Command Line Basics

Hanan Salim

A command line or a terminal is a way to interact with a computer via text.This may seem archaic but knowing how to use the command line can make doing your job much easier compared to the GUI alternative. Furthermore, learning a handful of commands can take you quite far. If we open up a command line prompt, we get something like this:

The first line tells us the user(hisal in my case) and the computer(LAPTOP-NFOMD8T5) you are using. The ∼ indicates that you are in the home or root directory. Right below that is a dollar sign, which is where we will enter our commands.

The first command we will introduce helps us answer the question "where am I?". If you type `pwd` and hit enter, you will get the path for your current location, which is like an address for a file or folder. There are two ways two represent this path. The first way is called an absolute path, which is the address of your file or folder in relation to the root directory. The path you see when you type `pwd` is the absolute path. Another way to write a path is relative to where you are at that moment. This is called the relative path and we will see more of it later.

One of the things that I struggled with early on when learning to use a command line was figuring out where I was. So, if you are ever lost or confused about where you are just type in `pwd`.

Typically, if you want to find some file on your computer you would open up your file manager and then click the folder or a sequence of folders until you reach said file. To do the same thing on a command line you would type the command `cd` followed by the directory you want to move. For example, in my home directory I have a folder called 'my_folder', which I can move into using the `cd` command.

```
hisal@LAPTOP-NF0MD8T5 ~
$ cd my_folder/

hisal@LAPTOP-NF0MD8T5 ~/my_folder
$
```

Notice that we did not type in the absolute path (/home/hisal/my_folder). Since hisal is the working directory and it contains the folder we want to move into, we can just type the relative path, i.e ',my_folder'.

We can see what is inside this folder by typing `ls`, which will list all of our files and sub-folders. As we can see in the image below, this folder contains a single file called 'file.txt'.

```
hisal@LAPTOP-NF0MD8T5 ~/my_folder
$ ls
file.txt
```

We can view the content on this file in several ways. The first way is through the `cat` command which will print out the entire file. Type in `cat` followed by the name of the file you would like to view.

```
hisal@LAPTOP-NF0MD8T5 ~/my_folder
$ cat file.txt
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
5.4,3.7,1.5,0.2,Iris-setosa
4.8,3.4,1.6,0.2,Iris-setosa
4.8,3.0,1.4,0.1,Iris-setosa
4.3,3.0,1.1,0.1,Iris-setosa
5.8,4.0,1.2,0.2,Iris-setosa
5.7,4.4,1.5,0.4,Iris-setosa
5.4,3.9,1.3,0.4,Iris-setosa
5.1,3.5,1.4,0.3,Iris-setosa
5.7,3.8,1.7,0.3,Iris-setosa
5.1,3.8,1.5,0.3,Iris-setosa
5.4,3.4,1.7,0.2,Iris-setosa
5.1,3.7,1.5,0.4,Iris-setosa
```

If the file is short, this is okay but often files are very large and we do not need to view them in their entirety. In this case, we can view the top portion of the file using the `head` command and the bottom portion of the file using the `tail` command.

```
hisal@LAPTOP-NF0MD8T5 ~/my_folder
$ head file.txt
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa

hisal@LAPTOP-NF0MD8T5 ~/my_folder
$ tail file.txt
6.9,3.1,5.1,2.3,Iris-virginica
5.8,2.7,5.1,1.9,Iris-virginica
6.8,3.2,5.9,2.3,Iris-virginica
6.7,3.3,5.7,2.5,Iris-virginica
6.7,3.0,5.2,2.3,Iris-virginica
6.3,2.5,5.0,1.9,Iris-virginica
6.5,3.0,5.2,2.0,Iris-virginica
6.2,3.4,5.4,2.3,Iris-virginica
5.9,3.0,5.1,1.8,Iris-virginica
```

Commands in unix tend to have options that allow a user to change or extend the behavior of the command. The head/tail command comes with an option -n, which allows a user to specify how many lines should be shown. You can use this line by typing `tail -n3`, where 3 is the number of lines we want to see.

```
hisal@LAPTOP-NF0MD8T5 ~/my_folder
$ head -n3 file.txt
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
```

Although we may not be interested in viewing the entire file, we may still want to know how many lines, word, or characters are in the file. We can find out by using `wc` followed by the file name. In the image below, we can see that file.txt has 151 lines, 150 words and 4551 characters. Typically the most useful information is the number of lines. We can find out directly by using `wc -l`.

```
hisal@LAPTOP-NF0MD8T5 ~/my_folder
$ wc file.txt
 151  150 4551 file.txt
```

