NYC DATA SCIENCE ACADEMY

Unix Shell Commands

Data Science Bootcamp

OVERVIEW

- **Introduction to Linux**
- **File System and File Operations**
  - **Basic file commands**
  - **Creating files**
- **Text-processing commands**
- **Other useful commands**

What is an OS?

- An *operating system* (OS) is a program that provides a variety of services designed to facilitate your work on the machine, and to keep different users from interfering with one another.
- OS services include:
  - **File system** - create/move/rename/delete/share/etc. files
  - **Scheduler** - run jobs, taking into account priorities, fairness, etc.
  - **I/O and communication** - read/write to disk, manage internet connections

What OS's are popular

- Currently, the most commonly used operating systems are:
  - Windows (Microsoft)
  - MacOS (Apple)
  - Linux (Open Source distributions)
  - Android (Google)
- Mac OS X and Linux are versions of **Unix**; Android is a version of Linux. Windows is a completely separate OS.
- MacOS and Windows are commercial systems - you pay for them. Linux and Android are open source, i.e. free.
  - Versions of Android used on most devices are extended with proprietary code, so are not entirely open source.

Interacting with an OS

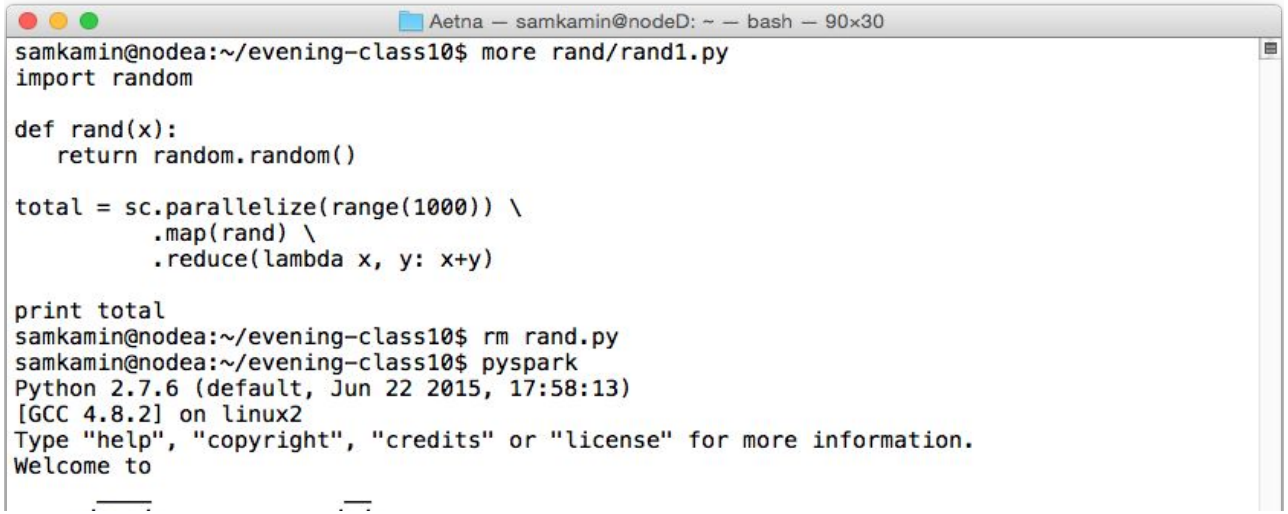
- As a user, you have two ways to request OS services:
  - A command-line interface (CLI, also called a “shell”)
  - A graphical interface (GUI)
- Operating systems generally provide both.
  - On Macs and Linux machines, open a command window using the “terminal” app. On Windows systems, select Start menu item “Command Prompt”.
  - You can open a command window on a remote Unix machine using the `ssh` (“secure shell”) command.
- Software developers and data engineers generally prefer to use the command line. **For this class, we will use the command-line.**

Linux distributions

- Linux is distributed by a variety of organizations, each with its own variations. They share a similar core, or kernel, but vary in what software is included, how new packages are installed, what GUI is included, and so on.
- Linux is distributed both in pure open source form and by commercial companies.
  - Open source distros: Debian, Slackware
  - Commercial distros: OpenSUSE, Red Hat, Ubuntu

Unix shell

- A terminal is a window for users to type commands. In Unix systems, this is called a *shell*. Here is a (partial) screenshot of a Mac terminal window:

