Linux Reference

Bailey Harrington

September 2019

Contents

Commands

`apropos` *Search for commands*

Description

Searches within command names and `man` pages and returns a list of potential matches. Useful for finding the correct command for a job.

Usage

`apropos` *searchterm*

Arguments

`searchterm` term to look for in command documentation

Output

Prints out a list of matches with the command name and a brief description.

Examples

```
apropos view
apropos find
```

`basename` *get the basename of a file*

Description

Removes the leading directories from the front of a file name, and can also remove a suffix, specified as a second argument, or using a flag.

Usage

`basename` *`file [suffix]`*

Arguments

`file`	the name of the file to print
`suffix`	suffix to remove (optional)

Output

A portion of the file name.

Useful options

`-a`	take multiple files; should come after other flags
`-s`	suffix (for use in conjunction with -a)

Examples

```
basename path/to/directory/file.txt
basename file.txt .txt
basename -a file.txt file2.txt
basename -s .txt -a file.txt file2.txt
basename path/to/directory/file.txt .txt
```

`cat` *print a file*

Description

Prints the contents of a file to standard output.

Usage

`cat` *`file [file2...]`*

Arguments

`file`	the name of the file to print
`file2...`	the name of additional files to print (optional)

Output

The complete contents of the file(s) provided as input.

Useful options

`-b`	number non-empty lines
`-s`	squeeze empty lines
`-v`	display non-printing characters

Examples

```
cat -b file.txt
cat file.txt file2.txt
```

`cd` *change directory*

Description

Navigate to a new directory. Both explicit paths or designated shortcuts may be used to specify the destination.

Usage

`cd` *[path]*

Arguments

`path`	the path to the destination directory

Useful options

`..`	moves up one level
`-`	moves to the previous directory
∼	moves to the home directory

Examples

```
cd path/to/directory
cd ..
cd ../..
cd ../path
cd -
cd ∼
```

`chmod` *manage file permissions*

Description

Allows modification of read, write, and execute permissions for files. There are three sets of permissions: user, group, and others. The permissions are designated as: `r`, `w`, and `x` for (r)ead, (w)rite, and e(x)ecute. Extended permissions exist for more advanced usage.

Usage

`chmod` *[operations] file*

Arguments

`file`	file whose permissions will be modified

Useful operations

`+x`	make file executable for owner
`go-w`	deny write permission to group and others

Examples

```
chmod +x file.txt
chmod go-w file.txt
```

`clear` *clear the screen*

Description

Moves the terminal prompt to the top of the window, leaving the screen empty.

Usage

```
clear
```

Examples

```
clear
```

`cp` *copy a file*

Description

Makes a copy of one or more specified files or directories. There are several ways this can be used, determined by the types of arguments given: copy a file locally, with a different name; copy one or more files to a new location, retaining the original names; copy a directory and its contents locally, with a different name; copy one or more directories and their contents to a new location, retaining the original names.

Usage

```
cp file newfile
cp file [file2...]  dest_directory
cp directory newdirectory
cp directory [directory2...]  dest_directory
```

Arguments

`file`	the file to be copied
`newfile`	the name of the copy being created
`file2...`	additional files to be copied (optional)
`dest_directory`	the destination location for the copies
`directory`	the directory to be copied
`newdirectory`	the name of the copy being created
`directory2...`	additional directories to be copied (optional)

Useful options

`-n`	do not overwrite an existing file

Examples

```
cp -n file.txt newfile.txt
cp directory/ another/directory/
cp * path/to/directory/
```

`echo` *write arguments to standard output*

Description

Writes the value of its arguments to the standard output. If variables or expressions are included, they are first evaluated, then written out.

Usage

```
echo [options] argument [argument2...]
```

Arguments

`argument`	argument to evaluate and write to the shell
`argument2...`	additional arguments to be written out

Output

Prints the evaluated argument(s) to the shell in the order in which they are given. By default, the output is followed by a newline character.

Useful options

`-e`	enables evaluation of escaped characters
`-n`	omits the newline character at the end of the output

Examples

```
echo "this will be printed"
x=5
echo $x
echo "The date is:  $(date +%D)"
```

---

grep *file pattern searcher*

Description

Searches within the file(s) given for a specified string. More advanced searches can be done using regular expressions.

