# A Short Introduction to Unix

## Unix Runs the Internet

- Unix is a command line interface, used by most large, powerful computers.
- In fact, Unix is the underlying structure for most of the Internet and most large scale scientific computation.
- A knowledge of Unix is likely to be helpful in your future career, regardless of where you pursue it.

## Unix Advantages

- It is very popular, so it is easy to find information and get help
  - pick up books at the local bookstore (or street vendor)
  - plenty of helpful websites
  - USENET discussions and e-mail lists
  - most Comp. Sci. students know Unix
- Unix can run on virtually any computer

  (IBM, Sun, Compaq, Mac, etc)
- Unix is free or nearly (always) free
  - Linux/open source software movement
  - Ubuntu, FreeBSD, MKLinux, LinuxPPC, etc.

## Stable and Efficient

- Unix is very stable - computers running Unix almost never crash
- Unix is very efficient
  - it gets maximum number crunching power out of your processor (and multiple processors)
  - it can smoothly manage extremely huge amounts of data
  - it can give a new life to otherwise obsolete Macs and PCs
- Much free software is created for Unix - it’s accessible to all programmers

## Unix has some Drawbacks

- Unix computers are controlled by a command line interface
  - **NOT** user-friendly
  - ....(but fortunately, not user-*antagonistic* either!)
  - difficult to learn, even more difficult to truly master
- There are many different versions of Unix with subtle differences

## Computer Hardware is not Free

- However, you can build a powerful Linux cluster for $20-50K

  (depending on how much power you need)
- The real cost is for a person to manage the machines, install the software, and train scientists to use it.
- Small schools can join together or affiliate with a larger neighbor.

## Logging in to the CS Server

- Open a SSH program from your computer
- Connect to: `csgateway.clarku.edu`
- Type your username and password
  - You can't backspace/delete while typing username and password
  - Notice that when you type a password, nothing shows up on the screen, this is for your security
- Can do this from anywhere!
- Software:
  - Mac: Directly from the Terminal app.
  - Windows: PuTTY, WinSCP, other SSH clients

## General Unix Tips

- **UNIX is case sensitive!!**
  - **myfile.txt** and **MyFile.txt** do **not** mean the same thing
  - Some like to use capital letters for directory names - it puts them at the top of an alphabetical listing
- Every program is independent
  - the core operating system (known as the ***kernel***) manages each program as a distinct process with its own little chunk of dedicated memory.
  - If one program runs into trouble, it dies, but does not affect the kernel or the other programs running on the computer.

## The Unix Shell

- You communicate with a Unix computer through a command program known as a ***shell***.
- The shell interprets the commands that you type on the keyboard.
- There are actually many different shells available for Unix computers, and on some systems you can choose the shell in which you wish to work.
- You can use shell commands to write simple programs (scripts) to automate many tasks

## Unix Commands

- Unix commands are short and cryptic like ***cp*** or ***rm***.
  - Computer geeks like it that way; you will get used to it.
- Every command has a host of modifiers which are generally single letters preceded by a hyphen:

  ***ls -l*** or ***rm -R***
  - Capital letters have different functions than small letters, often completely unrelated.
  - A command also generally requires an argument, meaning some file on which it will act:

    ***ls -l my_dir***

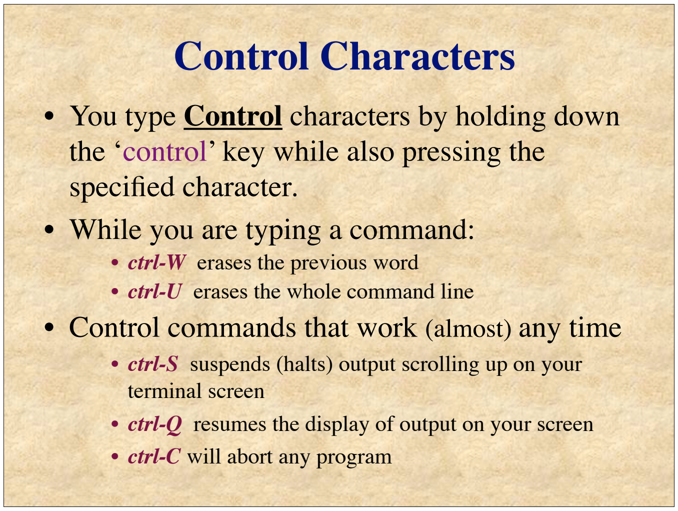
## Wildcards

- You can substitute the * as a wildcard symbol for any number of characters in any filename.
- If you type just * after a command, it stands for all files in the current directory:

  *lpr ** will print all files
- You can mix the * with other characters to form a search pattern:

