**Top 25 Bash Commands**

**Quick note:** Anything encased in [ ] means that it’s optional. Some commands can be used without options or specifying files.

**ls — List directory contents**

ls is probably the most common command. A lot of times, you’ll be working in a directory and you’ll need to know what files are located there. The ls command allows you to quickly view all files within the specified directory.

* Syntax: ls [option(s)] [file(s)]
* Common options: -a, -l

**echo — Prints text to the terminal window**

echo prints text to the terminal window and is typically used in shell scripts and batch files to output status text to the screen or a computer file. Echo is also particularly useful for showing the values of environmental variables, which tell the shell how to behave as a user works at the command line or in scripts.

* Syntax: echo [option(s)] [string(s)]
* Common options: -e, -n

**touch — Creates a file**

touch is going to be the easiest way to create new files, but it can also be used to change timestamps on files and/or directories. You can create as many files as you want in a single command without worrying about overwriting files with the same name.

* Syntax: touch [option(s)] file\_name(s)
* Common options: -a, -m, -r, -d

**mkdir — Create a directory**

mkdir is a useful command you can use to create directories. Any number of directories can be created simultaneously which can greatly speed up the process.

* Syntax: mkdir [option(s)] directory\_name(s)
* Common options: -m, -p, -v

**grep — search**

grep is used to search text for patterns specified by the user. It is one of the most useful and powerful commands. There are often scenarios where you’ll be tasked to find a particular string or pattern within a file, but you don’t know where to start looking, that is where grep is extremely useful.

* Syntax: grep [option(s)] pattern [file(s)]
* Common options: -i, -c, -n

**man — Print manual or get help for a command**

The man command is your manual and is very useful when you need to figure out what a command does. For example, if you didn’t know what the command rmdir does, you could use the man command to find that out.

* Syntax: man [option(s)] keyword(s)
* Common options: -w, -f, -b

**pwd — Print working directory**

pwd is used to print the current directory you’re in. As an example, if you have multiple terminals going and you need to remember the exact directory you’re working within, then pwd will tell you.

* Syntax: pwd [option(s)]
* Common options: options aren’t typically used with pwd

**cd — Change directory**

cd will change the directory you’re in so that you can get info, manipulate, read, etc. the different files and directories in your system.

* Syntax: cd [option(s)] directory
* Common options: options aren’t typically used with cd

**mv — Move or rename directory**

mv is used to move or rename directories. Without this command, you would have to individually rename each file which is tedious. mv allows you to do batch file renaming which can save you loads of time.

* Syntax: mv [option(s)] argument(s)
* Common options: -i, -b

**rmdir — Remove directory**

dir will remove empty directories. This can help clean up space on your computer and keep files and folders organized. It’s important to note that there are two ways to remove directories: rm and rmdir. The distinction between the two is that rmdir will only delete empty directories, whereas rm will remove directories and files regardless if they contain data or not.

* Syntax: rmdir [option(s)] directory\_names
* Common options: -p

**locate — Locate a specific file or directory**

This is by far the simplest way to find a file or directory. You can keep your search broad if you don’t know what exactly it is you’re looking for, or you can narrow the scope by using wildcards or regular expressions.

* Syntax: locate [option(s)] file\_name(s)
* Common options: -q, -n, -i

***Enjoying the article?****Scroll down to*[*sign up*](https://www.educative.io/blog/blog-newsletter-annoucement)*for our free, bi-monthly newsletter.*

**less — view the contents of a text file**

The less command allows you to view files without opening an editor. It’s faster to use, and there’s no chance of you inadvertently modifying the file.

* Syntax: less file\_name
* Common options: -e, -f, -n

**compgen — Shows all available commands, aliases, and functions**

compgen is a handy command when you need to reference all available commands, aliases, and functions.

* Syntax: compgen [option(s)]
* Common options: -a, -c, -d

**> — redirect stdout**

The > character is the redirect operator. This takes the output from the preceding command that you’d normally see in the terminal and sends it to a file that you give it. As an example, take echo “contents of file1” > file1. Here it creates a file called file1 and puts the echoed string into it.

* Syntax: >
* Common options: n/a

**cat — Read a file, create a file, and concatenate files**

cat is one of the more versatile commands and serves three main functions: displaying them, combining copies of them, and creating new ones.

* Syntax: cat [option(s)] [file\_name(s)] [-] [file\_name(s)]
* Common options: -n

**| — Pipe**

A pipe takes the standard output of one command and passes it as the input to another.

* Syntax: |
* Common options: n/a

**head — Read the start of a file**

By default, the head command displays the first 10 lines of a file. There are times when you may need to quickly look at a few lines in a file and head allows you to do that. A typical example of when you’d want to use head is when you need to analyze logs or text files that change frequently.

* Syntax: head [option(s)] file(s)
* Common options: -n

**tail — Read the end of a file**

By default, the tail command displays the last 10 lines of a file. There are times when you may need to quickly look at a few lines in a file and tail allows you to do that. A typical example of when you’d want to use tail is when you need to analyze logs or text files that change frequently.

* Syntax: tail [option(s)] file\_names
* Common options: -n

**chmod — Sets the file permissions flag on a file or folder**

There are situations that you’ll come across where you or a colleague will try to upload a file or modify a document and you receive an error because you don’t have access. The quick fix for this is to use chmod. Permissions can be set with either alphanumeric characters (u, g, o) and can be assigned their access with w, r, x. Conversely, you can also use octal numbers (0-7) to change the permissions. For example, chmod 777 my\_file will give access to everyone.

* Syntax: chmod [option(s)] permissions file\_name
* Common options: -f, -v

**exit — Exit out of a home directory**

The exit command will close a terminal window, end the execution of a shell script, or log you out of an SSH remote access session.

* Syntax: exit
* Common options: n/a

Master the Bash Shell

Join the millions of developers getting hands-on experience with over 600 courses.



**Master the Bash Shell**

Beginner

Certificate included

[Preview](https://www.educative.io/courses/master-the-bash-shell)

**history — list your most recent commands**

An important command when you need to quickly identify past commands that you’ve used.

* Syntax: history
* Common options: -c, -d

**clear — Clear your terminal window**

This command is used to clear all previous commands and output from consoles and terminal windows. This keeps your terminal clean and removes the clutter so you can focus on subsequent commands and their output.

* Syntax: clear
* Common options: n/a

**cp — copy files and directories**

Use this command when you need to back up your files.

* Syntax: cp [option(s)] current\_name new\_name
* Common options: -r, -i, -b

**kill — terminate stalled processes**

The kill command allows you to terminate a process from the command line. You do this by providing the process ID (PID) of the process to kill. To find the PID, you can use the ps command accompanied by options -aux.

* Syntax: kill [signal or option(s)] PID(s)
* Common options: -p

**sleep — delay a process for a specified amount of time**

sleep is a common command for controlling jobs and is mainly used in shell scripts. You’ll notice in the syntax that there is a suffix; the suffix is used to specify the unit of time whether it be s (seconds), m (minutes), or d (days). The default unit of time is seconds unless specified.

* Syntax: sleep number [suffix]
* Common options: n/a