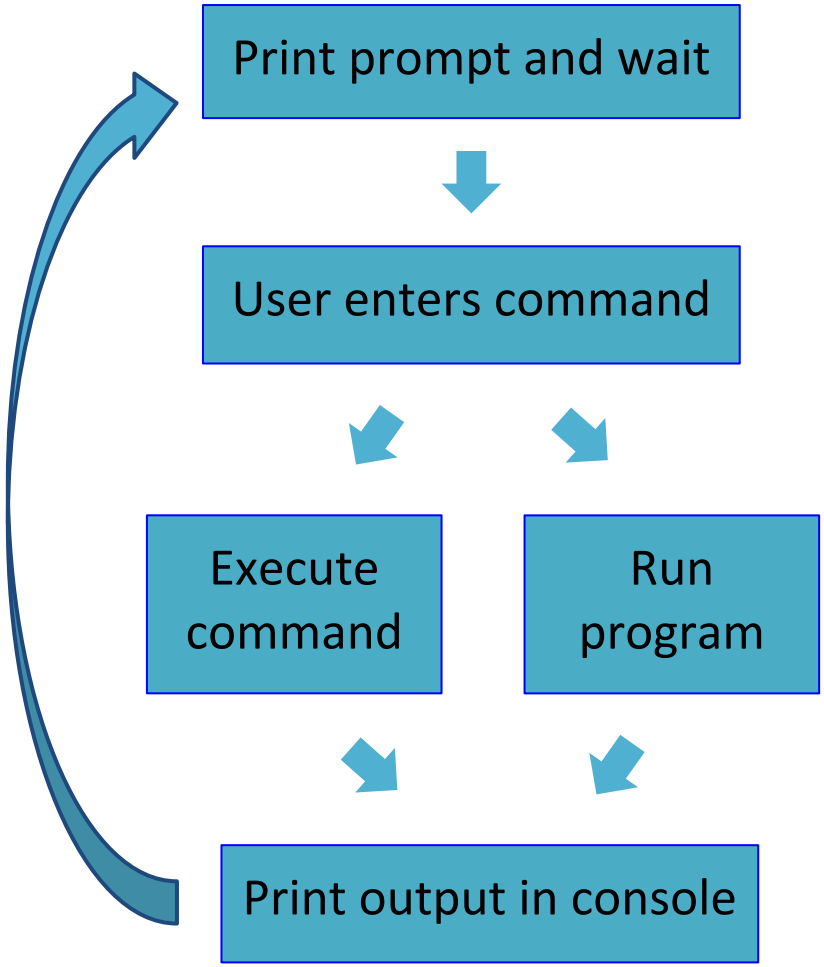


- "`samkamin@nodea:~evening-class10$`" is the *command prompt*. Everything after the prompt is a command typed by the user (`more rand/rand1.py`, `rm rand.py`, `pyspark`). Any line not starting with a prompt is output from the shell or another program.

Command-Line

The shell runs in a continuous loop:

1. Print a prompt; wait for user to enter a command.
2. When the user enters a command, interpret it, and act accordingly; either:
    i. Execute the command, or
    ii. Run the program requested by the user
3. Print output in the terminal.
4. Go to step 1.



Exercise 1 - Log in to Linux machine

- In this class, you will be using Linux by logging into our server remotely.
  - If you have a Mac:
    - Bring up a terminal window by running the “terminal” app.
    - Enter command: `ssh yourname@18.220.240.108`
    - You will be prompted for a password.
  - If you have a PC:
    - Download putty.
    - In the Session window, enter `18.220.240.108` and port `22`, and click the Open button.
- It should be obvious when you have succeeded. Enter: `date` and return.

