab character is used as a delimiter to define each field. Options include the following: - **-c** cuts characters. - **-f** cuts fields. - **-d** specifies the character used as the field delimiter. The default is a tab. - **-s** removes lines that do not have a field delimiter. - **-d' '** specifies a space as the field delimiter.	For the following example, a file named myfile has the following text: http://www.site1.com http://www.mysite.com http://www.anothersite.com **cut -c1-7 myfile** takes the first seven characters of each line and sends it to standard output. In this example, the first seven characters are *http://*. **cut -c8- myfile** takes character eight to the end of each line and sends it to standard output. This removes http:// from each line.
expand	Replaces a tab character with a specified number of spaces. - The default is eight spaces. - **-t** specifies the number of spaces to be used.	**expand -t 1 myfile** replaces each tab character in the file with a single space.
fmt	Formats lines in a file or text stream to a uniform length. This is useful to format long lines of text to fit in a terminal window. Options include: - **-w** specifies the number of characters for the width. The default is 75. - **-s** prevents the command from formatting lines shorter than the specified length. This command is often used with code text to keep lines of code separate.	**fmt -w 80 myfile** sends the contents of myfile to standard output with all lines having a uniform length of 80 characters.
join	Combines text from two files based on identical fields and sends the result to standard output. By default, fields are offset by whitespace. Options include the following: - **-i** ignores case when searching for identical text. - **-j** specifies the number of the field to use when joining. This specifies both files. - **-1** specifies the number of the field from the first listed file to use when joining. - **-2** specifies the number of the field from the second listed file to use when joining. - **-t** specifies the character to use as the field delimiter.	File1 has the following text: 1 Mark Twain 2 William Shakespeare 3 John Steinbeck File2 has the following text: 1 Tom Sawyer 2 Othello 3 Of Mice and Men **join file1 file2** sends the following text to standard output: 1 Mark Twain Tom Sawyer 2 William Shakespeare Othello 3 John Steinbeck Of Mice and Men **join -j 3 -t : fileA fileB** joins the files using the third field as the common field, and a colon as the field delimiter.

nl	Places a line number in front of each line in a text file and send the result to standard output. Options include the following: ▪ **-i** specifies the increment to use when numbering the lines. ▪ **-v** specifies the starting number. ▪ **-s** specifies the text to be placed between the number and the line. The default is two spaces.	**nl -s ":" myfile** adds the number, a colon, and a space to the front of each line in the file.
od	Displays the contents of any file in octal, decimal, hexadecimal, or character format. Options include the following: ▪ **-b** specifies an octal dump. ▪ **-d** specifies a decimal dump. ▪ **-x** specifies a hexadecimal dump. ▪ **-c** specifies a character dump.	**od -c /bin/tar** shows the contents of the tar command executable in character format.
paste	Adds the contents of one file to the contents of another file on a line-by-line basis. ▪ By default, the tab character is used to separate columns. ▪ **-d** specifies a character to place between the conjoined lines of each file. Only a single character can be specified.	**paste -d @ file1 file2** conjoins each line of file2 to the end of each line of file1 and places an @ between each line pair.
pr	Formats a text file for printing. By default, this command: ▪ Separates files into 66-line pages. ▪ Uses the first five lines to create a header that contains a page number, the time and date, and the path to the file. ▪ Uses the last five lines to create a footer of blank lines. Options include the following: ▪ **-d** double-spaces the lines. ▪ **-h** specifies text to replace the file name in the header. ▪ **-l** specifies the number of lines. The default is 66. ▪ **-t** prevents the command from creating the header and footer. ▪ **-o** creates a margin on the left side of the text.	**pr myfile** sends the text to standard output using default settings. **pr -d -l 60 -t -o 5 myfile** sends the text to standard output using double spacing, a page length of 60 lines, no headers or footers, and a five-space margin on the left side.
sed	Takes text or commands from the command line as input and modifies the text document named in the command line. **sed** is particularly useful under the following circumstances: ▪ When a file is too large to open and edit conveniently in a text editor. ▪ When the series of edits (for example, adding line spacing, margins, replacing text) is too complex to perform easily in a text editor. ▪ When it is easier to perform a series of global document changes. Flags and options include the following: ▪ **s** replaces the text behind the first / with the text behind the second /. To save the	**sed 's/Nancy/Nanci/' *originalfilename* >*newfilename*** replaces every occurrence of "Nancy" with "Nanci." **sed -n '/there were no credible/,/transfer assets abroad/p' *filename*** displays only the text of a paragraph beginning with "there were no credible" and ending with "transfer assets abroad." **sed -n 56,89p *filename*** displays lines 56 through 89 of the specified file. **sed -e 's/J.K.W/James K. Whitworth, Esq./' -e 's/Hillary Stuart/Ms. Mary Edwards' -e s/Johnson, Gabriel, and Hawkins/McPhee, Larkin, Simmons' *originalfilename* >*newfilename*** allows three substitution commands to occur at the same time. **sed -f *scriptfilename originalfilename* >*newfilename*** treats the scriptfilename file as a script file, running each command against the text in the original file and saving the results to the new file. **echo night day night | sed s/night/day/g** changes both instances of the term night to day. Without the trailing **g** flag, only the first instance changes.

