

# Getting Started with Mac OS X/Linux Command Terminal

Ziheng Yang  
University College London  
Updated March 2015

Asif Tamuri  
European Bioinformatics Institute

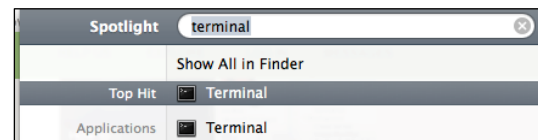
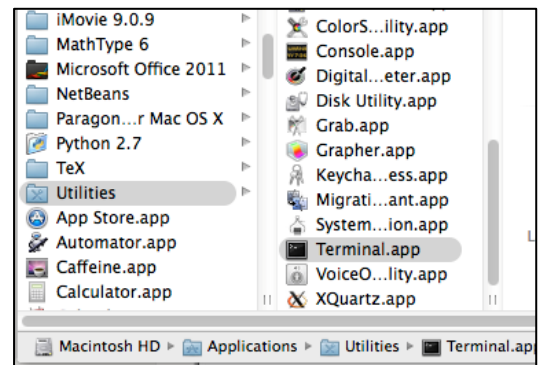


## What is a Command Terminal?

In the good old days, users interacted with computers through a command window. This is a text-based window for typing commands and receiving text-based output (see screen shot above). Mouse and menu do not work here but the command line is a powerful interface and is very convenient for running certain programs.

## How do I start a Command Terminal?

On Mac OS X, you can open the Terminal from Applications>Utilities>Terminal. Alternatively, you can use the Spotlight search in the top-right corner by searching for the keyword 'terminal'.



## Commands for manipulating directories (cd, md)

The OS X or Unix file system consists of a number of directories and sub-directories arranged hierarchically. The root directory is /. When I start the command terminal, I should be in my home directory. This may be /Users/ziheng/ or /home/ziheng, etc., depending on the system setup. The command prompt may show the current (working) directory, as follows: potto:~ ziheng\$. Here I am user 'ziheng' on a machine called 'potto', and I am in my home directory (which is indicated by the tilde symbol ~). The dollar symbol \$ is the command prompt.

Use cd to change directory. You can use an absolute path containing the entire directory structure. An *absolute* path starts with a backslash, which means we begin from the root of the file system. Without the leading backslash, the directory is *relative* to your working directory. The tilde character (~) represents your home directory. Thus no matter where you are,

```
cd /
```

will take you to the root directory, and

```
cd
```

```
or
```

```
cd ~
```

will take you to your home directory. Also

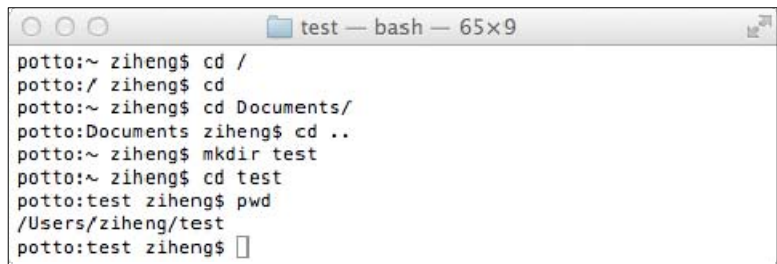
```
cd ~/test
```

will take you to the test directory inside your home directory.

`cd ..`  
moves up a level to the parent directory.

The command `pwd` prints the current (working) directory.

