Linux Command Line

Manipulating Text

cat

`cat` command is often used to read and print files as well as for simply viewing file contents

The `tac` command prints the lines of a file in reverse order. The syntax of `tac` is exactly the same as for `cat`

Command	Usage
`cat file1`	Display contents of file1 on the terminal
`cat file1 file2`	Concatenate multiple files and display the output; i.e., the entire content of the first file is followed by that of the second file
`cat file1 file2 > newfile`	Combine multiple files and save the output into a new file
`cat file >> existingfile`	Append a file to the end of an existing file
`cat > file`	Any subsequent lines typed will go into the file until CTRL-D is typed
`cat >> file`	Any subsequent lines are appended to the file until CTRL-D is typed

echo

`echo` simply displays (echoes) text.
`echo` is particularly useful for viewing the values of environment variables (built-in shell variables). For example, echo `$USERNAME` will print the name of the user who has logged into the current terminal.

Command	Usage
`echo string > newfile`	The specified string is placed in a new file
`echo string >> existingfile`	The specified string is appended to the end of an existing file
`echo $variable`	The contents of the specified environment variable are displayed

head & tail

`head` reads the first few lines of each named file (10 by default) and displays it on standard output. You can give a different number of lines in an option.
For example, If you want to print the first 5 lines from `app_log.txt`, use the following command: `$ head -n 5 app_log.txt`

`tail` prints the last few lines of each named file and displays it on standard output. By default, it displays the last 10 lines.

To continually monitor output in a log file: `$ tail -f atmtrans.txt`

z commands

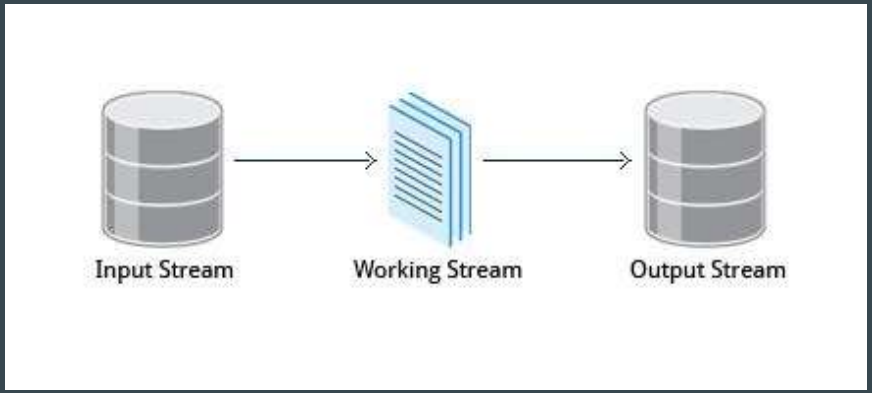
When working with compressed files many standard commands cannot be used directly. The z family of commands can be used for working with compressed files

Command	Usage
`$ zcat compressed-file.txt.gz`	To view a compressed file
`$ zless <filename>.gz` or `$ zmore <filename>.gz`	To page through a compressed file
`$ zdiff filename1.txt.gz filename2.txt.gz`	To compare two compressed files

sed

`sed` is a powerful text processing tool and is one of the oldest earliest and most popular UNIX utilities. It is used to modify the contents of a file. Its name is an abbreviation for stream editor.

Data from an input source/file (or stream) is taken and moved to a working space. The entire list of operations/modifications is applied over the data in the working space and the final contents are moved to the standard output space (or stream).



sed syntax

Command	Usage
`sed -e command <filename>`	Specify editing commands at the command line, operate on file and put the output on standard out (e.g., the terminal)
`sed -f scriptfile <filename>`	Specify a scriptfile containing sed commands, operate on file and put output on standard out

sed Basic Operations

Command	Usage
`sed s/pattern/replace_string/ file`	Substitute first string occurrence in a line
`sed s/pattern/replace_string/g  file`	Substitute all string occurrences in a line
`sed 1,3s/pattern/replace_string/g file`	Substitute all string occurrences in a range of lines
`sed -i s/pattern/replace_string/g file`	Save changes for string substitution in the same file
`$ sed s/pattern/replace_string/g file1 > file2`	Replace all occurrences of pattern with replace_string in file1 and move the contents to file2

awk

`awk` is used to extract and then print specific contents of a file and is often used to construct reports.