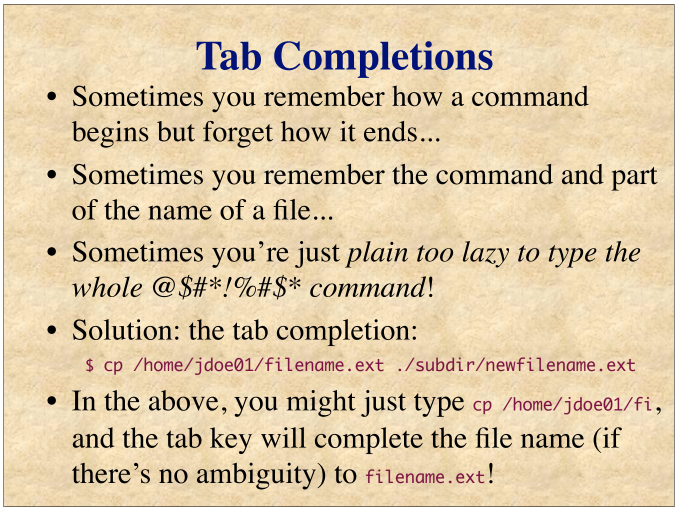
  *ls a*.txt* will list all files that start with "a" and end in ".txt"
- The "**?**" wildcard stands for any single character:

  *cp draft?.doc* will copy *draft1.doc*, *draft2.doc*, *draftb.doc*, etc.



## Control Characters

- You type **Control** characters by holding down the ‘control’ key while also pressing the specified character.
- While you are typing a command:
  - *ctrl-W* erases the previous word
  - *ctrl-U* erases the whole command line
- Control commands that work (almost) any time
  - *ctrl-S* suspends (halts) output scrolling up on your terminal screen
  - *ctrl-Q* resumes the display of output on your screen
  - *ctrl-C* will abort any program

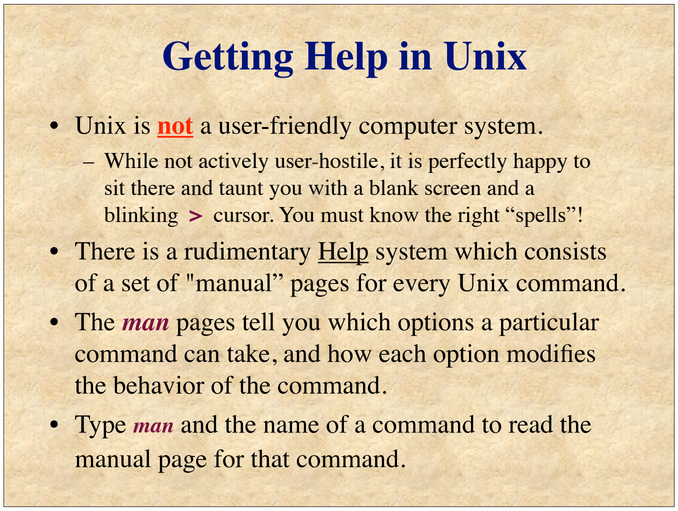


## Tab Completions

- Sometimes you remember how a command begins but forget how it ends...
- Sometimes you remember the command and part of the name of a file...
- Sometimes you’re just *plain too lazy to type the whole @$#*!%#$* command*!
- Solution: the tab completion:

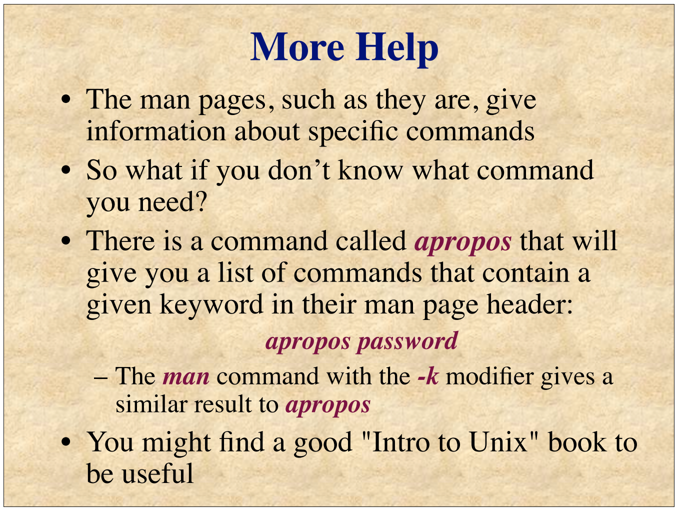
```
$ cp /home/jdoe01/filename.ext ./subdir/newfilename.ext
```

- In the above, you might just type `cp /home/jdoe01/fi`, and the tab key will complete the file name (if there’s no ambiguity) to `filename.ext`!



## Getting Help in Unix

- Unix is **not** a user-friendly computer system.
  - While not actively user-hostile, it is perfectly happy to sit there and taunt you with a blank screen and a blinking **>** cursor. You must know the right “spells”!
- There is a rudimentary Help system which consists of a set of "manual” pages for every Unix command.
- The ***man*** pages tell you which options a particular command can take, and how each option modifies the behavior of the command.
- Type ***man*** and the name of a command to read the manual page for that command.