Lets say we want to search for a pattern in a file. In the case of this file, we want to see all lines that contain the number 5. We can accomplish this by using `grep`. Type in `grep '5' file.txt`, where 5 is the pattern we want to search for and file.txt is the file we want to search. A useful extension is `-w`, which allows a user to search for an exact match. Furthermore, if we want the inverse of our search, we can use `grep -v [file name]`.

```
hisal@LAPTOP-NF0MD8T5 ~/my_folder
$ grep '5' file.txt
5.1,3.5,1.4,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
5.4,3.7,1.5,0.2,Iris-setosa
5.8,4.0,1.2,0.2,Iris-setosa
```

From prior knowledge, I know that this file contains three different types of Iris (Iris-setosa, Iris-versicolor, and Iris-virginica) but I am only interested in setosa. In fact, I want to know how many lines contain this flower. We can find out by combining our `grep` and `wc -l` using the pipe command. We will use the `grep` like usual and then type `|` followed by `wc -l`. The symbol `|` is the pipe command, it takes the output of the first command and pipes it as an input for the second command. We can use the pipe command to string together as many commands as we like.

```
hisal@LAPTOP-NF0MD8T5 ~/my_folder
$ grep 'setosa' file.txt | wc -l
50
```

Typical file formats are separate columns by a delimiter. If we are only interested in certain columns we can use the `cut` command. There are several important things we must tell the command. First, what is the symbol that separates our columns? This may be a space, a tab, or a comma. We can specify the delimiter by using `-d` followed by delimiter. We also need to say what column(s) we are interested in. We can specify this by using `-f` followed by the column(s) of interest. Several examples of its use are given below.

```
hisal@LAPTOP-NF0MD8T5 ~/my_folder
$ cut -d, -f5 file.txt | head -n3
Iris-setosa
Iris-setosa
Iris-setosa

hisal@LAPTOP-NF0MD8T5 ~/my_folder
$ cut -d, -f1,3 file.txt | head -n3
5.1,1.4
4.9,1.4
4.7,1.3

hisal@LAPTOP-NF0MD8T5 ~/my_folder
$ cut -d, -f1,3-4 file.txt | head -n3
5.1,1.4,0.2
4.9,1.4,0.2
4.7,1.3,0.2
```

We can save the output of our commands using the carrot symbol, `>` followed by name of the output file.

```
hisal@LAPTOP-NF0MD8T5 ~/my_folder
$ cut -d, -f1,5 file.txt > output.txt
```

We have learned some useful commands but we still do not know how to do some very basic tasks like creating new folders and moving files around. Creating folders is done using the `mkdir` command followed by the folder name.

```
hisal@LAPTOP-NF0MD8T5 ~/my_folder
$ mkdir data

hisal@LAPTOP-NF0MD8T5 ~/my_folder
$ ls
data  file.txt
```

Since this folder is empty, lets populate it with a file. Lets try to move the file output.txt into the folder we just created. We have two options, copy or move. Lets beginning with copying. This can be done using `cp` followed by the file we want to copy and then the path of the directory we want to copy this file into to. This path can be the absolute path or relative.

```
hisal@LAPTOP-NF0MD8T5 ~/my_folder
$ cp output.txt data/

hisal@LAPTOP-NF0MD8T5 ~/my_folder
$ ls data/
output.txt
```

We can move the file using the `mv` command. The structure of this command is similar to the copy command but the behavior is different. The move command will move the file into a directory but delete the original copy, where as copy will keep it. We can also use `mv` to rename files and folders. If we change directories and go into the data folder we created, we can change the name of output.txt to new.txt.

```
hisal@LAPTOP-NF0MD8T5 ~/my_folder/data
$ mv output.txt new.txt

hisal@LAPTOP-NF0MD8T5 ~/my_folder/data
$ ls
new.txt
```

Since we have two files with the same content and different names, lets remove one of them using `rm`. It is important to know that once you remove a file, there is no going back. So be careful when deleting files.

```
hisal@LAPTOP-NF0MD8T5 ~/my_folder/data
$ rm new.txt
```

We will end where we started by talking about changing directories. In particular how do we go back? We can specify the absolute path or we can use `..`, which takes us up one level in the hierarchy.

```
hisal@LAPTOP-NF0MD8T5 ~/my_folder/data
$ pwd
/home/hisal/my_folder/data

hisal@LAPTOP-NF0MD8T5 ~/my_folder/data
$ cd ..

hisal@LAPTOP-NF0MD8T5 ~/my_folder
$ pwd
/home/hisal/my_folder
```

Notice that when we typed `pwd` in the first line, we are in the data folder but using `cd ..` takes us back to my_folder. If we had wanted we could have used the absolute path, `cd /hom/hisal/my_folder`. Lastly, if we want to go back to the home directory, we can use $\sim$ symbol instead of the path.

```
hisal@LAPTOP-NF0MD8T5 ~/my_folder
$ cd ~

hisal@LAPTOP-NF0MD8T5 ~
$
```