To make a new directory called `test` in the current directory, type `mkdir test`



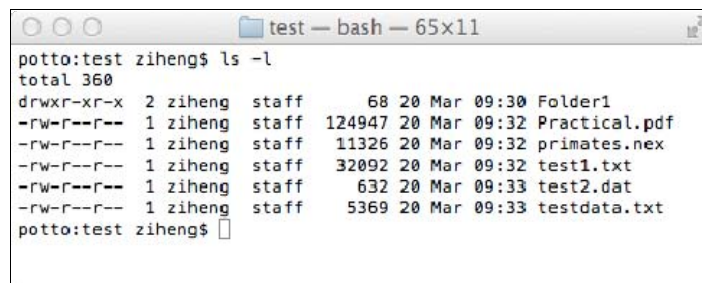
```
potto:~ ziheng$ cd /
potto:/ ziheng$ cd
potto:~ ziheng$ cd Documents/
potto:Documents ziheng$ cd ..
potto:~ ziheng$ mkdir test
potto:~ ziheng$ cd test
potto:test ziheng$ pwd
/Users/ziheng/test
potto:test ziheng$
```

### Getting directory listings (*ls*)

To list the contents of a directory, type `ls`

`ls -l`

The option `-l` means a long listing. The output may look like the following.



```
potto:test ziheng$ ls -l
total 360
drwxr-xr-x  2 ziheng  staff   68 20 Mar 09:30 Folder1
-rw-r--r--  1 ziheng  staff 124947 20 Mar 09:32 Practical.pdf
-rw-r--r--  1 ziheng  staff  11326 20 Mar 09:32 primates.nex
-rw-r--r--  1 ziheng  staff 32092 20 Mar 09:32 test1.txt
-rw-r--r--  1 ziheng  staff   632 20 Mar 09:33 test2.dat
-rw-r--r--  1 ziheng  staff  5369 20 Mar 09:33 testdata.txt
potto:test ziheng$
```

Each line provides the file's permissions (which we explain later), the owner (`ziheng`) and group (`staff`) of the file, the size of the file in bytes, the date and time the file was last modified and, finally, the filename.

### Wildcards

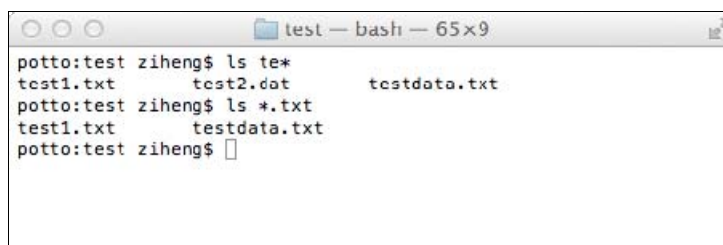
The special characters `*` and `?` can be used as wildcards when you specify file or directory names. The asterisk `*` means any number of any characters while `?` means one character of any kind. Thus

`ls te*`

will list all the files and directories that start with "te".

`ls *.txt`

lists all files that end with `.txt` (the text files).



```
potto:test ziheng$ ls te*
test1.txt      test2.dat      testdata.txt
potto:test ziheng$ ls *.txt
test1.txt      testdata.txt
potto:test ziheng$
```

### Copying and deleting files

The commands `cp` and `rm` are for copying and removing files.

```
cp test1.txt test2.txt
ls -lF
rm test2.txt
ls -lF
```

```
test — bash — 65x9
potto:test ziheng$ cp test1.txt test2.txt
potto:test ziheng$ ls
Folder1      primates.nex  test2.dat      testdata.txt
Practical.pdf test1.txt     test2.txt
potto:test ziheng$ rm test2.txt
potto:test ziheng$ ls
Folder1      primates.nex  test2.dat
Practical.pdf test1.txt     testdata.txt
potto:test ziheng$
```

Wildcards and relative paths can be used together. Suppose I have two directories `test` and `test2` in my home directory, and I am currently in `test`. Then the first command below will copy all files in the `test` folder that have the string `fish` in their names into the `test2` folder, and the second command will delete all files in `test2` that end with `.txt`.

```
cp *fish* ../test2/
ls -l ../test2/
rm ../test2/*.txt
ls -l ../test2/
```

### Viewing files on the screen

```
cat test1.txt
less test1.txt
less ../test2/test1.txt
```

The command `cat` shows the content of the file on the screen. This works for plain text files only. If the file is binary (executables and picture files are for example binary files), rubbish and noise will pop up. The command `less` does the same as `cat` but allows forward and backward movement within the file using the arrow and page-up/down keys.

