A Primer on Command Line

4 BASH in LINUX/UNIX

Richard Van

The Go-To Guide for the Uninitiated

Yihan Shao Research Group (CC-ATS)
Department of Chemistry & Biochemistry
University of Oklahoma
USA
September 27th, 2022

Contents

1 Getting Started

It is very likely you are reading this document because you have little to no experience using Linux and are about to access Schooner for the first time. Linux is an essential skill to learn especially in the field of high-performance computing. The following primer will give you an idea of some of the most fundamental bash commands required to use Schooner.

2 Directories & Paths

2.1 pwd - print working directory

Pathnames help you figure out where you are in relation to the whole file system. Your current working directory (where you are) can be found with **pwd**. This command is useful when you need the full pathname to your current directory for copying/moving/downloading files and folders! Or sometimes you just forget where you are.

```
$ pwd
/home/<username>
```

2.2 ls - list

The **ls** command will list the contents of your working directory. If you are new on the supercomputer, you will not have any files/folders to list. If you're on your personal computer, you might see the following output (or something similar). Recall that directories are often blue unless modified.

```
$ ls
Desktop/ Downloads/
```

You can modify **ls** with *flags* to get more information about your contents. *Flags* are set by including the dash symbol (**-**) followed by a sort letter after the command.

For **ls**, some flags we use are:

```
$ ls -h # Human readable filesize (with -l returns 1K, 2G, etc.)
$ ls -l # Use a long listing format
$ ls -t # Sort by time, newest first
```

You can also combine flags to one dash (**-**), giving you the same output:

```
$ ls -hlt # Readable filesize, all listed, and sorted by time
```

2.3 mkdir - make directory

The **mkdir** command allows you to make a new directory in the current directory if no path is specified.

```
$ mkdir amino_acids
```

To see the directory you just created, type **ls**:

```
$ ls
Desktop/ Downloads/ amino_acids/
```

Subdirectories can be made if a path is specified, by adding the **-p** flag to **mkdir**. Multiple directories can be also be made if followed by a space (Check whether the directory is made by **ls**!!):

```
$ mkdir -p amino_acids/ala amino_acids/asn
$ ls
Desktop/ Downloads/ amino_acids/
$ ls amino_acids
ala asn
```

2.4 cd - changing directory

To change your current working directory and go in to another directy, you want the **cd** command. This command requires 1 argument which specifies the directory you want to change to. For example:

```
$ cd [path]
```

So if we wanted to **cd** into the **ala/** folder, we can type (Check whether you're in the directory you want with **pwd**!!):

```
$ cd amino_acids/ala
$ pwd
/home/<username>/amino_acids/ala
```

To go back a folder, follow the **cd** command by two periods (**..**). To go back 2 directories, just repeat the 2 periods (**..**) followed by a backslash (\). If you want to go all the way back to **/home/username** , just type **cd**. (Check with **pwd**!!)

```
$ cd ..
$ pwd
/home/<username>/amino_acids
```

or

```
$ cd ../../
$ pwd
/home/<username>
```

or

```
$ cd
$ pwd
/home/<username>
```

3 Managing Files

3.1 cp - copy

This command takes 2 arguments, for example:

```
$ cp [source1] [source2]
```

Where [**source1**] is the original file you want to copy, and [**source2**] is the new copied file. Note that [**source2**] can be a new file name or a pathname. We can copy an example script from my directory to your **/home/username** directory, if you **ls**, you should see the file:

```
$ cp /home/van/Scripts/bash_tutorial/example.txt /home/<username>
$ ls
Desktop Downloads amino_acids example.txt
```

The **cp** command can also copy directories if the flag **-r** is specified:

```
$ cp -r amino_acids aa_residues
$ ls
Desktop Downloads amino_acids aa_residues example.txt
```

A shorthand notation we often use is the single period symbol (**.**), which means the current working directory (or here).

```
$ cp /home/van/Scripts/bash_tutorial/aa.txt .
$ ls
Desktop Downloads amino_acids aa_residues example.txt aa.txt
```

3.2 mv - move

This command takes 2 arguments, for example:

```
$ mv [source] [destination]
```

Where [**source**] is the file that is to be moved and [**destination**] is the location where the file will be moved to. It should be noted that you do not have to be in the same directory as [**source**] or [**destination**] to use the command the only requirement is the usage of the correct pathnames.

3.3 mv - rename

```
$ mv [original] [desired name]
```

Where [**original**] is the file that is to be renamed and [**desired name**] is the new name. Make sure that [**desired name**] is not the same as a directory or [**original**] will be moved to a different directory instead of being renamed.

3.4 rm - remove

The most powerful of all commands, which is to be used if there are old files that are not necessary anymore or if there are empty files in a directory. The command only takes one argument. An example would be:

```
$ rm [target]
```

One key flag to be aware of is the -r flag which allows for the removal of directories. Whenever using the rm command, it would be wise to ensure that the wildcard is not improperly used as the combination of the rm command and improper usage of the wildcard could lead to the deletion of all important files/work. Also important to note is that the usage of the -r flag can delete the actual file a symbolic link is pointing to and not just the link itself.

4 Display Files on Screen

4.1 clear - clear screen

After spending some time on Schooner, your terminal window might get very full with previously run commands. The solution to that problem is the clear command. Utilization of the command is as follows:

```
$ clear
```

4.2 cat - concatenate

At times, you will likely want to view the contents of a file, but entering the text editor can be a hassle. To answer that problem, there is the cat command which can be used to print the contents of a file to your terminal window.

4.3 less

4.4 head

4.5 tail