Week 1

File Permissions and Demo Using Notebook ...

We can issue commands directly to the OS using the terminal window, or using our notebook, or in our python code. Here's just a demo about file permissions as an example; compare these with a terminal window's behavior. You may at times see a `File not found` message but in fact it may be that your login doesn't have permissions to that file.

Compare the two commands ls and ls with the - (meaning a "switch") with options for that command F(ull) l(ist) a(lphaetical).

In [2]:
```
%ls
```

```
filepermissions1.png  week-01.key*          welcome.txt
filepermissions2.png  week-01.pdf
unix.ipynb            week-01.pptx
```

In [3]:
```
ls -Fla
```

```
total 35624
drwxr-xr-x   10 gb  staff      320 Dec 31 14:48 ./
drwxr-xr-x@ 204 gb  staff     6528 Dec 31 12:18 ../
drwxr-xr-x    3 gb  staff       96 Dec 31 14:04 .ipynb_checkpoints/
-rw-r--r--@   1 gb  staff    19877 Jan  9  2021 filepermissions1.png
-rw-r--r--@   1 gb  staff    18764 Jan  9  2021 filepermissions2.png
-rw-r--r--    1 gb  staff    66212 Dec 31 14:48 unix.ipynb
-rwx------@   1 gb  staff  8016045 Dec 31 13:38 week-01.key*
-rw-r--r--@   1 gb  staff  4594805 Dec 31 13:38 week-01.pdf
-rw-r--r--@   1 gb  staff  4756310 Dec 31 13:58 week-01.pptx
-rw-r--r--@   1 gb  staff      274 Dec 31 14:09 welcome.txt
```

In [4]:
```
ls -l welcome.txt
```

```
-rw-r--r--@ 1 gb  staff  274 Dec 31 14:09 welcome.txt
```

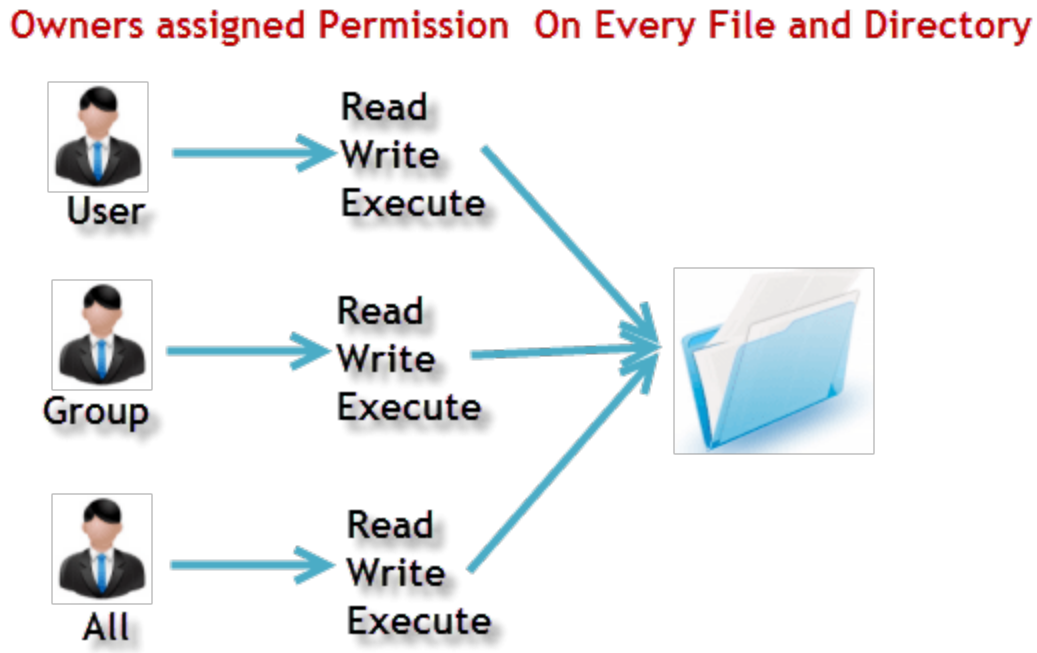
In [5]:
```
print(2 ** 0)
print(2 ** 1)
print(2 ** 2)
pow(2, 3)
```

```
1
2
4
```