### Running programs from the command line

Programs are executable files. You run the program by typing the file name at the command line. The following will run a program called `BPP`, which is in the `bin/` directory under my home account:

```
~/bin/bpp
```

### File permissions

```
drwxr-xr-x 2 ziheng users      4096 Mar 10 14:43 b/
-rw-r--r-- 1 ziheng users     15889 Mar 10 14:43 test1.txt
-rw-r--r-- 1 ziheng users       358 Mar 10 14:43 z.bat
```

The above shows the output from the `ls -lF` command.

In the **first column** above, **d** means a **directory** while dash (-) means a file. The next 9 fields specify the file permissions, in which **r**, **w**, **x**, mean **readable**, **writable**, and **executable** while a dash (-) means no permission. The 9 fields are in three blocks, for **user (owner)**, **group** and **other (world)**, respectively. Thus for the file `test1.txt`, `rw-` means the user can read and write but not execute the file, `r--` means the group can read but not write or execute, while `r--` means that other (everyone with an account on the system) can read the file but can't write or execute it. In other words, `test1.txt` is readable by everybody (user, group and other), writable by owner only, and is not executable. Note that you need executable permission to move (`cd`) into a directory.

Sometimes the file is an executable program, but you can't run it if its permission is not set correctly. This happens often when files are transferred across platforms. In that case you use the `chmod` (change mode) command to set the permissions. The following makes `program1` executable by user (owner) and group.

```
chmod ug+x program1
```

### A few tips

- Use forward-slash / to specify folders on OS X or UNIX. Use back-slash \ on Windows.
- Commands and file and directory names are case-sensitive on OS X or UNIX, while they are case-insensitive on Windows (MS-DOS).
- Given that different fields on the command line are separated by spaces, it is in general a good practice to avoid using spaces or other strange symbols in file names.

### Getting help

Use the command `man` to view the manual page for any particular command.

```
man cp
```

## Common useful Windows/Unix commands

Windows	UNIX/OSX	Function
<code>cd</code>	<code>cd</code>	Change directory (folder)
<code>md</code> or <code>mkdir</code>	<code>md</code> or <code>mkdir</code>	make a new directory
<code>dir</code>	<code>ls</code>	List files and directories
<code>copy file1 file2</code>	<code>cp file1 file2</code>	Make a copy of file1 and name it file2
<code>ren file1 file2</code>	<code>mv file1 file2</code>	Rename file1 as file2
<code>move file1 file2</code>		
<code>del</code>	<code>rm</code>	delete (remove) files
<code>rd</code>	<code>rmdir</code>	remove an empty directory
<code>time</code>	<code>time</code>	date and time mean different things in windows and unix
<code>date</code>	<code>date</code>	
<code>exit</code>	<code>exit</code>	exit
<code>help</code>	<code>man</code>	help or manual
<code>more</code>	<code>more</code>	show file a screen a time
<code>type</code>	<code>cat</code>	show file
<code>↑, ↓</code>	<code>↑, ↓</code>	Use the Up & Down arrow keys (↑ and ↓) to
<code>←, →</code>	<code>←, →</code>	cycle through past commands. Then use ← and → or Ctrl← and Ctrl→ to move around to edit.
<code>Tab</code>	<code>Tab</code>	The Tab key completes file or folder names
<code>&gt;</code>	<code>&gt;</code>	redirection: screen output will go into file
<code>&lt;</code>	<code>&lt;</code>	redirection: keyboard input will come from file
<code> </code>	<code> </code>	pipe: output from one program will be input to the next program
<code>Esc</code>	<code>Esc</code>	Cancel command
<code>Ctrl-C</code>	<code>Ctrl-C</code>	terminate job
	<code>nice +20 mb</code>	run a job at low priority
	<code>nice +20 mb &amp;</code>	& places the job at the background
	<code>Ctrl-Z</code>	pause a foreground job
	<code>bg</code>	place the paused job at the background