NYC DATA SCIENCE ACADEMY

Introduction to Linux

Data Science Bootcamp

OVERVIEW

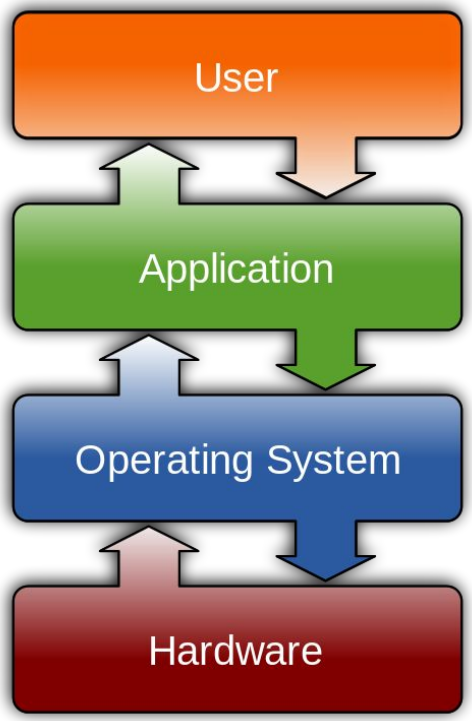
- **Operating Systems and Linux**
  - **File System and File Operations**
    - **Basic file commands**
    - **Creating files**
  - **Text-processing commands**
  - **Other useful commands**

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/modify/delete/etc. files
  - **Scheduler** - run jobs, etc
  - **I/O and communication**

    - read/write to disk, manage networks.



Image

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 screenshot of Linux:

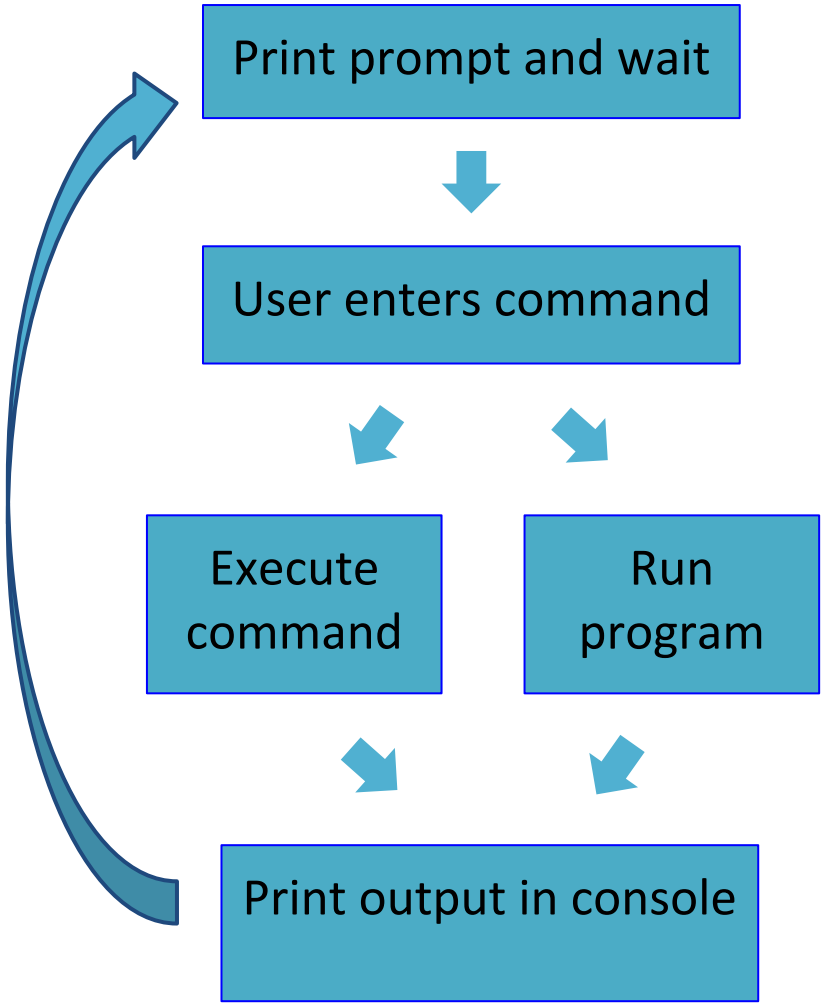
```
> docker run -it --rm nycdsa/linux-toolkits
 * Starting OpenBSD Secure Shell server sshd                                [ OK ]
 * Starting MySQL database server mysqld                                    [ OK ]
Creating MySQL Database
Writing default config to: /home/ubuntu/.jupyter/jupyter_notebook_config.py


    __    _                    ______                ____   _ __
   / /   (_)___  __  ___  __   /_  __/___ _____     / / /__(_) /_____
  / /   / / __ \/ / / / |/_/    / / / __ \/ __ \   / / //_/ / __/ ___/
 / /___/ / / / / /_/ />  <     / / / /_/ / /_/ /  / / ,< / / /_(__  )
/_____/_/_/ /_/\__,_/_/|_|    /_/  \____/\____/_/_/|_/_/_/\__/____/


Ubuntu 16.04.5 LTS
Python 3.5.2
mysql  Ver 14.14 Distrib 5.7.23, for Linux (x86_64) using  EditLine wrapper

ubuntu@8b5212215bfc:~$
```

