Command Index

cat

Purpose: Concatenate files and display their contents.

Usage syntax: `cat [OPTIONS] [FILE]`

```
$ cat ShoppingList.txt
Milk
Eggs
Cheese
Tacos
```

Output of the cat command

The `cat` command concatenates and displays files. Executing `cat` on the `ShoppingList.txt` file displays its contents as shown in the above example.

Common usage examples:

`cat [FILE]`	Display the contents of the specified file
`cat [FILE1] [FILE2] [ETC]`	Join the specified files
`cat -n [FILE]`	Display numbered output for each line
`cat -s [FILE]`	Suppress blank lines

Creating a file using cat

Cat > filename

Abcdef.....

^d

`$ cat` *target-file(s)*

displays the contents of *target-file(s)* on the screen, one after the other. You can also use it to create files from keyboard input as follows

```
$ cat > hello.txt ↵
hello world! ↵
```
[ctrl-d]
```
$ ls hello.txt ↵
hello.txt
$ cat hello.txt ↵
hello world!
$
```

mkdir / rmdir

Purpose: Create/remove directories.

Usage syntax: `mkdir [OPTIONS] [DIRECTORY]`

```
# mkdir test
# ls -ld test/
drwxr-xr-x 2 root root       4096 Jun  4 09:00 test
```

Creating a directory with mkdir

Usage syntax: `rmdir [DIRECTORY]`

Common usage examples:

`mkdir [DIRECTORY]`	Create the specified directory
`mkdir -p [PATH/DIRECTORY]`	Create parent directories if needed
`rmdir [DIRECTORY]`	Remove the specified directory

pwd

Purpose: Display the current/working directory.

Usage syntax: `pwd`

```
$ pwd
/home/nick
```

Using the pwd command the display the current directory

The **pwd** command (short for **P**rint **W**orking **D**irectory) displays your current location within the file system. In the above example, executing **pwd** displays `/home/nick` as the current working directory.

cd

Purpose: Change (navigate) directories.

Usage syntax: `cd [DIRECTORY]`

```
$ cd /etc
$ pwd
/etc
```

Using the cd command to navigate to the /etc directory

Common usage examples:

`cd [DIRECTORY]`	Navigate to the specified directory
`cd`	Navigate to the user's home directory
`cd -`	Go back to the previous working directory
`cd ..`	Navigate up one level in the directory tree

who / whoami

Purpose: Display who is logged into the system.

Usage syntax: `who [OPTIONS]`

Usage syntax: `whoami`

```
$ whoami
nick
```

Using whoami to display the name of the current user

Common usage examples:

`who`	Display who is currently logged into the system
`who -b`	Display the last system boot time
`who -r`	Display the current run level
`whoami`	Display the name of the current user

touch

Purpose: Update time stamps on a file.

Usage syntax: `touch [OPTIONS] [FILE]`

```
$ ls -l testfile
-rw-r--r-- 1 root root 251 2009-04-21 15:50 testfile
$ touch testfile
$ ls -l testfile
-rw-r--r-- 1 root root 251 2009-05-23 14:54 testfile
$ date
Sat May 23 14:54:35 CDT 2009
```

Using the touch command to update the time stamp on a file

If the file does not exist, the `touch` command will create an empty file with the specified file name, as demonstrated in the next example.

```
$ ls -l MyFile
ls: cannot access MyFile: No such file or directory
$ touch MyFile
$ ls -l MyFile
-rw-r--r-- 1 nick nick 0 2009-05-23 14:54 MyFile
```

Creating a new empty file with the the touch command

Common usage examples:

`touch [FILE]`	Update the time stamp on the specified file
`touch -a [FILE]`	Update the access time stamp on the specified file
`touch -m [FILE]`	Update the modified time stamp on the specified file

ls

Purpose: List the contents of a directory.