Out[5]:
```
8
```

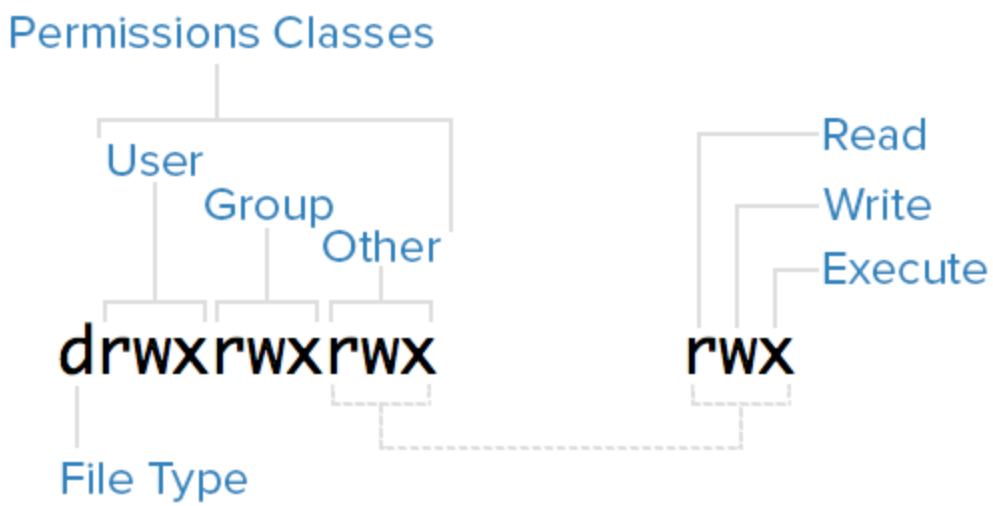
In [6]:
```
from IPython import display
display.Image("filepermissions1.png")
```

Out[6]:



In [7]:
```
display.Image("filepermissions2.png")
```

Out[7]:



Note that we can change file & folder permissions: each has an owner, can be part of a group, and can accessed by "the world" (or "other").

An *owner* is the person who creates the file and has control over the **read**, **write**, and **execute** permissions on that file/directory.

Usually a folder/file may be shared, say by everyone in the sales department, and so form a *group*. The owner can extend to others in the group the same permissions.

Finally, we may want to share a file with the *world* (or other), as we do with web pages.

Typically, our files by default may have permissions of **0644** for the owner to read/write, and others can only read the file. [Or **0755** (for webpages).]

If the file is a script or program, we may want to run (execute) the file - but not let others (like hackers) do so. Let's look at the file permissions and how we can issue commands to the operating system in both the terminal window and Jupyter Notebook.

owner	group	world
r | w | x	r | w | x	r | w | x
4 | 2 | 1	4 | 2 | 1	4 | 2 | 1
4 | **2** | 1	**4** | 2 | 1	**4** | 2 | 1
r | w | x	r | w | x	r | w | x
2^2 | 2^1 | 2^0	2^2 | 2^1 | 2^0	2^2 | 2^1 | 2^0

Some examples of communicating with Unix terminal window.

In [8]:
```
%ls -l welcome.txt
```

```
-rw-r--r--@ 1 gb  staff  274 Dec 31 14:09 welcome.txt
```

In [9]:
```
cat welcome.txt
```

```
Welcome to W200.
Students have a range of backgrounds - completely new to coding to many years
of experience.
In week 1 it's useful to get to know about backgrounds and identify any missi
ng fundamentals.  Enjoy the course.  Cheers

btw, the encoding of this file is UTF-8.
```

In [10]:
```
!date
```

```
Fri Dec 31 14:50:19 EST 2021
```

Communicating with the OS

To communicate between the OS and Notebook, use !. A similar command is % (or %%), called a "magic function." To invoke the Unix terminal use the magic function `%%bash`. For Windows, you could try `%%cmd`.

In [11]:
```
%%bash
head welcome.txt
date
```

```
Welcome to W200.
Students have a range of backgrounds - completely new to coding to many years
of experience.
In week 1 it's useful to get to know about backgrounds and identify any missi
ng fundamentals.  Enjoy the course.  Cheers

btw, the encoding of this file is UTF-8.Fri Dec 31 14:50:21 EST 2021
```

In [12]:
```
%%bash
clear  # clear the screen

history # show the list of previous commands (using the up/down arrow - very

date # shows current date and time
uptime # how long the system has been on...

whoami # identity crisis averted - username
id
who
```

```
Fri Dec 31 14:50:22 EST 2021
14:50  up  2:49, 3 users, load averages: 1.24 1.18 1.18
gb
uid=501(gb) gid=20(staff) groups=20(staff),12(everyone),61(localaccounts),79
(_appserverusr),80(admin),81(_appserveradm),98(_lpadmin),33(_appstore),100(_l
poperator),204(_developer),250(_analyticsusers),395(com.apple.access_ftp),102
(com.apple.access_screensharing-disabled),101(com.apple.access_ssh-disabled),
400(com.apple.access_remote_ae)
gb       console  Dec 31 12:01
gb       ttys000  Dec 31 14:01
gb       ttys001  Dec 31 14:10
```