Exercise 1 - Log in to Linux machine

- Enter these commands:

```
ls
ls /etc
```

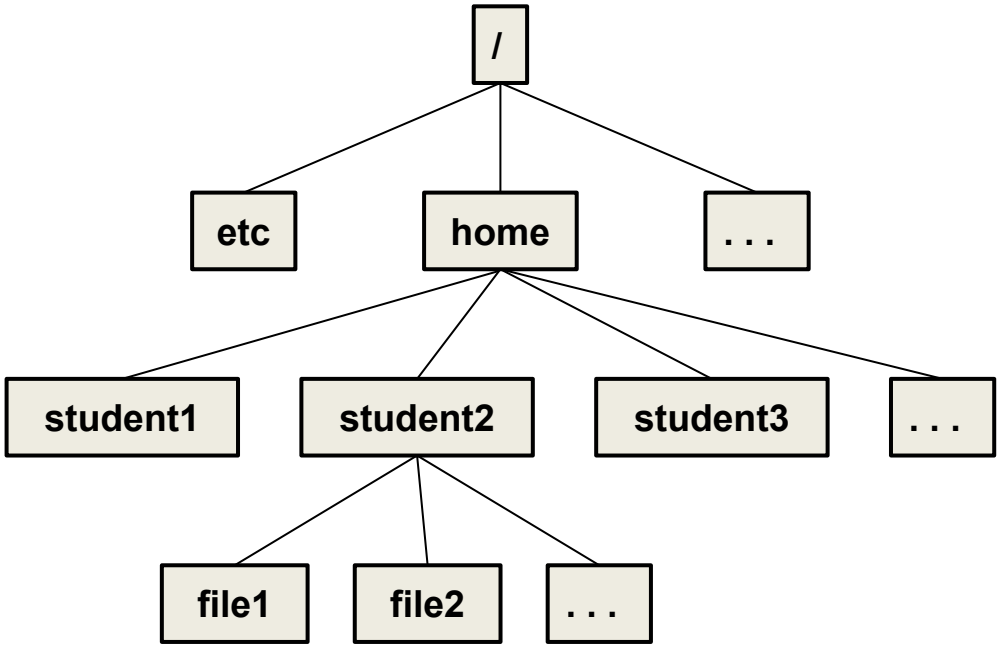
  - Explanation: ls lists the files in a directory.
    - The first command printed nothing, because your own directory is currently empty.
    - The /etc directory contains a lot of system files.

OVERVIEW

- Introduction to Linux
- File System and File Operations
  - **Basic file commands**
    - Creating files
- Text-processing commands
- Other useful commands

The Filesystem

- Filesystem hierarchy, looks like an inverted tree.



- In Unix, the traditional word for folder is *directory*. We use “folder” and “directory” interchangeably.

The working directory

- A user is always “in” a directory, called the *working directory*.
  - You can access the files in the directory directly, using their simple names.
  - If you ask for a list of files, the system will list files in this directory.
- To see what your working directory is right now, enter: `pwd`
- We will soon see commands to change your working directory.
- For each user, there is a directory, with the same name as the user, which is the working directory when you log on. It is called your *home directory*. (After exercise 1, you are in your home directory.)

Pathnames

- In the CLI, you often have to type a file's name - you don't have a GUI. In those cases, you use its *pathname*, in one of two forms.
- The *full* (or *absolute*) *pathname* of a file is its name with all containing folders, up to the root, separated by /. The root directory is just '/'.
  - The full pathname of my file test.csv is: `/home/samkamin/test.csv`.
- The *relative pathname* of a file is its name with all containing folders, up to the current *working directory*, with no opening '/'.
  - The relative pathname of my file `ex1` in my directory `examples` (not shown in the tree) is: `examples/ex1`.
- This is what's special about the working directory: it is the base for relative pathnames.

File system commands

- File systems provide operations for things like copying and renaming files. Unix includes lots of other handy "utility" functions for all sorts of things, from sorting files to giving the current time. We'll try to tell you about the most useful ones.
- The basic file operations include:
  - create a file/folder
  - copy, remove or move files/folders
  - change the permission or ownership of files/folders

  We'll start with these.

ls: List files in directory

- The `ls` command lists the file at the given path. With no arguments, that defaults to the current working directory. (Remember: when you first log on, the working directory is your home directory.)
  - In exercise 1, you used "`ls`" to list the files in your home directory, and "`ls /etc`" to list files in the /etc directory.
- Finding documentation about Unix commands:
  - "`man ls`" produces a "man page." This is a standard Unix documentation format.
  - "`ls --help`" produces a similar documentation list.
  - Googling "`ls`" or "`unix ls`" produces many hits, though many are copies of the man page. You can also try "`unix ls tutorial`".