	results of the command, use > to redirect the output to a new file. - **d** deletes lines that contain the specified term. - **g** changes all occurrences of the term in a line. - **p** prints the modified lines in addition to the standard output. - **-n** suppresses all printing. The **p** flag can be used to print the modified lines. - **-e** allows multiple commands in a **sed** operation. - **-f** calls a file filled with editing commands (one command per line) to perform a number of operations at one time instead of doing them individually from the command line.	
awk	Creates reports based on the data you retrieve from files, builds databases, or performs mathematical operations against numbers in text files. Be aware of the following patterns and actions: - **-f** specifies a file containing **awk** commands to be used. - **-F** specifies the field delimiter to be used. The default is whitespace. - *$#* is used to designate fields. For example, *$6* is the sixth field in a line. - **\t** inserts a tab. - **\n** inserts a newline character. - **\f** inserts a form-feed character. - **\r** inserts a carriage return.	**awk -F: '{print $1}' /etc/passwd | sort** prints a sorted list of the user names in **/etc/passwd**. **ls -l | awk '{print "File name: "$9"\tOwner: "$3"\tModified date and time: "$6"\t"$7"\t"$8}'** customizes the **ls -l** command. From the long listing, it rearranges the ninth field to come first, labels each printed field, omits unwanted fields, and adds a tab between fields.
sort	Sorts each line of text in a file or from a text stream alphabetically. Options include the following: - **-b** ignores leading blank spaces. - **-d** uses the first alpha-numeric character and ignores special characters. - **-f** ignores case. - **-M** sorts by month. - **-n** sorts according to the string numeric value. - **-r** reverses the sort order.	**ls | sort -r** reverses the sort order of files from the **ls** command. **sort -b -d -f myfile** sorts each line in myfile and ignores leading spaces, character case, and special characters.
split	Splits lines of text from a file or a text stream into segments of a specified number of lines. Options include: - **-l, -*number*** specifies the number of lines per file. - **-b** splits text into a specified byte size instead of number of lines. - **-d** uses numeric suffixes rather than alphabetic. - **-a** specifies the number of characters in the suffix.	**split -50 -d -a 3 AllNames FiftyNames-** splits the AllNames file into individual files containing 50 lines each from the content of the AllNames file. The output is FiftyNames-001, FiftyNames-002, and so on.
tr	Transposes characters in a text stream. **tr** only works with character streams. The command uses two character sets. - The first set specifies the characters to be changed. - The second set specifies what they should be changed to. Options include the following:	**cat myfile | tr a A** changes every lowercase a to an uppercase A in the output from myfile. **cat myfile | tr abc lmn** changes each a to an l, each b to an m, and each c to an n in the output from myfile. **cat myfile | tr -d asdf** deletes each a, s, d, and f from the output of myfile. **cat myfile | tr -c e f** changes every character in the output from my file to an f except for the letter e. **cat myfile | tr -s t** changes double tt to a single t. **cat myfile | tr -t abcde lmn** ignores the dd and the e in the first set and only

	▪ **-c** changes all characters except those specified in the first set. ▪ **-d** deletes characters found in the first set. ▪ **-s** changes double-characters to single ones. ▪ **-t** truncates the first set of characters to match the size of the second set.	changes a, b, and c. Without the **-t** option, every c, d, and e, is changed to an n. Use **a-m** to specify all characters a through m.
unexpand	Changes spaces into a tab. Options include the following: ▪ **-a** specifies that the command change all occurrences. Without **-a**, the command only changes leading spaces. ▪ **-t** specifies the number of spaces to be changed. The default is eight.	**unexpand -a -t 3,4,5 myfile** changes each occurrence of three, four, or five consecutive spaces into a tab using text from myfile.
uniq	Filters identical lines from a file. The lines must be adjacent. Options include the following: ▪ **-d** prints only the duplicate lines. ▪ **-f** specifies the number of initial words to skip. Words are delimited by white space. ▪ **-s** specifies the number of initial characters to skip. ▪ **-w** specifies the number of characters to compare in each line. ▪ **-u** leaves out the duplicate lines.	**uniq myfile** omits all repeated lines in myfile. It prints the first occurrence only. **uniq -d myfile** prints only the repeated lines. **uniq -u myfile** prints only the unique lines. **uniq -f 4 myfile** skips the first four words when comparing lines. **uniq -s 4 myfile** skips the first four characters when comparing lines. **uniq -w 4 myfile** uses only the first four characters when comparing lines.
wc	Prints the number of bytes, characters, lines, or words, or the length of the longest line from the text of a file or text stream. Options include the following: ▪ **-c** specifies bytes. ▪ **-m** specifies characters. Character count is often identical to byte count. ▪ **-l** specifies line count. ▪ **-L** specifies length of the longest line. ▪ **-w** specifies word count. When no options are used, the command prints line count, word count, and byte count, respectively.	**wc myfile** displays line, word, and character count. **wc -L myfile** displays the length of the longest line in the file. **wc -m myfile** displays the number of characters in the file.