Command Line
Software Engineering for Scientists

The command line is a text interface to your computer or remote server. Commands (programs) are executed by typing out the program name and following it by options. Unix or Linux based systems are the most prevalent types of computer systems on which computationally intensive science is done. The best way of interacting with these systems is through the command line. This document will cover the basics.

Structure of a command

A command is typed into a prompt (ends with a '$'). The command consists of a program to run and a list of parameters. Once you've typed out the full command you hit return to execute it. The program takes those parameters as input and produces some output. In this way the program is a "black box" in that the user only cares about its inputs and outputs.

For example, let's say we have a python script called `compute.py` which performs basic mathematical operations. It is run as follows:

```
$ python compute.py add 4 7
11
```

In this command, the program is `python` and the options are `compute.py`, `add`, `4`, and `7`. In this case the python program interprets the code in `compute.py` and passes to it the three following parameters (`add, 4, 7`). This command returns `11`, the addition of 4 and 7.

File system navigation

`pwd`	Print the "present working directory" --- the full path to your location in the file system.	`$ pwd`
		`/home/jovyan/swefs`
		`$`

`ls`	List the contents of a given directory (current directory if no path is specified). Options -l lists detailed info of files. -a also lists hidden files.	`$ ls`
		`file1.txt  file2.txt  file3.txt  README.md`
		
		`$ ls -l`
		`total 16`
		`-rw-r--r-- 1 jovyan users 59 Aug 11 21:51 file1.txt`
		`-rw-r--r-- 1 jovyan users 84 Aug 11 18:45 file2.txt`
		`-rw-r--r-- 1 jovyan users 54 Aug 12 00:12 file3.txt`
		`-rw-r--r-- 1 jovyan users 12 Aug  9 21:57 README.md`
		`$`

`cd`	Change directory to specified location ("..." = up one directory, "." = present directory	`$ pwd`
		`/home/jovyan/swefs`
		`$ cd ..`
		`$ pwd`
		`/home/jovyan`
		`$ cd ./`

		`$ pwd`
		`/home/jovyan/swefs`
		`$`

`mkdir`	Make a new directory with a specified name.	`$ mkdir ./data`
		`$ ls`
		`data`
		`$`

`rmdir`	Remove directory	`$ rmdir ./data`
		`$ ls`
		
		`$`

File manipulation

`cat`	Print the full contents of a file or files to stdout.	`$ cat counties_data.csv`
		`County,State,Population,Area_sq_miles`
		`Boulder,CO,106392,740.0`
		`Broomfield,CO,70465,33.55`
		`Gilpin,CO,6243,150.0`
		`Grand,CO,15734,1870.0`
		`Jefferson,CO,582881,774.0`
		`Larimer,CO,356899,2634.0`
		`Weld,CO,324492,4017.0`
		`$`

`head`	Prints the first few lines of a file to stdout	`$ head counties_data.csv`
		`County,State,Population,Area_sq_miles`
		`Boulder,CO,106392,740.0`
		`Broomfield,CO,70465,33.55`
		`$`

`tail`	Print the last few lines of a file to stdout	`$ tail counties_data.csv`
		`Jefferson,CO,582881,774.0`
		`Larimer,CO,356899,2634.0`
		`Weld,CO,324492,4017.0`
		`$`

`less`	View the contents of a file within the terminal window. Can navigate through the contents with “j” and “k”. Also allows for searching of the file. Different from “cat” because it is interactive and is not sent to stdout.	`$ less counties_data.csv`
		`County,State,Population,Area_sq_miles`
		`Boulder,CO,106392,740.0`
		`Broomfield,CO,70465,33.55`
		`Gilpin,CO,6243,150.0`
		`Grand,CO,15734,1870.0`
		`Jefferson,CO,582881,774.0`
		`Larimer,CO,356899,2634.0`
		`Weld,CO,324492,4017.0`

`cp`	Copy a file to a new location	`$ cp data.txt ./newdir/`
		`$ ls`
		`data.txt`
		`$ ls ./newdir/`
		`data.txt`

		`$`

`mv`	Move a file to a new location	`$ cp data.txt ./newdir/`
		`$ ls`
		
		`$ ls ./newdir/`
		`data.txt`
		`$`

`rm`	Remove a file	`$ ls ./newdir/`
		`Data.txt`
		`$ rm ./newdir/data.txt`
		`$ ls ./newdir/`
		
		`$`

Other useful commands

`grep`	Search for all lines containing a string	`$ grep Boulder counties_data.csv` `Boulder,CO,106392,740.0`

