16 Must-Know Bash Commands for Data Scientists

Exploring some of the most commonly used bash commands

Giorgos Myrianthous

Photo by Brian McGowan on Unsplash

It is very important for Data Scientists to have a basic

understanding around bash and its commands. Often referred to as the terminal, console or command line, Bash is a *Unix shell* that can help you navigate within your machine and perform certain tasks.

In today's article, we are going to explore a few of the most commonly used bash commands that every Data Scientist must know.

ls

The `ls` (list) command is used to list directories or files. By default (i.e. running `ls` with no options at all) the command will return the directories and files of the current directory, excluding any hidden files. Some of the most useful options are:

- `ls -a`: List all the files in the current directory including hidden files too
- `ls -l`: Long listing of all the files and their size in the current directory

Syntax

```
ls [OPTIONS] [FILES]
```

Example

```
$ ls -la
drwx------@  31 user  grp    992 15  May 15:56 .
drwxr-xr-x+ 131 user  grp    4192 16 May 17:33 ..
-rw-r--r--@   1 user  grp     200  9 May 18:39 .gitignore
-rw-r--r--@   1 user  grp     200  9 May 18:39 Backup/
drwxr-xr-x    3 user  grp      96 17 Feb  2020 my-script.sh
```

Long list of all directories and files (including hidden) of the current directory

```
$ ls -la
```

cd

The `cd` (change directory) command is used to navigate in the directory tree structure.

Syntax

```
cd [OPTIONS] directory
```

The command can take only two options`-L` to specify if symbolic links should be followed or `P` to specify that they shouldn't.

Example

Change current directory

```
$ cd myproject
```

rm

`rm` (remove) command is used to delete files, directories or even symbolic links from your file system. Some of the most useful options are:

- `rm -i`: Remove all the files in the directory but let user confirm before deleting it
- `rm -r`: Remove non-empty directories including all the files within them
- `rm -f`: Remove files or directories without prompting even if they are write-protected — f stands for force.

Syntax

```
rm [OPTIONS]... FILE...
```

Example

Force deletion of the directory with name ‘directoryName’

```
$ rm -rf directoryName
```

mv

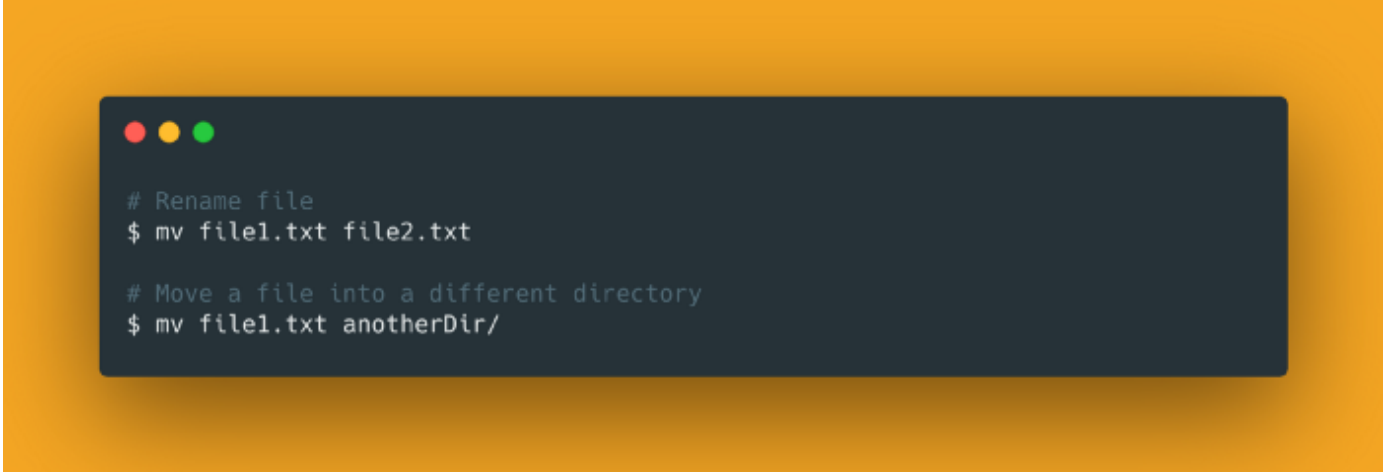
`mv` (move) command is used to move one or more directories or files from one location in the file system to another.