- `ubuntu@<host_ip>:~$` is the *command prompt*.

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 run Linux environment using docker container.
- Run the following command in your terminal to start the container.

```
> docker run -it --rm nycdsa/linux-toolkit
```

- You should see a similar screen when your docker container is running. Enter: `date` and return.
- Read Docker QuickStart for more information.

Exercise 1 - Log in to Linux machine

- If you have a remote Linux server to access, then you can use SSH to connect.
  - If you have a Mac (or **Git bash** on Windows):
    - Bring up a terminal window by running the “terminal” app.
    - Enter command: `ssh <username>@remote-ip`
    - You will be prompted to enter your password.
  - If you have a PC:
    - Download putty.
    - In the Session window, enter `remote-ip` 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
```

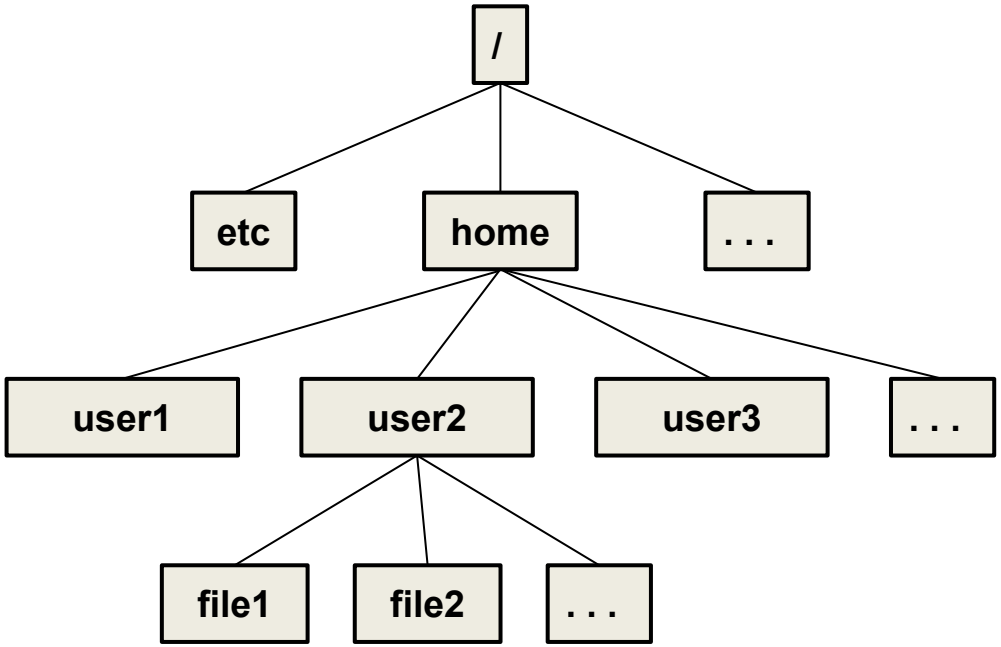
  - Explanation: `ls` lists the files in a directory.
    - The first command printed nothing, because your own directory is currently empty.
    - The /etc directory contains a lot of system files.

OVERVIEW

- Operating Systems and 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 '/'.
  - Assume we have a user of username `linuxuser`, The full path of the file `sample.txt` in it's home directory is:

```
/home/linuxuser/sample.txt
```

- The *relative pathname* of a file is its name with all containing folders, up to the current *working directory*, with no opening '/'.
  - Assuming you're currently in you home directory, then the relative path of the file `sample.txt` in `linuxuser`'s directory is:

```
../linuxuser/sample.txt
```

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 "`linux ls`" produces many hits, though many are copies of the man page.

ls: List files in directory

- In Linux, 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.

```
test_user@ip-172-31-11-204:~$ ls -l /etc
total 808
drwxr-xr-x 3 root root    4096 May 22 10:55 acpi
-rw-r--r-- 1 root root    3028 May 22 10:54 adduser.conf
drwxr-xr-x 2 root root    4096 Jul  1 22:14 alternatives
drwxr-xr-x 3 root root    4096 May 22 10:54 apm
... ...
permissions  | links | user | group     | size | last modification time | name
```

- The permissions are broken into 4 sections. For example: `drwxr-xr--`
  1. `d`: '–' -> file, '`d`' -> directory, '`l`' -> link
  2. `rwx`: permissions for the owner
  3. `r-x`: permissions for members of the group owning the file
  4. `r--`: permissions for other users

  Where '`r`'-> read, '`w`'-> write/modify, '`x`'-> execute, '–'-> no permission

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 magic nethosts
```

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
examples nethosts

$ ls examples
magic multiple
```

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 magicData
$ ls
magicData multiple
```

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 magicData magicData.txt
$ ls
magicData magicData.txt multiple
$ rm magicData.txt
$ ls
magicData multiple
```

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/magicData .
$ mv examples/multiple .  # move file to empty examples
$ rmdir examples
```

- Instead, use: `rm -r`

```
$ ls
magicData multiple nethosts
$ mv magicData multiple
$ rm -r multiple    # be careful with rm -r
$ ls multiple
ls: cannot access examples: No such file or directory
```

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 nethosts   # We could cat /etc/hosts as well
127.0.0.1   localhost localhost.localdomain localhost4
localhost4.localdomain4
::1         localhost6 localhost6.localdomain6
```

- `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.

```
$ wget https://raw.githubusercontent.com/nycdatasci/bootcamp/master/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

- 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 (the ">" character redirect standard output to a file):

```
cat de* > de_files
```

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

OVERVIEW

- Operating Systems and 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 /home > ls_out
$ less ls_out
total 1168
drwxr-xr-x  2 7dandiaz          7dandiaz          4096 Aug 11  2015 7dandiaz
drwxr-xr-x  8 abohun            abohun            4096 Mar 15  2016 abohun
drwxr-xr-x 11 acone             acone             4096 Apr 18 14:26 acone
drwxr-xr-x  7 adam              adam              4096 Aug 25  2015 adam
drwxr-xr-x  4 aglander          aglander          4096 Jan 26  2016 aglander
drwxr-xr-x  8 akosar            akosar            4096 Aug 31 20:34 akosar
... 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".
- 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.

vi - Basic operation

- Let's first create a file using echo and open it with vi:

```
$ echo "Hello, World!" > input.txt
$ vi input.txt
```

- You will see the screen looks like this:

```
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.

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>`
  - Hit *a* or *i* to enter insert mode and then type:

    `This is line 1<return>This is line 2.<return>`
  - Hit ESC key to quit insert mode.
  - Type: `ZZ`
  - Type: `cat smallfile.txt`

Exercise 4 - Using vi

- Edit a file:
  - At the Unix prompt, enter: `ls -l / > 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

- Operating Systems and 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 ... ]
```