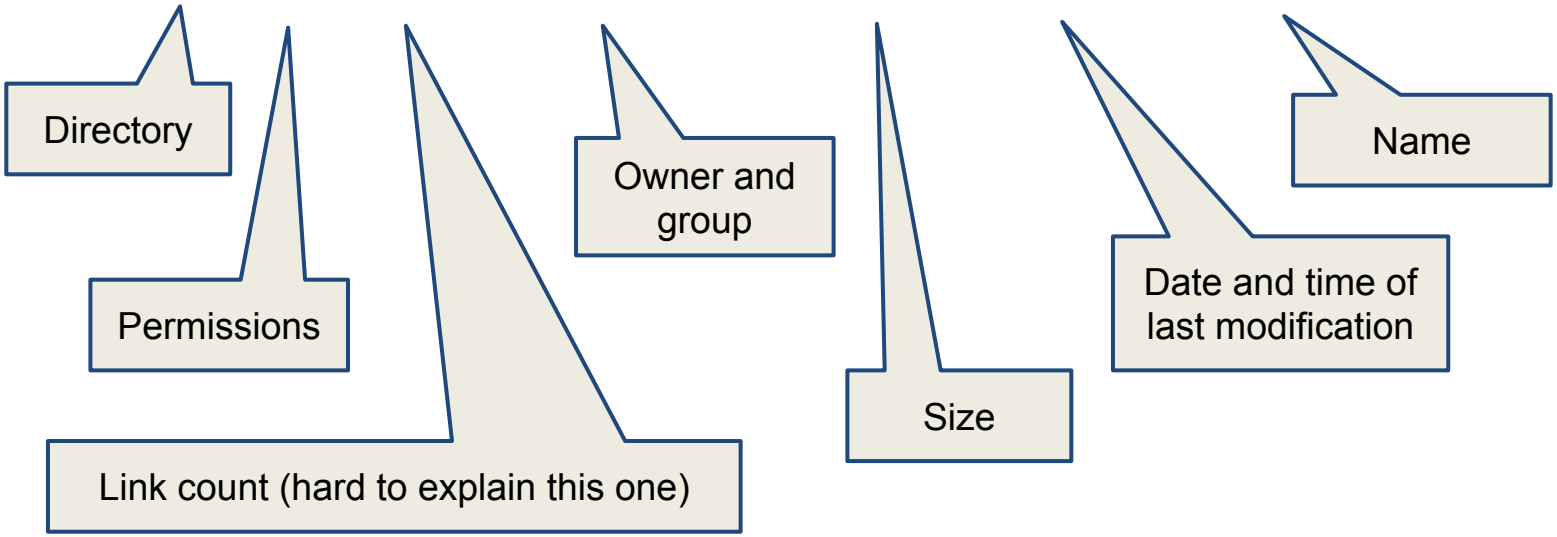
ls: List files in directory

- In Unix, commands often have special arguments, called *options*, introduced by a single (or sometimes double) dash. We'll talk about a few of the options for `ls` (there are about 40 all together).
- "`-a`" produces a file listing that includes "hidden files" - files whose name starts with a period.
  - There are two special hidden files: "." (a single period) is the current directory (the one being listed) and ".." (two periods) is its parent directory.
- "`-t`" sorts the file listing by modification date.
- Combine arguments by including several separately or by combining them after a single dash: `ls -a -t` or `ls -at`

ls: List files in directory

- Argument "-l" produces a long form of the file listing, including file ownership, size, permissions, and other information.

```
samkamin@nodeD:~$ ls -l
total 1588
-rw-rw-r-- 1 samkamin samkamin      4 Jul  6 17:19 abc.txt
drwxr-xr-x 3 samkamin samkamin   4096 Aug  4 15:05 baseball
-rw-r--r-- 1 samkamin samkamin     36 Jul 23 18:00 bigdata.txt
drwxrwxr-x 3 samkamin samkamin   4096 Aug 20 16:28 class3
```



cd: Change the working directory

- The `cd` command changes the current working directory.
- With an argument:
  - `cd` *pathname* changes the working directory to that pathname. The pathname can be absolute or relative.
- Without an argument:
  - cd alone changed the working directory to your home directory.
  - Tilde (~) is an alias for your home directory. "`cd ~`" is the same as "`cd`". You can write "`cd ~/subdir`" to go to `subdir` in your home directory.
- Remember that ".." is an alias for the parent directory of the working directory, so "`cd ..`" is very useful.

mkdir: Make a new directory

- `mkdir` creates a new, empty directory.
  - Make a new directory in your home directory:

```
$ cd ~
$ mkdir examples
$ ls examples
$
```

mkdir: Make a new directory

- Make multiple, nested directories, regardless of whether parts of the specified path exists already, by adding a flag: `mkdir -p`

```
$ mkdir -p ~/examples/multiple/levels/down
$ ls -R ~/examples   # -R means list contents “recursively”
examples:
multiple

examples/multiple:
levels

examples/multiple/levels:
down

examples/multiple/levels/down:
```

cp: Copy files

- The `cp` command supports making copies of any file. The syntax is:

```
cp [-r] source destination
```

  - Add the `-r` option if you are copying a folder.