Syntax

```
mv [OPTIONS] SOURCE DESTINATION
```

- `SOURCE` can be one ore more directories or files
- `DESTINATION` can be a file (used for renaming files) or a directory (used for moving files and directories into other directories.

Example




```
# Rename file
$ mv file1.txt file2.txt# Move a file into a differ
$ mv file1.txt anotherDir/
```

cp

`cp` is a utility that lets you copy files or directories within the file system. Some of the most useful options are:

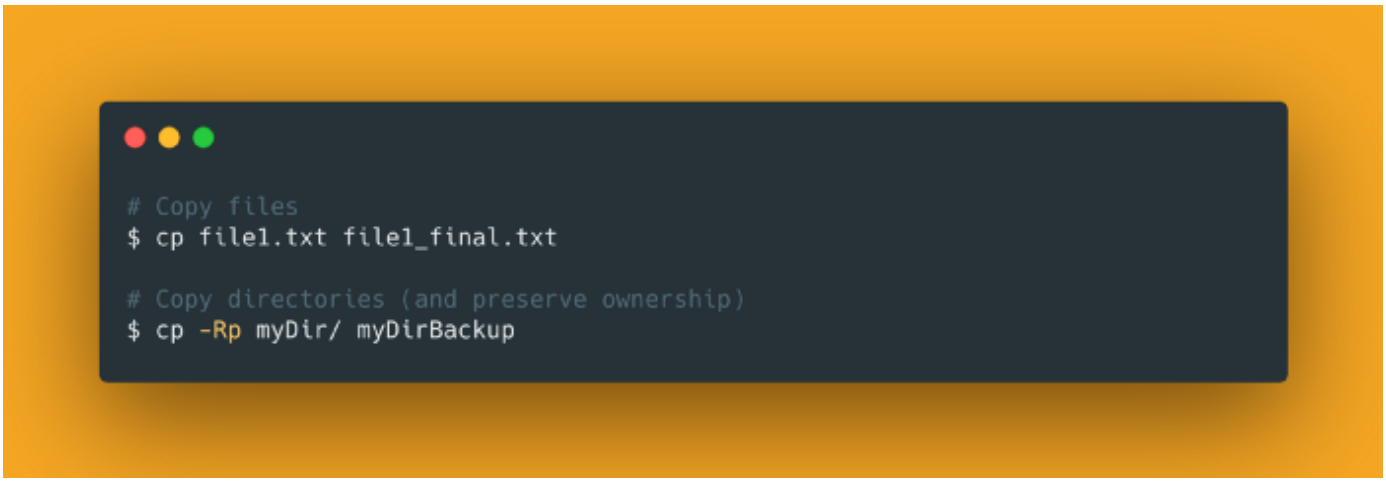
- `cp -u file1.txt file1_final.txt`: Copy the content of `file1.txt` into `file1_final.txt` only if the former (source) is newer than the latter (destination)
- `cp -R myDir/ myDir_BACKUP`: Copy directories
- `cp -p file1.txt file1_final.txt`: Copy `file1.txt` and preserve ownership

Syntax

```
cp [OPTIONS] SOURCE... DESTINATION
```

- `SOURCE` may contain one or more directories or files
- `DESTINATION` must be a single directory or file

Example




```
# Copy files
$ cp file1.txt file1_final.txt# Copy directories (a
$ cp -Rp myDir/ myDirBackup
```

mkdir

The `mkdir` command is useful when it comes to creating new directories in the file system.

Syntax

```
mkdir [OPTION] [DIRECTORY]
```

- `DIRECTORY` can be one or more directories

Example

Creating a new directory

```
# Create new directory with name myNewDir
$ mkdir myNewDir
```

pwd

The `pwd` (print working directory) command can be used to report the absolute path of the current working directory.