`awk` has the following features:

- It is a powerful utility and interpreted programming language.
- It is used to manipulate data files, retrieving, and processing text.
- It works well with fields (containing a single piece of data, essentially a column) and records (a collection of fields, essentially a line in a file).

awk Syntax

Command	Usage
`awk 'command' var=value file`	Specify a command directly at the command line
`awk -f scriptfile var=value file`	Specify file that contains the script to be executed along with f

awk Basic Operations

Command	Usage
`awk '{ print $0 }' /etc/passwd`	Print entire file
`awk -F: '{ print $1 }' /etc/passwd`	Print first field (column) of every line, separated by a space
`awk -F: '{ print $1 $6 }' /etc/passwd`	Print first and sixth field of every line

The `-F` option specifies a particular field separator character.

sort

`sort` is used to rearrange the lines of a text file either in ascending or descending order, according to a sort key. You can also sort by particular fields of a file. The default sort key is the order of the ASCII characters (i.e., essentially alphabetically).

Syntax	Usage
`sort <filename>`	Sort the lines in the specified file
`cat file1 file2 | sort`	Append the two files, then sort the lines and display the output on terminal
`sort -r <filename>`	Sort the lines in reverse order

When used with the `-u` option, sort checks for unique values after sorting the records (lines). It is equivalent to running `uniq`

uniq

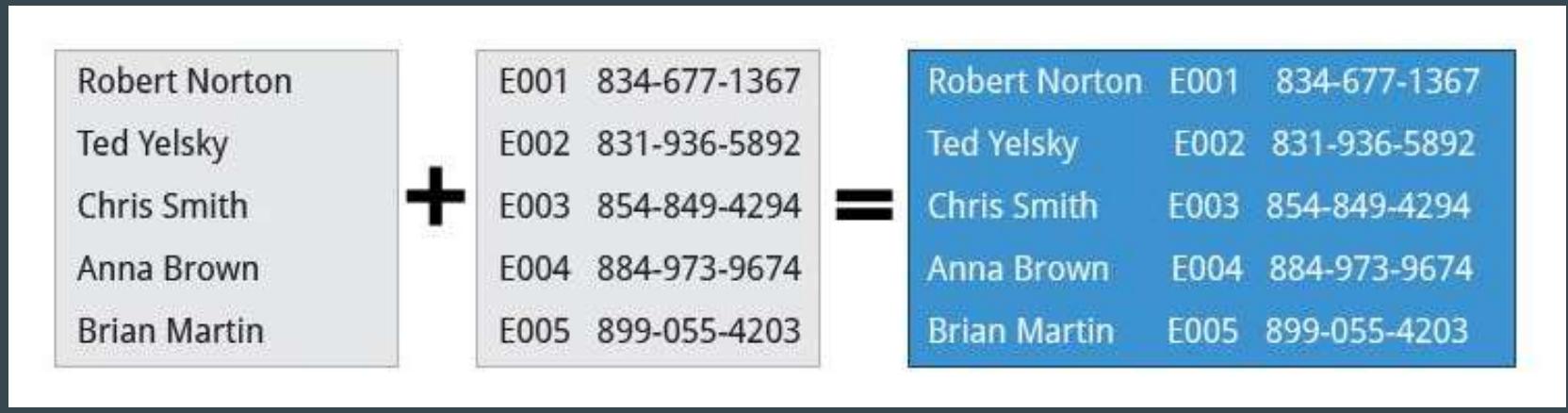
uniq is used to remove duplicate lines in a text file and is useful for simplifying text display. uniq requires that the duplicate entries to be removed are consecutive. Therefore one often runs sort first and then pipes the output into uniq; if sort is passed the -u option it can do all this in one step.