Usage syntax: `ls [OPTIONS] [DIRECTORY/FILE]`

```
$ ls
Notes.txt  ShoppingList.txt  ToDoList.txt
```

Typical output of the ls command

Executing the `ls` command displays a simple list of files in the current directory, as shown in the above example. To see more information about the files in a directory you can use command line options to activate additional features, as demonstrated in the next example.

```
$ ls -l
-rw-r--r-- 1 nick sales 35068 2009-05-19 08:41 Notes.txt
-rw-r--r-- 1 nick sales    23 2009-05-19 08:43 ShoppingList.txt
-rw-r--r-- 1 nick sales    37 2009-05-19 08:43 ToDoList.txt
```

Using the -l option with the ls command

Permissions	Number of Links	Owner & Group	Size	Modification Date	File or Directory
-rw-r--r--	1	nick sales	35068	2009-05-19 08:41	Notes.txt

Description of fields displayed with the ls -l command

```
$ ls -l ShoppingList.txt
-rw-r--r-- 1 nick users 254 2009-06-01 15:35 ShoppingList.txt
```

Output of the ls -l command displaying file permissions

	User	Group	Other
Symbolic	rw-	r--	r--
Meaning	Read & Write	Read only	Read only

Example of file permissions

Common usage examples:

`ls`	Display a basic list of files in the current directory
`ls [DIRECTORY]`	Display a basic list of files in the specified directory
`ls -l`	List files with details
`ls -la`	List hidden files
`ls -lh`	List file sizes in "human readable format" (KB, MB, etc.)
`ls -R`	Recursively list all subdirectories
`ls -d [DIRECTORY]`	List only the specified directory (not its contents)

mv

Purpose: Move or rename files and directories.

Usage syntax: `mv [OPTIONS] [SOURCE] [DESTINATION]`

cp

Purpose: Copy files and directories.

Usage syntax: `cp [OPTIONS] [SOURCE] [DESTINATION]`

```
$ cp MyFile MyFile.copy
$ ls -l
-rw-r--r-- 1 nick nick    55 2009-05-20 15:32 MyFile
-rw-r--r-- 1 nick nick    55 2009-05-20 15:32 MyFile.copy
```

Creating a copy of a file

Common usage examples:

`cp [SOURCE] [DEST]`	Create a copy of the specified file
`cp -r [SOURCE] [DEST]`	Recursively copy a directory
`cp -i [SOURCE] [DEST]`	Prompt before overwriting the destination file
`cp -f [SOURCE] [DEST]`	Force overwriting if the destination file exists
`cp -v [SOURCE] [DEST]`	Display verbose messages while copying

rm

Purpose: Remove files.

Usage syntax: `rm [OPTIONS] [FILE]`

```
$ rm MyFile
$ ls -l MyFile
ls: cannot access MyFile: No such file or directory
```

Using the rm command to remove a file

```
$ rm -i MyFile
rm: remove regular file 'MyFile'? y
```

Using the -i option with the rm command for interactive prompts

tree

Purpose: Display the contents of a directory in a tree hierarchy format.

Usage syntax: `tree [OPTIONS] [DIRECTORY]`

Common usage examples:

`tree`	Display the contents of the current directory in tree form
`tree [DIR]`	Display the contents of the specified directory in tree form
`tree -a`	Include hidden files in the tree listing
`tree -d`	List directories only
`tree -L [NUM]`	List the specified number of levels deep

find

Purpose: Search for files and directories.

Usage syntax: `find [PATH] [OPTIONS] [CRITERIA]`

```
# find / -name hosts
/etc/avahi/hosts
/etc/hosts
/usr/share/hosts
```

Using the find command to locate files with the word "hosts" in their name

Common usage examples:

`find [PATH] -name [NAME]`	Find files with the specified name
`find [PATH] -user [USERNAME]`	Find files owned by the specified user
`find [PATH] -size [FILESIZE]`	Find files larger than the specified size
`find [PATH] -mtime 0`	Find files modified in the last 24 hours

date

Purpose: Display or set the system clock.