- The destination can be a filename or an existing folder.
  - If an existing folder, `source` is copied into that folder.
  - If not an existing folder, a copy of `source` is made, with the `destination` as its name.

- *Be careful*, as the `cp` command can overwrite existing files.

cp: Copy files

- Here is an example of copying. (Note: # is the comment character; everything after a # is ignored.):

```
$ cd    # go to home directory

# copy a file into home directory; note the .
$ cp /etc/magic .
$ ls
examples  magic

# copy a file into home directory with a new name
$ cp /etc/hosts nethosts
$ ls
examples  nethosts magic
```

mv: Move or rename files

- The `mv` command supports the efficient moving of files. The syntax is the same as copy, but no need to add a `-r` option even for moving folders.

```
mv source destination
```

- Example:

```
$ mv magic examples

$ ls    # no *Price.csv in current folder
examples  nethosts

$ ls examples
magic
```

mv: Move or rename files

- When the destination is a new name (rather than an existing folder), mv acts as a renaming operation.

```
mv old_file_name new_file_name
```

```
$ cd examples
$ mv magic price.csv
$ ls
price.csv
```

rm: Delete files

- The `rm` command is for removal of files.
- **Note: once a file is deleted through the command-line, it is removed permanently.** There is no "trash can" in Unix where deleted files are moved. With the wrong arguments, you could delete all your files permanently!

```
$ cd ~/examples
$ cp price.csv price_copy.csv
$ ls
price.csv price_copy.csv
$ rm price_copy.csv
$ ls
price.csv
```

rm: Delete files

- The `rmdir` command supports deleting empty directories, but is not often used.

```
$ cd ~
$ rmdir examples
rmdir: failed to remove ‘examples’: Directory not empty

$ mv examples/price.csv .  # move file to empty examples
$ rmdir examples
```

- Instead, use: `rm -r`

```
$ mkdir examples
$ mv price.csv examples
$ rm -r examples
$ ls examples
ls: cannot access examples: No such file or directory
```

- But be super careful with rm -r!

Commands to view files

- The next group of commands are the ones that let you view the contents of files.
- These operations include: cat and less.

cat: Dump file contents to stdout

- The simplest command to view the file contents is `cat` (short for 'concatenate'). It prints the entire file to the console ("standard output").

```
$ cp /etc/hosts nethosts
$ cat hosts   # We could cat /etc/hosts just as well
127.0.0.1   localhost
127.0.1.1   localhost
216.230.228.88   nodeD.nycdatascience.com nodeD
...
```

- `cat` can take multiple files as arguments. It dumps them all to standard output, with no line breaks in between.
- Be careful with `cat`. If the file is very large, it could print for a long time. To stop it, type ^C (ctrl-C).

less: View files one screenful at a time

- We can use the `less` command to view longer files without writing everything out to the screen at once. Less will only print one screenful of data to the terminal.

```
$ less /etc/services
```

  - Scroll up/down one line at a time with arrow keys.
  - Use the space bar to scroll through a screen at a time.
  - Use the / key to search for a term: `/apple`
  - Press h to see a quick list of other options.
  - Press q to exit less.

Exercise 2 - Practice file commands

- Your home directories are empty. We're now going to do some commands that will create/copy/delete files and directories. Just enter these commands. After doing this, take a few minutes to practice the commands on your own.
  - `cp /mnt/data/iris.csv .`
  - less iris.csv
  - mkdir flowers
  - cp iris.csv flowers
  - cp -r flowers irises
  - ls
  - cd flowers
  - rm iris.csv
  - cd ..