Usage

`grep` *[options] pattern file [file2...]*

Arguments

`pattern`	the `pattern` to look for in the given file(s)
`file`	the file to be searched for instances of `pattern`

Output

Prints out each line containing a match from within `file`.

Useful options

`-c`	outputs the number of lines containing `string`
`-C`	print two lines before and after the match
`-e` *`pattern`*	enables the use of regular expressions
`-f` *`file`*	a file containing a list of `pattern`; one per line
`-F` *`"string"`*	specifies `pattern` should be interpreted literally
`-i`	ignore case
`-l`	list input files that do match
`-L`	list input files that do not match
`-m` *`#`*	stop searching after finding `#` matches
`-n`	print the line number for each match
`-r`	recursively search through directories
`-v`	print non-matching lines
`-w`	matches must be full-word matches

Examples

```
grep pattern file.txt
grep -Fwnc "pattern" file.txt file2.txt
grep -r pattern directory
```

`head` *print the beginning of the file*

Description

By default, prints out the first ten lines of each file given.

Usage

`head [options]` *`file [file2...]`*

Arguments

`file`	the file to print the top of (may be piped)
`file2...`	additional files (optional)

Output

Prints the first ten lines of the file to standard output.

Useful options

`-n #`	specifies the number of lines to print

Examples

```
head file.txt
head -n 20 file.txt file2.txt
```

`history` *print the command history*

Description

Prints the command history to the standard output in list form. Previous commands may be rerun using *!#* notation, where `#` is the number of the command to be run.

Usage

```
history
```

Output

Prints the list of previously run commands to the standard output. The extent of the history depends on values of environmental variables.

Examples

```
history
!500
```

`less` *preview a file*

Description

Opens the file passed in a navigator so its contents may be examined.

Usage

`less` *[options] file*

Arguments

`file`	the file to be previewed

Useful options

`-N`	prints line numbers
`-s`	squeeze empty lines
`-S`	do not wrap text

Examples

```
less file.txt
less -NS file.txt
```

`ls` *list files*

Description

Prints a list of files and subdirectories in the current working directory.

Usage

`ls` *`[options] [file...]`*

Arguments

`file...`	files and directories to list (optional)

Output

By default, lists the contents of the current working directory, excluding hidden files. Subsets of files or other directories may be specified.

Useful options

`-a`	show hidden files
`-h`	print file sizes in human-readable format
`-l`	print detailed information on each file
`-r`	reverse the order in which files are listed
`-t`	list files by time modified

Examples

```
ls path/to/directory
ls -ahlrt
ls *.txt
```

`man` *`display manual pages for commands`*

Description

Displays documentation for a command, if the documentation is found on the system. There may not be a `man` page for every command

installed.

Usage

`man` *`command`*

Arguments

`command`	the command whose documentation is displayed

Examples

```
man grep
man ls
```

`mkdir` *create a new directory*

Description

Makes a new directory inside the current directory, or another specified location.

Usage

`mkdir` *`directory`*
`mkdir` *`path/to/dir`*

Arguments

`directory`	the name of the directory to create
`path/to/dir`	location in which to create `dir`

Useful options

`-p`	creates any nonexistent parent directories

Examples

```
mkdir directory
mkdir path/to/directory
```

`mv` *move or rename files*

Description

Moves one or more specified files or directories. There are several ways this can be used, determined by the types of arguments given: rename a file (move locally to a new name); move a file to a new directory, retaining the original name; rename a directory (move locally to a new name); and move a directory and its contents to a new location, retaining the original names.

Usage

```
mv file renamed_file
mv file dest_directory
mv directory renamed_dir
mv directory dest_directory
```

Arguments

`file`	the file to be moved/renamed
`renamed_file`	the new name for the file
`dest_directory`	the new location for the file/directory
`directory`	directory to be moved/renamed
`renamed_dir`	the new name for the directory

Useful options

`-i`	warn if the `mv` will overwrite an existing file
`-n`	do not overwrite an existing file

Examples

```
mv oldfile.txt newfile.txt
mv -n file.txt path/to/dest_directory
mv -i directory path/to/dest_directory
```

`pwd` *print the working directory*

Description

Prints the current working directory.