Usage syntax: `date [OPTIONS] [TIME/DATE]`

```
$ date
Wed Jun 10 20:33:27 CDT 2009
```

Output of the date command

Common usage examples:

`date`	Display the time and date
`date -s [HH:MM]`	Set the time
`date -s ["MM/DD/YYYY HH:MM"]`	Set the time and date

cal

Purpose: Display a calendar on the command line.

Usage syntax: `cal [OPTIONS] [MONTH] [YEAR]`

Common usage examples:

`cal`	Display a calendar for the current month
`cal -m`	Display Monday as the first day of the week
`cal [MONTH] [YEAR]`	Display a calendar for the specified month and year
`cal [YEAR]`	Display a calendar for the specified year
`cal -y`	Display a calendar for the current year

Redirection

The output of a command can be redirected to other locations such as a text file. Redirection is initiated by using the > character on the keyboard.

```
$ date > date.txt
$ ls -l date.txt
-rw-r--r-- 1 nick nick 29 2009-06-10 11:37 date.txt
```

Redirecting the output of the date command to a file

```
$ date >> date.txt
```

Appending the output of a command to a file

WC

Purpose: Count the number of lines, words, and characters in a file.

Usage syntax: `wc [OPTIONS] [FILE]`

```
$ wc /etc/hosts
 10 28 251 /etc/hosts
```

Output of the wc command

The `wc` command (short for Word Count) displays the total number of lines, words, and characters in the specified file. The above example displays the totals for the `/etc/host` file. The table below explains the output fields generated by the `wc` command.

Lines	Words	Characters	File
10	28	251	/etc/hosts

Output fields of the wc command

Common usage examples:

`wc [FILE]`	Display the number of lines, words, and characters in a file
`wc -w [FILE]`	Display the number of words in a file
`wc -l [FILE]`	Display the number of lines in a file
`wc -c [FILE]`	Display the number of characters in a file

head

Purpose: Display the first part of a file.

Usage syntax: `head [OPTIONS] [FILE]`

```
$ head -n 2 ShoppingList.txt
Milk
Eggs
```

Using the head command to display the first two lines of a file

Common usage examples:

`head [FILE]`	Display the first 10 lines of the specified file
`head -n [NUM] [FILE]`	Display the specified number of lines

tail

Purpose: Display the last part of a file.

Usage syntax: `tail [OPTIONS] [FILE]`

```
$ tail -n 2 ShoppingList.txt
Cheese
Tacos
```

Displaying the last two lines in a file with tail

Common usage examples:

`tail [FILE]`	Display the last 10 lines of the specified file
`tail -n [NUM] [FILE]`	Display the specified number of lines
`tail -f [FILE]`	Follow the file as it grows

sort

Purpose: Sort the contents of an input stream or file.

Usage syntax: `sort [OPTIONS] [FILE]`

```
$ cat ShoppingList.txt
Milk
Eggs
Cheese
Tacos
$ sort ShoppingList.txt
Cheese
Eggs
Milk
Tacos
```

Using the sort command to sort a file

Common usage examples:

`sort [FILE]`	Sort and display the specified file
`sort -r [FILE]`	Reverse sort the specified file
`[COMMAND] | sort`	Sort the output of the specified command

diff

Purpose: Compare files.

Usage syntax: `diff [OPTIONS] [FILE]`

```
$ diff ShoppingList.txt ShoppingList.old
4c4
< Tacos
---
> Nachos
```

Default output of the diff command

The **diff** command allows you to compare two text files line by line and display the differences between them. The **diff** command provides two types of output:

1. Single column (default)
2. Two column side-by-side comparison (activated with the `-y` option)

In the above example the **diff** command displays the default single column output which only shows the differences between the two files. Indicators are used to mark the differing lines:

< Indicates the text in the first file

> Indicates the text in the second file

Common usage examples:

`diff [FILE1] [FILE2]`	Compare files and display differences
`diff -y [FILE1] [FILE2]`	Compare files side by side
`diff -i [FILE1] [FILE2]`	Ignore case when comparing files

chmod

Purpose: Change file and directory permissions.

