**Chapter 7**

**Pipes and Filters**

**1. Pipes and Filters**

The purpose of this lesson is to introduce us to the way that we can construct powerful Unix command lines by combining Unix commands.

**1.1 Concepts**

Unix commands alone are powerful, but when we combine them together, we can accomplish complex tasks with ease. The way we combine Unix commands is through using pipes and filters.

**1.2 using a Pipe**

The symbol | is the Unix pipe symbol that is used on the command line. What it means is that the standard output of the command to the left of the pipe gets sent as standard input of the command to the right of the pipe. Note that this functions a lot like the > symbol used to redirect the standard output of a command to a file. However, the pipe is different because it is used to pass the output of a command to another command, not a file.

Here is an example:

**$ cat apple.txt**

**core**

**worm seed**

**jewel**

**$ cat apple.txt | wc**

**3 4 21**

**$**

In this example, at the first shell prompt, we show the contents of the file apple.txt to us. In the next shell prompt, we use the cat command to display the contents of the applex.txt file, but we sent the display not to the screen, but through a pipe to the wc (word count) command. The wc command then does its job and counts the lines, words, and characters of what it got as input.

We can combine many commands with pipes on a single command line. Here's an example where we count the characters, words, and lines of the apple.txt file, then mail the results to nobody@december.com with the subject line "The count."

**$ cat apple.txt | wc | mail -s "The count" nobody@december.com**

**1.3 using filter**

In UNIX and UNIX-like operating systems, a filter is program that gets most of its data from standard input (the main input stream) and writes its main results to standard output (the main output stream). UNIX filters are often used as elements of pipelines. The pipe operator ("|") on a command line signifies that the main output of the command to the left is passed as main input to the command on the right.

List of UNIX filter programs

* awk
* cut
* grep
* head
* sed
* sort
* tail
* tac
* tee
* tr
* uniq
* wc

**1.3.1 head**

head is a program on Unix and Unix-like systems used to display the first few lines of a text file or piped data. The command-syntax is:

**$ head [options] <file\_name>**

By default, head will print the first 10 lines of its input to the standard output. The number of lines printed may be changed with a command line option. The following example shows the first 20 lines of filename:

**$ head -20 filename**

This displays the first 5 lines of all files starting with foo:

**$ head -5 foo\***

**1.3.2 tail**

tail is a program on Unix and Unix-like systems used to display the last few lines of a text file or piped data. The command-syntax is:

**$ tail [options] <file\_name>**

By default, tail will print the last 10 lines of its input to the standard output. With command line options the number of lines printed and the printing units (lines, blocks or bytes) may be changed. The following example shows the last 20 lines of filename:

**$ tail -20 filename**

This example show all lines of filename after the first 2 lines:

**$ tail +2 filename**

**File monitoring**

tail has a special command line option -f (follow) that allows a file to be monitored. Instead of displaying the last few lines and exiting, tail displays the lines and then monitors the file. As new lines are added to the file by another process, tail updates the display. This is particularly useful for monitoring log files. The following command will display the last 10 lines of messages and append new lines to the display as new lines are added to messages:

**$ tail -f /var/adm/messages**

To interrupt tail while it is monitoring, break-in with CTRL-C

**1.3.3 more**

more is better, isn't it? Better than what? Better than the cat command. cat dumps its arguments to stdout, which is the terminal (unless we redirect it with > or >>). But what if we're working on your dissertation, and we'd like to read it page by page, we'd use a command like:

**$ more dissertation.txt**

This will generate a nice page-by-page display of your masterpiece. Type

**$ man more**

at a command prompt, and check out the man page to get more details. Here we are only going to tell us the most important features of more ( i.e. the features that we use). There are three important things we should know:

7. Typing q while examining a file quits more

8. Typing /SEARCHSTRING while examining a file searches for SEARCHSTRING

9. more is a great example of a filter

**1.3.4 less**

Opposite of the more command.

Both less and more display the contents of a file one screen at a time, waiting for us to press the Spacebar between screens. This lets us read text without it scrolling quickly off your screen. The less utility is generally more flexible and powerful than more, but more is available on all Unix systems while less may not be.

The less command is a pager that allows us to move forward and backward (instead of only forward, as the more pager behaves on some systems) when output is displayed one screen at a time. :

To read the contents of a file named textfile in the current directory, enter:

**$ less textfile**

The less utility is often used for reading the output of other commands. For example, to read the output of the ls command one screen at a time, enter:

**$ ls -la | less**

In both examples, we could substitute more for less with similar results. To exit either less or more, press q . To exit less after viewing the file, press q .

**1.3.5 wc**

In Unix, to get the line, word, or character count of a document, use the wc command. At the Unix shell prompt, enter: wc filename Replace filename with the file or files for which we want information. For each file, wc will output three numbers. The first is the line count, the second the word count, and the third is the character count. For example, if we entered wc .login, the output would be something similar to the following: 38 135 847 .login To narrow thefocus of your query, we may use one or more of the following wc options:

|  |  |
| --- | --- |
| Option | Entities counted |
| -c | bytes |
| -l | lines |
| -m | characters |
| -w | words |

Note: In some versions of wc, the -m option will not be available or -c will report characters. However, in most cases, the values for -c and -m are equal.

**Syntax:**

To count the characters in a file. Here it counts the no of characters in the file abc.txt

**$ Wc –c / abc.txt**

For example, to find out how many bytes are in the .login file, we could enter:

**$ wc -c .login**

We may also pipe standard output into wc to determine the size of a stream. For example, to find out how many files are in a directory, enter:

**/bin/ls -l | wc -l**

**1.3.6 sort**

sort is a standard Unix command line program that prints the lines of its input or concatenation of all files listed in it's argument list in sorted order. The -r flag will reverse the sort order.