- The file `/etc/passwd` is a text file which contains a list of system's accounts, giving for each account some useful information like user ID, group ID, home directory, shell, etc. We can use grep to search lines that contain the string "user":

```
$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
...etc...
$ grep user /etc/passwd
linuxuser:x:1731:1732::/home/linuxuser:/bin/bash
```

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. Let's create a file ls_root with the following command:

```
$ ls -l / > ls_root
$ cat ls_root
total 173
drwxr-xr-x    2 root root  4096 Sep  7 15:39 bin
drwxr-xr-x    4 root root  5120 Sep 20 06:40 boot
drwxr-xr-x    3 root root  4096 Sep  2 11:44 data
drwxr-xr-x.   5 root root  4096 Aug 21  2015 data1
... etc ...
```

- Sort on the first field (permissions).

```
$ sort ls_root
drwx------    2 root root 16384 May 29  2015 lost+found
drwx------    7 root root  4096 Aug  6  2015 root
drwxrwxrwt   29 root root 28672 Sep 26 00:59 tmp
drwxr-xr-x   10 root root  4096 May 29  2015 usr
... etc. ...
```

sort

- Sort on the name field:

```
$ sort -k9 ls_root
total 173
drwxr-xr-x    2 root root  4096 Sep  7 15:39 bin
drwxr-xr-x    4 root root  5120 Sep 20 06:40 boot
drwxr-xr-x    3 root root  4096 Sep  2 11:44 data
... etc ...
```

- Sorting on a number field (add -n flag to use numerical value):

```
$ sort -k5 -n ls_root
dr-xr-xr-x   13 root root     0 Sep 12 16:38 sys
dr-xr-xr-x  242 root root     0 Sep 12 16:37 proc
total 173
lrwxrwxrwx    1 root root    30 Aug 30 06:55 vmlinuz.old ->
boot/vmlinuz-3.13.0-95-generic
lrwxrwxrwx    1 root root    30 Sep 20 06:39 vmlinuz ->
boot/vmlinuz-3.13.0-96-generic
... etc ...
```

- Can you find a way to sort on the Month field (the 6th column)?

Exercise 5 - Text-processing commands

- Make sure you are in your home directory. See if you still have iris.csv there; if not, download it again:

```
$ wget https://raw.githubusercontent.com/nycdatasci/bootcamp/master/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. Use `sort --help` to see if you can find a way to specify the separator.

OVERVIEW

- Operating Systems and 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
  - `wget or curl` - Download a web page

date

- Get the current date and time.

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

SSH (requires remote server)

- Log in to a remote machine

```
$ ssh <username>@<host_ip>
<username>@<host_ip>'s password:
Welcome to Ubuntu 16.04.4 LTS (GNU/Linux 4.4.0-1060-aws
x86_64)

... ...

 * Documentation:  https://help.ubuntu.com/
Last login: Thu Sep 17 16:53:57 2015 from ***.***.***.***
<username>@host_ip $
```

SCP (requires remote server)

- Use scp ("secure copy") to copy files to or from a remote machine.
- Both the two operations need to be done on your local machine: (why?)
  - Remote to Local

```
$ scp <username>@<host_ip>:~/iris.csv ~/Desktop/
<username>@<host_ip>'s password:
iris.csv                                  100% 4026     3.9KB/s   00:00
$ ls ~/Desktop/
```

  - Local to Remote

```
$ scp ~/Desktop/iris.csv <username>@<host_ip>:~/iris2.csv
<username>@<host_ip>'s password:
iris.csv                                  100% 4026     3.9KB/s   00:00
$
```

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` works very similar. Please see Exercise 2 (page 32)

```
$ curl -O "http://linuxcommand.org/lc3_learning_the_shell.php"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time
Current
                                 Dload  Upload   Total   Spent    Left
Speed
100  4984  100  4984    0     0  26060      0 --:--:-- --:--:-- --:--:--
26094
$ less curl1.html
    <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">

    <html lang="en">
... ...
```

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.