In [13]:
```
%%bash
ls # "list show" - the files and directories in the cwd (current working dire
pwd  # "print" the working directory
```

```
filepermissions1.png
filepermissions2.png
unix.ipynb
week-01.key
week-01.pdf
week-01.pptx
welcome.txt
/Users/gb/Documents/UCB-DataSci/Sp22
```

These commands affect files - they're important to know. Remember that the space is a significant code - it separates the command from inputs (parameters and switches) to that command.

- `ls` - list show - all files & directories
- `cp` - copy files (from source to destination, e.g., `cp mya.txt myb.txt`
- `rm` - remove files and directories, e.g., rm myb.txt
- `rm` - rename or move (from ... to ...)
- `chmod` - change mode - changes file/directory access permissions
- `chown` - change the ownership of a file
- `cat` - concatenate a couple of files and shows in the stdout
- `head` - show the first lines of a test file, default is 10
- `tail` - show the bottom lines, 10 default
- `grep` - search for patterns in text files.
- `cd` - change directory; e.g., `cd ..` means go up to the parent directory

- `ln` - make links and symlinks to files/directories
- `mkdir` - make a new directory, e.g., mkdir w200homeworks
- `rmdir` - remove a directory, e.g., rmdir myfolder
- `sudo` - change temporarily to superuser (use with great care!); some shells use su instead of sudo
- `ps` - show the running processes (tasks, jobs) - the PID (process ID), input source (tty), time running, and the command used to start the process
- `top` - show tasks and system status
- `kill` - kill a process - but you need to know the process ID
- `>` - a "redirector" - to redirect the output from the stdout to something else, like a file. E.g., `ls > listOfMyFiles.txt`
- `|` - a "pipe" - a Unix communication channel - used often to take the results of one command and send them (pipe them) to another command

---

Some network commands

There is always only one active **stdin** (standard input, default is keyboard]), **stdout** (standard output, default is the monitor), and **stderr** (standard error, an error log file). We redirect these when reading/writing to files and when communicating over the net. Contemporary programming languages allow switching easy between local and network systems, by identifying the **protocol** or via some bridge. Compare accessing files using different protocols `ftp:` (file transfer protocol, very insecure), `sftp` (encrypted ftp), `wget` (download files from remote uservers using http/https or ftp, `ssh` (secure socket shell, enrypted remote access client), `http://`, `https://`, `file://` and so on. We can't review file transfers here but let's look at some network-related commands.

Below are some network commands used to reveal info about the Unix system: notice hostname will likely include "local" in the name (as opposed to an IP address that you'll see in the ip and netstat commands). `uname` prints the Unix system info - on a Mac you'll see "Darwin". When distributing code it's useful to determine the users' OS and install any dependencies (libraries) for your code.

In [1]:
```
%%bash
hostname
w
uname
```

```
MainMac.local
14:47  up  2:46, 3 users, load averages: 1.26 1.14 1.18
USER     TTY      FROM              LOGIN@  IDLE WHAT
gb       console  -                 12:01    2:45 -
gb       s000     -                 14:01      45 /Users/gb/opt/anaconda3/bin/
py
gb       s001     -                 14:10       6 -zsh a3/      ��    /bin/zsh
Darwin
```

The below commands are eye-opening about addresses - run on your own machine.

In [ ]:
```
!ip
!netstat

!ifconfig # show and set the IP address
```

Practice ...

If you're not comfortable using Unix terminal, practice making a file using cat or pico, check the files/directories in your current working directory; navigate up the directory hierarchy (e.g., from the current working directory, confirm (using pwd) and then issue cd .. and confirm the new directory. Get to know your working environment. Issue the `printenv` to see how your computer is set up; note the HOME and PATH variables.

Finally ...

Unix has a variety of flavors, such as **zsh** (z-shell, the current MacOS default), **csh**, **bash** (very common), **tsch** (t-shell) and so on ... each do the same job but some of the commands may vary, so it's useful to learn (a) to switch between shells and (b) use the `echo` command to extract data to send to/print/echo on the stdout.

What shell are you using? `echo $SHELL` . What does the prompt look like? For zsh you'll see %. Enter the command `bash` and see the prompt change to $.

Dec 31, 2021

In [ ]:
```
```