`cut`	Extract particular fields separated by delimiter	`$ cut -d ",” -f 1,3 counties_data.csv`
		`County,Population`
		`Boulder,106392`
		`Broomfield,70465`
		`Gilpin,6243`
		`Grand,15734`
		`Jefferson,582881`
		`Larimer,356899`
		`Weld,324492`
		`$`

`sort`	Order the input by a set of fields. In this example we are doing a numeric sort (-n) on fourth field (-k 4) using a comma delimiter (-t “,”)	`$ sort -n -k 4 -t ",” counties_data.csv`
		`County,State,Population,Area_sq_miles`
		`Broomfield,CO,70465,33.55`
		`Gilpin,CO,6243,150.0`
		`Boulder,CO,106392,740.0`
		`Jefferson,CO,582881,774.0`
		`Grand,CO,15734,1870.0`
		`Larimer,CO,356899,2634.0`
		`Weld,CO,324492,4017.0`
		`$`

`wc`	Count the number of lines, words, and/or characters in the input.	`$ wc counties_data.csv` `  8   8 204 counties_data.csv`

Help and History

In order to help you navigate the command line utilities, most commands have a '`-h`' and/or '`--help`' option which will print info about the options available for the command and how to use it. Typically, the '`--help`' option produces a more detailed print.

```
$ pwd --help
pwd: pwd [-LP]
    Print the name of the current working directory.

    Options:
      -L        print the value of $PWD if it names the current working
                directory
      -P        print the physical directory, without any symbolic links

    By default, `pwd' behaves as if `-L' were specified.

    Exit Status:
    Returns 0 unless an invalid option is given or the current directory
    cannot be read.
$
```

Additionally, an entire history of all the commands you have run at the terminal are kept and can be accessed with the `history` command. This history can then be searched with things like `grep`.

```
$ history
…
  506  grep -h
  507  pwd --help
  508  history
$
```

I/O redirection

The command-line environment has three named "streams" of data: standard input (**STDIN**), standard output (**STDOUT**), and standard error (**STDERR**).

```
$ grep Boulder counties_data.csv
Boulder,CO,106392,740.0                    Standard output
$ grepp Boulder counties_data.csv
bash: grepp: command not found             Standard error
```

Standard input streams data to a program, which could be typed in by a user or another file (does not include command line inputs). When a program prints a value (i.e. print(a+b)), that information will be included in the standard output stream and typically end up displayed on the screen. Similarly, standard error carries error messages, which also typically end up displayed on the screen.

However, both these streams can be redirected to destinations other than the screen. Namely, they can be directed to files (so the results can be saved) or to the standard input streams of other programs (so various commands/programs can be chained together).

Redirecting to a file uses the '>' character, in one of two ways:

>	Overwrite a file with standard output	`$ pwd`
		`/home/jovyan/`
		`$ pwd > directories.txt`
		`$ cat directories.txt`
		`/home/jovyan/`
		`$`

>>	Append to a file with standard output	`$ cd ./data/`
		`$ pwd`
		`/home/jovyan/data/`
		`$ pwd >> directories.txt`
		`$ cat directories.txt`
		`/home/jovyan/`
		`/home/jovyan/data/`
		`$`

2>	Overwrite a file with standard error	`$ pwdd 2> error_log.txt`
		`$ cat error_log.txt`
		`bash: pwdd: command not found`
		`$`

2>>	Append to a file with standard error	`$ ls -e 2>> error_log.txt`
		`$ cat error_log.txt`
		`bash: pwdd: command not found`
		`ls: invalid option -- 'e'`
		`Try 'ls --help' for more information.`
		`$`

In addition to redirecting streams to files, the standard output of one program can be sent to the standard input of another with the pipe ("|") command. For this to work the program being piped to needs to contain logic that allows it to accept input from standard input. Many, but not all, command-line programs can accept input from standard input. For example:

```
$ cat counties_data.csv | grep CO
Boulder,CO,106392,740.0
Broomfield,CO,70465,33.55
Gilpin,CO,6243,150.0
Grand,CO,15734,1870.0
Jefferson,CO,582881,774.0
Larimer,CO,356899,2634.0
Weld,CO,324492,4017.0
$
```

```
$ cat counties_data.csv | grep CO | sort -n -k 3 -t ','
Gilpin,CO,6243,150.0
Broomfield,CO,70465,33.55
Boulder,CO,106392,740.0
Grand,CO,15734,1870.0
Jefferson,CO,582881,774.0
Weld,CO,324492,4017.0
Larimer,CO,356899,2634.0
$
```

```
$ cat counties_data.csv | grep CO | sort -n -k 3 -t ',' | cut -d ',' -f 1,3
Gilpin,6243
Broomfield,70465
Boulder,106392
Grand,15734
Jefferson,582881
Weld,324492
Larimer,356899
$
```