Usage syntax: `chmod [OPTIONS] [MODE] [DIRECTORY/FILE]`

```
# chmod 664 ShoppingList.txt
# ls -l ShoppingList.txt
-rw-rw-r-- 1 root root 23 2009-05-27 22:31 ShoppingList.txt
```

Using the chmod command to change file permissions

The table below provides a cross reference of symbolic and octal permissions.

Permission	Symbolic	Octal
Read	r	4
Write	w	2
Execute	x	1
None	-	0

Permissions cross reference

The sum of the octal permissions becomes what is known as the *mode*. The valid modes are described in the following table.

Mode	Octal	Symbolic	Effective Permission
7	4+2+1	rwx	Read/Write/Execute
6	4+2	rw-	Read/Write
5	4+1	r-x	Read/Execute
4	4	r--	Read
0	0	---	None

Mode cross reference

The combination of 3 modes determines the permissions for the file. A mode of 664 would create `rw-rw-r--` permissions giving read/write access to the user and group, and read only to everyone else.

Wildcards

Wildcards are used to pattern match one against one or more text elements. They are helpful on the command line for performing bulk tasks such as listing or removing groups of files. The table below lists the different types of wildcards that can be used on the command line.

Wildcard	Function
*	Matches 0 or more characters
?	Matches 1 character
[abc]	Matches one of the characters listed
[a-c]	Matches one character in the range
[!abc]	Matches any character not listed
[!a-c]	Matches any character not listed in the range
{tacos,nachos}	Matches one word in the list

Types of wildcards

The asterisk (*) is the simplest and most helpful wildcard. The example below demonstrates using the asterisk wildcard to display all files that match a file name.

```
$ ls -l /etc/host*
-rw-r--r-- 1 root root  92 2008-12-23 12:53 /etc/host.conf
-rw-r--r-- 1 root root   6 2009-04-23 15:50 /etc/hostname
-rw-r--r-- 1 root root 251 2009-05-22 14:55 /etc/hosts
-rw-r--r-- 1 root root 579 2009-04-20 09:14 /etc/hosts.allow
-rw-r--r-- 1 root root 878 2009-04-20 09:14 /etc/hosts.deny
```

Listing files using the asterisk wildcard

Typing `ls -l /etc/host*` lists all the files in the `/etc` directory that start with the word *host*. Other examples of wildcards are demonstrated below.

```
$ ls -l /etc/hosts.{allow,deny}
-rw-r--r-- 1 root root 579 2009-04-20 09:14 /etc/hosts.allow
-rw-r--r-- 1 root root 878 2009-04-20 09:14 /etc/hosts.deny
$ ls -l /etc/hosts.[!a]*
-rw-r--r-- 1 root root 878 2009-04-20 09:14 /etc/hosts.deny
$ ls -l /etc/host?
-rw-r--r-- 1 root root 251 2009-05-22 14:55 /etc/hosts
```

Examples of other wildcards

In this example, the first command uses `{allow,deny}` to display all matches that end with the word allow or deny. The second command uses `[!a]*` to display matches that do not begin with the letter *a* (after the period). The third example uses the `?` wildcard to match only a single character.

Pipes

Pipes (also referred to as pipelines) can be used to direct the output of one command to the input of another. Pipes are executed using the | key (usually located above the backslash key) on the keyboard.

```
$ ls -l /etc | more
```

Using `ls -l` on the `/etc` directory would normally rapidly scroll the contents of the directory across the screen. Piping the output of `ls -l` to the `more` command (discussed on page 71) displays the contents of the `/etc` directory one page at a time.

Ls –l |wc –l

Displays the number of files in the current directory.

grep

Purpose: Match patterns and filter data.

Common usage examples:

`grep [STRING] [FILE]`	Display matching lines in a file
`grep -c [STRING] [FILE]`	Count the number of matches in a file
`grep -i [STRING] [FILE]`	Ignore case when matching
`[COMMAND] | grep [STRING]`	Filter a command's output to match a string