Command	Usage

`sort file1 file2 | uniq > file3`
OR
`sort -u file1 file2 > file3`
	To remove duplicate entries from files

`uniq -c filename`	To count the number of duplicate entries

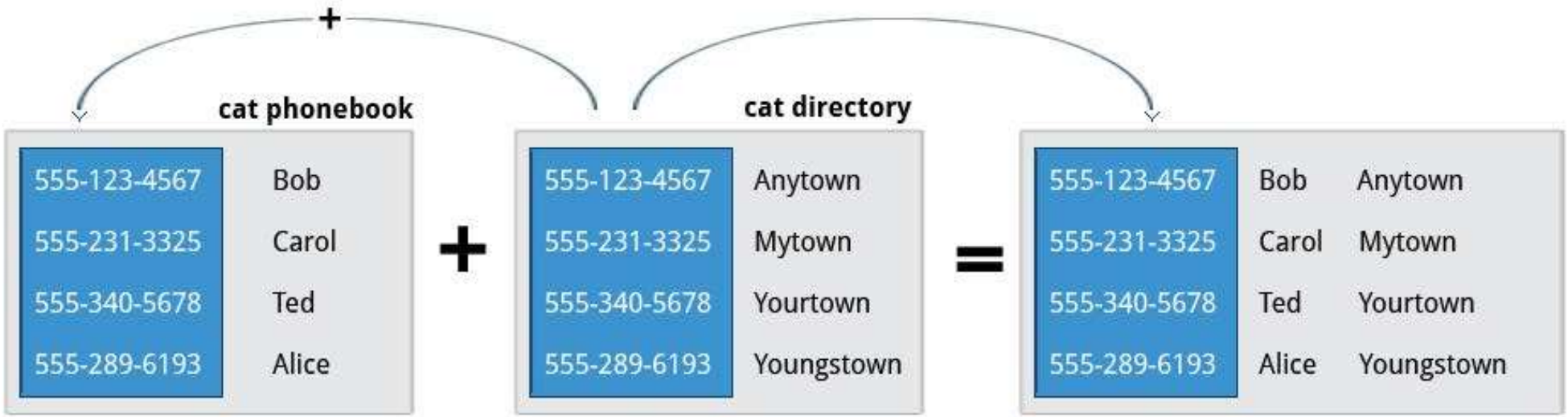
paste



`paste` accepts the following options:

- `-d` delimiters, which specify a list of delimiters to be used instead of tabs for separating consecutive values on a single line. Each delimiter is used in turn; when the list has been exhausted, paste begins again at the first delimiter.
- `-s` which causes paste to append the data in series rather than in parallel; that is, in a horizontal rather than vertical fashion.

join



Used to join files with similar columns.

```
$ join file1 file2
```

split

`split` is used to break up (or split) a file into equal-sized segments for easier viewing and manipulation, and is generally used only on relatively large files. By default split breaks up a file into 1,000-line segments. The original file remains unchanged, and set of new files with the same name plus an added prefix is created.

```
$ split infile <Prefix>
```

Regular Expressions

Regular expressions are text strings used for matching a specific pattern, or to search for a specific location, such as the start or end of a line or a word. Regular expressions can contain both normal characters or so-called metacharacters, such as * and $.

Pattern	Description
`.(dot)`	Match any single character
`a|z`	Match a or z
`$`	Match end of string
`*`	Match preceding item 0 or more times

Regular Expressions - Example

the quick brown fox jumped over the lazy dog

Some of the patterns that can be applied to this sentence are as follows:

Pattern	Description
`a..`	matches azy
`b.|j.`	matches both br and ju
`..$`	matches og
`l.*`	matches lazy dog
`l.*y`	matches lazy
`the.*`	matches the whole sentence

grep

`grep` is extensively used as a primary text searching tool. It scans files for specified patterns and can be used with regular expressions

Pattern	Description
`grep [pattern] <filename>`	Search for a pattern in a file and print all matching lines
`grep -v [pattern] <filename>`	Print all lines that do not match the pattern
`grep [0-9] <filename>`	Print the lines that contain the numbers 0 through 9
`grep -C 3 [pattern] <filename>`	Print context of lines (specified number of lines above and below the pattern) for matching the pattern. Here the number of lines is specified as 3.

wc

wc (word count) counts the number of lines, words, and characters in a file or list of files.

For example `wc -l filename` prints the number of lines contained in a file

Option	Description
`-l`	display the number of lines
`-c`	display the number of bytes
`-w`	display the number of words