Example

Reporting the path to the current working directory

```
$ pwd
```

```
/Users/administrator
```

touch

The `touch` command allows you to create new empty files or update the timestamp on existing files or directories. If you use `touch` with files that already exist, then the command will just update their timestamps. If the files do not exist then this command will simply create them.

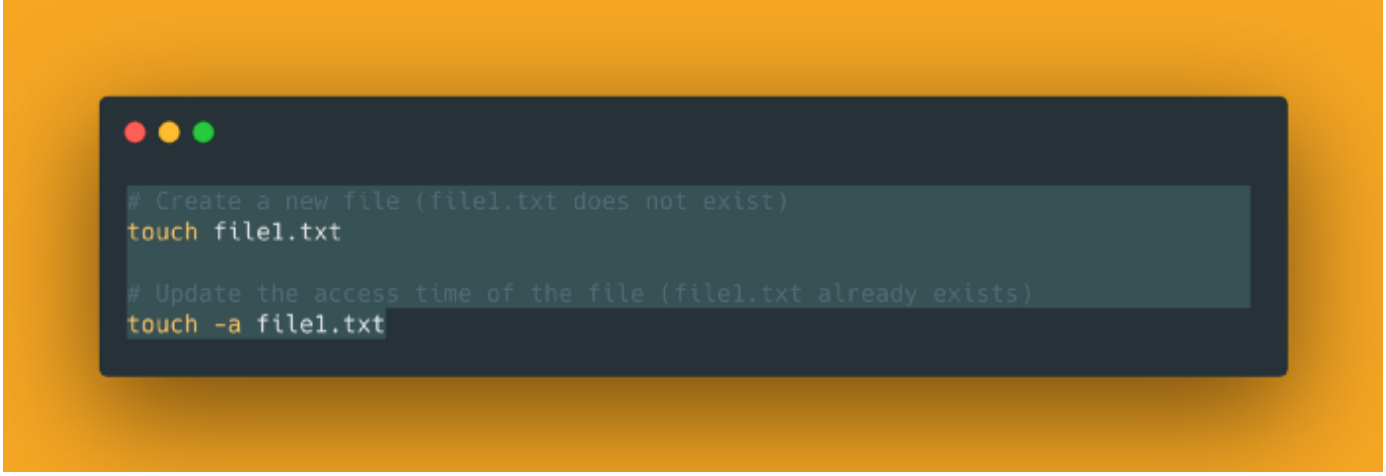
Some of the most useful options are:

- `touch -c file1.txt`: If file `file1.txt` already exists then this command will update the file's timestamps otherwise it will do nothing.
- `touch -a file1.txt`: Update only the access timestamp of the file
- `touch -m file1.txt`: Update only the modify time of the file

Syntax

```
touch [OPTIONS] [FILES]
```

Example




```
# Create a new file (file1.txt does not exist)
touch file1.txt# Update the access time of the file
touch -a file1.txt
```

cat

`cat` is a very commonly used command that allows users to read concatenate or write file contents to the standard output.

Some of the most useful options are:

- `cat -n file1.txt`: Display the contents of the file `file1.txt` along with line numbers.
- `cat -T file1.txt`: Display the contents of the file `file1.txt` and distinguish tabs and spaces (tabs will be displayed as `^I` in the output)

Syntax

```
cat [OPTIONS] [FILE_NAMES]
```

- `FILE_NAMES` can be none or more file names

Example


```
# Display the content of file $HOME/.pip/pip.conf
cat $HOME/.pip/pip.conf# Append the content of file
cat file1.txt >> file2.txt
```

less

The `less` command lets you display the contents of a file one page at a time. `less` won't read the entire file when it is being called and thus it leads to way faster load times.

Some of the most useful options are:

- `less -N file1.txt`: Display the content (first page) of the file `file1.txt` and show line numbers.

- `less -X file1.txt`: By default when you exit `less` the content of the file will be cleared from the command line. If you want to exit but also keep the content of the file on the screen use the `-X` option.