Usage

```
pwd
```

Output

The full path to the current working directory.

Examples

```
pwd
```

`rm` *remove files*

Description

Remove the specified files. May also be used to remove directories using the `-r` option; this does not provide a warning in the case of non-empty directories.

Usage

`rm` *`file [file2...]`*

Arguments

`file`	the file to remove
`file2...`	additional files to remove (optional)

Useful options

`-i`	request confirmation before removing each file
`-r`	recursively remove a directory and its contents

Examples

```
rm file.txt
rm -r directory
```

`rmdir` *remove directories if empty*

Description

Removes a directory if and only if it is empty.

Usage

`rmdir` *`directory [directory2...]`*

Arguments

`directory`	`directory` to be removed
`directory2...`	additional directories (optional)

Examples

```
rmdir directory
```

`tac` *print a file in reverse*

Description

The reverse of cat. Prints the contents of a file to standard output, starting at the end.

Usage

`tac` *file [file2...]*

Arguments

`file`	the file to print
`file2...`	additional files to print (optional)

Output

The complete contents of the file(s) provided as input, in reverse.

Examples

```
tac file.txt
```

`tail` *print the end of the file*

Description

By default, prints out the last ten lines of each file given.

Usage

`tail` *`[options] file [file2...]`*

Arguments

`file`	the file to print the bottom of (may be piped)
`file2...`	additional files (optional)

Output

Prints the last then lines of the file to standard output.

Useful options

`-f`	continually update the end of the file
`-n #`	print the last # lines
`-n +#`	print the file from line # to the end
`-q`	omit headers if there are multiple files
`-r`	print the lines in reverse

Examples

```
tail file.txt
tail -qn +50 file.txt file2.txt
```

`wc` *word count*

Description

Prints summary statistics for the given files.

Usage

`wc` *`[options] file [file2...]`*

Arguments

`file`	the file to summarise
`file2...`	additional files (optional)

Output

By default, prints the line, word, and character counts for the specified file(s).

Useful options

`-c`	print byte count
`-l`	print line count
`-m`	print character count
`-w`	print word count

Examples

```
wc file.txt
wc -l file.txt file2.txt
```

`whatis` *briefly describe what a command does*

Description

Searches within names and one-line descriptions of commands' functions for the searchterm specified. Useful to learn what a command does, or see other commands with similar functions.

Usage

`whatis` *`searchterm`*

Arguments

`searchterm` a known command name or a string

Output

A list of command names and their one-line descriptions. A name/description pair will be printed if either contains `searchterm` as a full-word match. This is different to `apropos`, where `searchterm` may occur anywhere in the documentation, and may also be a substring in a larger word.

Examples

```
whatis mkdir
whatis tail
```

`which` *locate a program file*

Description

Locates a program file in the user's path.

Usage

`which` *program*

Arguments

`program` name of the program file to search for

Output

Prints out the full path for the program file.

Examples

```
which python
```

Special characters

`pipe` *ways to redirect output from stdout*

Description

Used following a command to pass the command's output into another command or write it to a file.

Redirection characters

|	feed output to another command
>	write output to file (overwrites if file exists)
>>	write output to end of file (appends if file exists)

Examples

`ls *.txt | grep string`
searches for string inside all .txt files

`echo "Some text." > file.txt`
creates/overwrites file.txt and writes "Some text."

`echo "Some text." >> file.txt`
creates/opens file.txt; adds "Some text." to the end

`wildcards` *specify all files that fit a pattern*

Description

Special notation that can be used to indicate one or more characters that may be unknown, or to specify a group of files/directories that are similarly named.

Wildcard characters

`*`	zero or more characters
`?`	one character

Useful options

`doc?.tsv`	matches files like doc1.tsv or doca.tsv
`doc.???`	matches files like doc.tsv, doc.txt, doc.csv
`*.tsv`	matches any file ending in .tsv
`doc*.tsv`	matches doc.tsv, doc1.tsv, doc1234.tsv, et cetera
`doc.*`	matches doc.tsv, doc.txt, doc.tsv.gz
`doc*`	matches doc.tsv, doc.txt, doc.tsv.gz, document1.tsv

Index

V

W