Filename globs

- Many commands can operate on more than one file. To facilitate specifying multiple files, Unix commands can include filename patterns, called *globs*.
  - If you've heard of "regular expressions," globs are like them but simpler.
- Globs are filenames containing *metacharacters* that can stand in for different sequences of characters. The most useful glob characters is '*', which stands for any sequence of characters. E.g.
  - `ls *txt` - list the names of all files whose names end with txt.
  - `cat *abc*` - list the contents of all files whose names contain 'abc'.

Exercise 3 - File commands with globs

- Directory `/etc` has a lot of files, as we've seen. We'll copy some of them just to play around with globs.
- Make a directory etc in your own home directory: `mkdir etc`
- Copy the configuration files from `/etc` to `etc`:

```
cp /etc/*conf etc
```

- Change directories to your etc directory (`cd etc`) and do `ls`.
- Concatenate all the files that start with "de" into a file `de-files`:

```
cat de* > def-files
```

- Go back to your home directory (`cd`, `cd ~`, or `cd ..`) and delete the `etc` directory (`rm -r etc`).

OVERVIEW

- Introduction to Linux
- File System and File Operations
  - Basic file commands
  - **Creating files**
- Text-processing commands
- Other useful commands

Text files

- We will be working with “plain text files” - files with characters but no formatting information.
- Plain text files are created and edited with text editors. There are GUI-based (cut-and-paste) editors for plain text files (e.g. wordpad on Windows, TextEdit on macs). However, we will use the editor most used by data engineers: vi.
- Today we’ll just do a very little bit with vi. You’ll have to learn how to use it better when we do more data engineering later in the bootcamp.

Creating text files from stdout

- You can create files by “redirecting” the output of a command into a file.
- Two examples:

```
$ echo Some random text > random.txt
$ cat random.txt
Some random text

$ ls -l > lsout
$ cat lsout
total 44072
drwx------    4 samkamin  staff       136 Jun 23 12:25 Applications
drwxr-xr-x    9 samkamin  staff       306 Aug 27 18:17 DataScienceBootcamp
drwx------+   4 samkamin  staff       136 Sep 17 14:01 Desktop
drwx------+  22 samkamin  staff       748 Sep 11 12:01 Documents
drwx------+ 463 samkamin  staff     15742 Sep 17 14:01 Downloads
drwx------@  17 samkamin  staff       578 Sep 16 10:20 Dropbox (NYCDSA)
-rw-r--r--    1 samkamin  staff     14518 Sep  3 16:00 HD_log.txt
... etc ...
```

- The trick is the “>”, which says: instead of printing to the terminal, put the output into a file. This is called “I/O redirection.”

Creating text files with vi

- `vi` is a text editor in Unix systems.
- You start vi by typing: `vi file`
  - `file` will be opened if it exists, created if it doesn't.
- `vi` is **modal**, meaning you are always in one of two modes:
  - *Insert mode*: Characters you type go into the file. (But careful: the file is not saved until you request it.)
  - *Command mode*: Characters are interpreted as commands, e.g. the character k means "move the cursor up one line".

vi - Basic operation

- By default, when you open a file using vi, it is in command mode, and some information about the file is displayed at the bottom of the screen.

```
Hello, World!

"input.txt" 1L, 14C                                1,1           All
```

- Press 'i', then you can see the "--INSERT--" characters at the bottom left corner. Press "Esc" to quit insert mode, go back to command mode.

```
Hello, World!

-- INSERT --                                       1,1           All
```

- In command mode, type ZZ to save the file and exit vi.
- The next two slides have a cheat sheet for basic vi commands.

vi - Basic operation

- Remember: vi is modal: you are either in insert mode (what you type goes in the file) or command mode (what you type is interpreted as a command to the editor).

Enter/exit insert mode	Command mode
a - insert after cursor	x - delete one character
i - insert before cursor	dw - delete one word
o - insert a new line below	dd - delete the line
O - insert a new line above	D - delete the rest of the line
ESC - exit insert mode	u - undo the last action

vi - Basic operation

- You can move cursor on both insert and edit modes, while the saving commands can only works in the edit mode.(Since it will insert characters in the insert mode)

Moving cursor	Saving
↑ or k - up one line	ZZ - save and exit
↓ or j - down one line	:wq - save and exit
← or h - backward one character	:w - save without exiting
→ or l - forward one character	:q! - exit without saving