Example of sort in action:

**$ cat phonebook**

**Smith, Brett 555-4321**

**Doe, John 555-1234**

**Doe, Jane 555-3214**

**Avery, Cory 555-4321**

**Fogarty, Suzie 555-2314**

**$ cat phonebook | sort**

**Avery, Cory 555-4321**

**Doe, Jane 555-3214**

**Doe, John 555-1234**

**Fogarty, Suzie 555-2314**

**Smith, Brett 555-4321**

The -n option makes the program to sort according to numerical value:

**$ du /bin/\* | sort -n**

**4 /bin/domainname**

**4 /bin/echo**

**4 /bin/hostname**

**4 /bin/pwd**

**...**

**24 /bin/ls**

**30 /bin/ps**

**44 /bin/ed**

**54 /bin/rmail**

**80 /bin/pax**

**102 /bin/sh**

**304 /bin/csh**

The -n +1 option makes the program to sort according to numerical value, using the second column of data:

**$ cat zipcode**

**Adam 12345**

**Bob 34567**

**Joe 56789**

**Sam 45678**

**Wendy 23456**

**$ cat zipcode | sort -n +1**

**Adam 12345**

**Wendy 23456**

**Bob 34567**

**Sam 45678**

**Joe 56789**

The -r option just reverses the order of the sort:

**$ cat zipcode | sort -nr +1**

**Joe 56789**

**Sam 45678**

**Bob 34567**

**Wendy 23456**

**Adam 12345**

**1.3.7 tr**

tr (abbreviated from translate or transliterate) is a command in Unix-like operating systems.

When executed, the program reads from the standard input and writes to the standard output. It takes as parameters two sets of characters, and replaces occurrences of the characters in the first set with the corresponding elements from the other set. The following inputs, for instance, shift the input letters of the alphabet back by one character.

**$ echo "ibm 9000" >computer.txt**

**$ tr a-z za-y <computer.txt**

**hal 9000**

Note: when ever we are using the “tr” operator we have to use inpur rediction operator

In some older versions of tr (not POSIX-compliant), the character ranges must be enclosed in brackets, which must then be quoted against interpretation by the shell:

**$ tr "[a-z]" "z[a-y]" <computer.txt**

If it's not known in advance which variant of tr is being invoked, then in this example one would have to write the ranges in unabbreviated form (tr abcdefghijklmnopqrstuvwxyz zabcdefghijklmnopqrstuvwxy). For some applications,a single invocation is portable despite containing ranges: ROT13 can be portably implemented as tr "[A-M][N-Z][a-m][n-z]" "[N-Z][A-M][n-z][a-m]". (This works because the brackets, which are not part of the range syntax in POSIX tr, align properly in the two strings, and hence will be safely interpreted as a request to map the bracket character to itself.)

Perl also has a tr operator, which operates analogously

**1.3.8 cut**

cut is a Unix command which is typically used to extract a certain range of characters from a line, usually from a file.

**Syntax**

**cut [-c] [-f list] [-d delim] [-s] [file]**

Flags which may be used include

* -c Characters; a list following -c specifies a range of characters which will be returned, e.g. cut -c1-66 would return the first 66 characters of a line
* -f Specifies a field list, separated by a delimiter
* list A comma separated or blank separated list of integer denoted fields, incrementally ordered. The - indicator may be supplied as shorthand to allow inclusion of ranges of fields e.g. 4-6 for ranges 4 - 6 or 5 – as shorthand for field 5 to the end, etc.
* -d Delimiter; the character immediately following the -d option is the field delimiter for use in conjunction with the -f option; the default delimiter is tab. Space and other characters with special meanings within the context of the shell in use must be enquoted or escaped as necessary.
* -s Bypasses lines which contain no field delimiters when -f is specified, unless otherwise indicated.
* file The file (and accompanying path if necessary) to process as input. If no file is specified then standard input will be used

**Example**

Extract columns of data

**$ cut -f -3,5,7-9 -d ' ' infile1 > outfile1**

**-f 2,4-6 field**

**-c 35-44 character**

**-d ':' delimiter (default is a tab)**

**1.3.9 paste**

Paste is a Unix utility tool which is used to join files horizontally (parallel merging), e.g. to join two similar length files which are comma delimited. It is effectively the horizontal equivalent to the utility cat command which operates on the vertical plane of two (or more) files, i.e. by adding one file to another in order.

**Example**

To paste several columns of data together, enter:

**$ paste who where when > www**

This creates a file named www that contains the data from the names file in one column, the places file in another, and the dates file in a third. If the names, places, and dates file look like:

**Who where when**

**Sam Detroit January 3**

**Dave Edgewood February 4**

**Sue Tampa March 19**

then the www file will contain:

**Sam Detroit January 3**

**Dave Edgewood February 4**

**Sue Tampa March 19**

**1.3.10 uniq**

uniq is a Unix utility which, when fed a text file, outputs the file with adjacent identical lines collapsed to one. It is a kind of filter program. Typically it is used after sort. It can also output only the duplicate lines (with the -d option), or add the number of occurrences of each line (with the -c option).

An example: To see the list of lines in a file, sorted by the number of times each occurs:

**$ sort file|uniq -c|sort -n**

Using uniq like this is common when building pipelines in shell scripts.

**Switches**

* -u Print only lines which are not repeated in the original file
* -d Print one copy only of each repeated line in the input file.
* -c Generate an output report in default style except that each line is preceded by a count of the number of times it occurred. If this option is specified, the -u and -d options are ignored if either or both are also present.
* -i Ignore case differences when comparing lines
* -s Skips a number of characters in a line
* -w Specifies the number of characters to compare in lines, after any characters and fields have been skipped
* --help Displays a help message
* --version Displays version number on stdout and exits.