Syntax

```
less [OPTIONS] filename
```

Example


```
# Display the content of file $HOME/.pip/pip.conf
less $HOME/.pip/pip.conf
```

more

`more` command can also be used for displaying the content of a file in the command line. In contrast to `less`, `more` command loads the entire file at once and this is why `less` seems to be faster.

Some of the most useful options are:

- `more -p file1.txt`: Clear the command line screen and then display the content of `file1.txt`
- `more +100 file1.txt`: Display the content of `file1.txt` starting from the 100th line onwards.

Syntax

```
more [OPTION] filename
```

Example


```
# Display the content of file $HOME/.pip/pip.conf
more $HOME/.pip/pip.conf
```

---

grep

The `grep` (global regular expression) command is useful when you wish to search for a particular string in files.

Some of the most useful options are:

- `grep -v Andrew employees.txt`: Invert match `Andrew` in `employees.txt`. In other words, display all the lines that do not match the pattern `Andrew`
- `grep -r Andrew dirName/`: Recursuvely search for pattern `Andrew` in all files in the specified directory `dirName`
- `grep -i ANdrEW employees.txt`: Perform a **case insensitive** search

Syntax

```
grep [OPTIONS] PATTERN [FILE...]
```

- `PATTERN` is the search pattern
- `FILE` can be non to more input file names

Example

Search for export command in the user profile

```
# Search for `export` (case insensitive) in user pr
```

```
$ grep -i export ~/.bash_profile
```

curl

The `curl` command is used to download or upload data using protocols such as FTP, SFTP, HTTP and HTTPS.

Syntax

```
curl [OPTIONS] [URL...]
```

Example


```
$ curl -L google.com
```

which

`which` command is used to identify and report the location of the provided executable. For instance, you may wish to see the location of the executable when calling `python3`.

Syntax

```
which [OPTIONS] FILE_NAME
```

Example


```
$ which python3
/usr/local/bin/python3
```

---

top

`top` command can help you monitor running processes and the resources (such as memory) they are currently using.

Some of the most useful options are:

- `top -u myuser`: Display processes for the user `myuser`

Example

```
Processes: 646 total, 2 running, 644 sleeping, 3147 threads
 14:39:08
Load Avg: 2.96, 2.81, 2.64  CPU usage: 5.84% user, 4.80% sys, 89.35% idle
SharedLibs: 413M resident, 52M data, 128M linkedit.
MemRegions: 255238 total, 4419M resident, 185M private, 2317M shared.
PhysMem: 15G used (2867M wired), 1089M unused.
VM: 4398G vsize, 2305M framework vsize, 11232614(0) swapins, 12517865(0)
swapout
Networks: packets: 56274778/59G in, 34808267/15G out.
Disks: 15371047/347G read, 25513670/310G written.

PID    COMMAND      %CPU TIME     #TH   #WQ  #PORTS MEM    PURG   CMPRS  PGRP
17929  Mail         28.7 01:46:32 14    7    979    124M   3264K  46M    17929
56923  top          16.3 00:01.56 1/1   0    25     6296K+ 0B     0B     56923
147    WindowServer 6.1  15:41:01 14    6    3097   1024M  59M    127M   147
31462  Terminal     5.1  00:36.64 8     2    332    44M+   40K    6620K  31462
0      kernel_task  5.1  13:55:04 192/8 0    0      571M   0B     0B     0
39913  com.docker.h 5.1  09:49:34 13    0    36     5542M  0B     1392M  39857
50487  Google Chrom 1.9  07:22.99 19    1    177    193M   0B     15M    101
```

Output of `top` command

history

`history` command displays the history of the commands that you've recently run.

Some of the most useful options are:

- `history -5`: Display the last 5 commands
- `history -c`: Clear the history list
- `history -d 10 20`: Delete lines 10–20 from history list

Example

Get the recent commands from history that include `python3` keyword

```
$  history | grep python3
```

Conclusion

In this article, we explored only a small subset of some of the most commonly used bash commands. It is very important for Data Scientists to be able to use the command line as this will definitely help them perform basic tasks easily and most importantly efficiently.

Although it’s not mandatory for Data Scientists to become gurus of bash, it’s a very important skill that you may want to consider mastering. At the end of the day, bash is fun :)