Exercise 4 - Using vi

- You should be in your home directory. (If you're not sure, you can use `pwd` to see where you, and `cd` to go to your home directory.)
- We will do two simple operations with vi, creating a small file, and editing a file.
- Create a small file from scratch; enter exactly what is given here:
  - At the Unix prompt, enter: `vi smallfile.txt`*<return>*
  - Enter: `a`This is line 1*<return>*This is line 2.*<return>*
  - Hit the ESC key.
  - Type: `ZZ`
  - Type: `cat smallfile.txt`

Exercise 4 - Using vi

- Edit a file:
  - At the Unix prompt, enter: `ls -l /mnt/data > filelist<return>`
  - Type `cat filelist<return>`
  - At the Unix prompt, enter: `vi filelist<return>`
  - Use the arrow keys to move the cursor up and down in the file.
  - At any point, enter dd to delete a line.
  - Use the right and left arrows to move within an existing line. At any point, enter D to delete everything after the cursor.
  - Type: `ZZ`
  - Type: `cat filelist`

OVERVIEW

- Introduction to Linux
- File System and File Operations
  - Basic file commands
  - Creating files
- **Text-processing commands**
- Other useful commands

Text Processing

- Unix systems have a ton of useful commands for searching and modifying text files. We'll introduce just two of them:
  - `grep` - Search through a given file using a string
  - wc - Count lines and words
  - `sort` - Sort all lines in file

grep

- grep is a tool to search for words in files.

```
grep word file1 [file2 ... ]
```

- Here are two commands that search files from your home directory.

```
$ grep kamin /etc/passwd
samkamin:x:1025:1027:Samuel Kamin,,,:/home/samkamin:/bin/bash
```

- You can search for all the files in a directory like this:

```
$ cd
$ grep hdfs pythontest/*
pythontest/hdfs.py:from hdfs import TokenClient
pythontest/hdfs.py:print dir(hdfs)
```

- The * is an example of a file pattern, or “glob;” these are really useful things in Unix, which we’ll learn more about later in the course.

wc

- `wc` ("word count") is simple but useful: Give the number of lines, words, and characters in a file.

```
wc file1 [file2 ... ]
```

- Find the lengths of "configuration" files in /etc:

```
$ wc /etc/*conf
     85     461    2981 /etc/adduser.conf
     10      53     321 /etc/blkid.conf
    184     247    7773 /etc/ca-certificates.conf
     44     225    1332 /etc/colord.conf
    ...
     39     218    1260 /etc/ucf.conf
      4      39     321 /etc/updatedb.conf
   2269    8773   70196 total
```

sort

- Sort is used to sort the lines in your file. You can sort it by the entire line or one specific field. The output from ls -l is a good example, because it has multiple fields.

```
$ ls -l > files.txt
$ cat files.txt
-rw-rw-r-- 1 samkamin samkamin      4 Jul  6 17:19 abc.txt
drwxr-xr-x 3 samkamin samkamin   4096 Aug  4 15:05 baseball
-rw-r--r-- 1 samkamin samkamin     36 Jul 23 18:00 bigdata.txt
... etc. ...
```

- Sort on the first field (permissions).

```
$ sort files.txt
drwxrwxr-x 2 samkamin samkamin   4096 Aug 12 12:44 pythontest
drwxrwxr-x 2 samkamin samkamin   4096 Aug 20 20:58 evening-class4
drwxrwxr-x 2 samkamin samkamin   4096 Jul 11 11:14 hadoopTeaching
... etc. ...
```

sort

- Sort on the time field:

```
$ sort -k8 files.txt
-rwxr-xr-x 1 samkamin samkamin    231 Jul  1 10:17 map.py
-rwxr-xr-x 1 samkamin samkamin    374 Jul  1 10:17 reduce.py
-rw-r--r-- 1 root     root      20480 Jul 10 10:23 h.tar
```

- Sorting on the size doesn't quite work ...

```
$ sort -k5 files.txt
-rw-rw-r-- 1 samkamin samkamin      0 Sep 17 17:09 files.txt
-rw-r--r-- 1 samkamin samkamin 112128 Aug 18 14:48 week1-88c.tar
-rw-r--r-- 1 samkamin samkamin    119 Aug 13 11:27 ex.py
```

- … because sorting on *characters* is not the same as sorting on *numbers*.

```
$ sort -k5 -n files.txt
-rw-rw-r-- 1 samkamin samkamin      0 Sep 17 17:09 files.txt
-rw-rw-r-- 1 samkamin samkamin      4 Jul  6 17:19 abc.txt
-rw-rw-r-- 1 samkamin samkamin     28 Aug 19 12:05 test.csv
```

Exercise 5 - Text-processing commands

- Make sure you are in your home directory. See if you still have iris.csv there; if not, copy it again: `cp /mnt/data/iris.csv .`
- Use wc to find the number of lines in the file.
- Use grep to find all the “setosa” lines.
  - You can extract these lines and put them into a separate file by using I/O redirection (>).
- Sort on the first field (Sepal.length).
  - Sorting on the second field isn’t quite so simple. We’ll do that a little later.

OVERVIEW

- Introduction to Linux
- File System and File Operations
  - Basic file commands
  - Creating files
- Text-processing commands
- **Other useful commands**

Useful commands

- Unix has a huge number of useful commands. We give a sampling.
  - `date` - Get the current date and time
  - ssh - sign on to a remote machine
  - scp - copy a file from or onto a remote machine
  - `curl` and `wget` - Download a web page

date

- Get the current date and time.

```
$ date
Thu Sep 17 17:24:37 EDT 2015
```

ssh

- Log in to a remote machine

```
$ ssh samkamin@18.220.240.108
Welcome to Ubuntu 14.04.2 LTS (GNU/Linux 3.13.0-62-generic
x86_64)

 * Documentation:  https://help.ubuntu.com/
Last login: Thu Sep 17 16:53:57 2015 from 157.130.31.226
$
```

scp

- Use scp ("secure copy") to copy files to or from a remote machine.
- Local to Remote

```
$ scp local_file.tar.gz
user1@remote_ip:path/to/data/local_file

$ scp local_file.tar.gz user1@remote_ip:path/to/data/

$ scp local_file.tar.gz
user1@remote_ip:path/to/data/renamed
```

- Remote to Local

```
$ scp user1@remote_ip:path/to/data/remote_file local_file
```

curl: Getting data From the web

- Use `curl` to download an html page from the web. The -O (that's capital O, not zero) means the file should be saved locally with the same name as it has remotely (`curl1.html`, in this case). (`wget` is a very similar command; some systems have `wget`, some have `curl`, some have both.)

```
$ curl -O "http://linuxcommand.org/man_pages/curl1.html"
  % Total    % Received % Xferd  Average Speed   Time    Time
Time  Current
                                 Dload  Upload   Total   Spent
Left  Speed
100 69936  100 69936    0     0   317k      0 --:--:-- --:--:--
--:--:--  317k
$ less curl1.html
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<HTML>
... etc. ...
```

Exercise 6 - Using curl

- You should be in your home directory. (If you're not sure, you can use `pwd` to see where you, and `cd` to go to your home directory.)
- Go to any tab in an open browser and right-click on the address field and copy it.
- Go back to your ssh window. Enter curl -O and then paste the URL you just copied.
- Enter ls and you should see a file whose name is the same as the last part of the URL. Cat the file to see what's in it.

Aside: Finding documentation about Unix commands

- Unix is widely used on the internet, so there are numerous resources.
- On the command line, there are a number of options. Not all of these are available on all systems.
  - "`man` *command*" produces a "man page." This is a standard Unix documentation format.
  - "info *command*" is another standard format.
  - "*command* `--help`" produces similar documentation to man.
- Googling "unix *command*" is likely to produce many hits, though many will be copies of the man page. You can also try "`unix` *command* `tutorial`".

Exercise 7 - Using google

- If there's something you think there should be a command for, your should try searching on google.
- Find a calculator by searching on "unix calculator." You will find a lot of hits, but should find something you can use pretty quickly.
- Look at the man page for "sort", or search on google, to find out how to sort iris.csv on the second field.

Summary: Unix

- Unix is an operating that is very popular among hackers of all kinds. It is the basis of MacOS, iOS, and Android. Most internet servers use it.
  - More specifically, Linux is the flavor of Unix most widely used on the web and in clusters. MacOS and iOS use a different flavor of Unix. (But all flavors are pretty much the same.)
  - MS Windows is the only major OS that has an entirely different origin.
- Almost every cluster (e.g. Google's and Amazon's gigantic clusters) uses it. That makes it the preferred OS of data engineers. And that is why it is important for data scientists to be familiar with it.