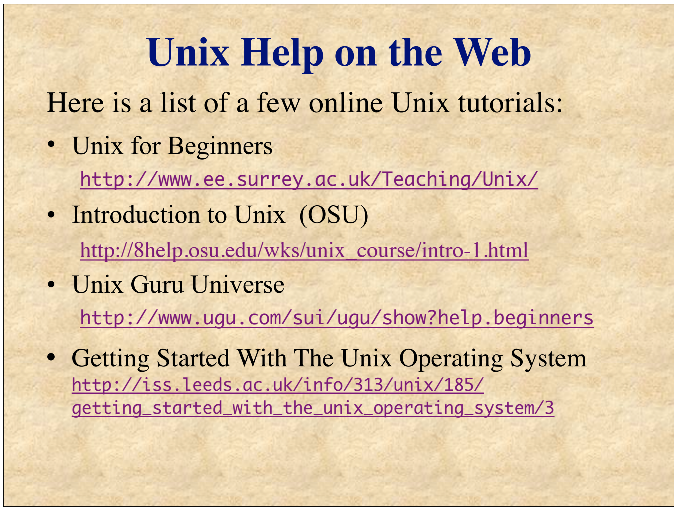


## More Help

- The man pages, such as they are, give information about specific commands
- So what if you don’t know what command you need?
- There is a command called ***apropos*** that will give you a list of commands that contain a given keyword in their man page header:

***apropos password***

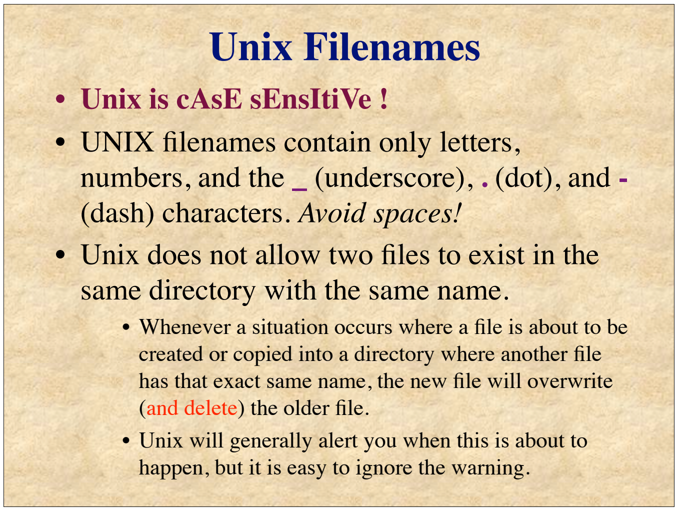
  - The ***man*** command with the ***-k*** modifier gives a similar result to ***apropos***
- You might find a good "Intro to Unix" book to be useful



## Unix Help on the Web

Here is a list of a few online Unix tutorials:

- Unix for Beginners
  http://www.ee.surrey.ac.uk/Teaching/Unix/
- Introduction to Unix (OSU)
  http://8help.osu.edu/wks/unix_course/intro-1.html
- Unix Guru Universe
  http://www.ugu.com/sui/ugu/show?help.beginners
- Getting Started With The Unix Operating System
  http://iss.leeds.ac.uk/info/313/unix/185/getting_started_with_the_unix_operating_system/3



## Unix Filenames

- **Unix is cAsE sEnsItiVe !**
- UNIX filenames contain only letters, numbers, and the **_** (underscore), **.** (dot), and **-** (dash) characters. *Avoid spaces!*
- Unix does not allow two files to exist in the same directory with the same name.
  - Whenever a situation occurs where a file is about to be created or copied into a directory where another file has that exact same name, the new file will overwrite (and delete) the older file.
  - Unix will generally alert you when this is about to happen, but it is easy to ignore the warning.

# Filename Extensions

- Most UNIX filenames start with a lower case letter and end with a dot followed by one, two, or three letters: ***myfile.txt***
  - However, this is just a common convention and is not required.
  - It is also possible to have additional dots in the filename.
- The part of the name following the dot is called the "extension."
- The extension is often used to designate the type of file: ***myprogram.pl***

# Some Common Extensions

- By convention:
  - files that end in ***.txt*** are text files
  - files that end in ***.c*** are source code in the "C" language
  - files that end in ***.html*** are HTML files for the Web
  - Compressed files have the ***.zip*** or ***.gz*** extension
- Unix **does not require** these extensions (unlike Windows), but it is a sensible idea and one that you should follow

# Working with Directories

- Directories are a means of organizing your files on a Unix computer.
  - They are equivalent to folders in Windows and Mac computers
- Directories contain files, executable programs, and sub-directories
- Understanding how to use directories is crucial to manipulating your files on the Unix system.

# Typical UNIX directory structure

/

pronounced 'slash' or 'root'.

/bin = where the programs live. Hands off!

/lib = programming libraries. Ignore

/etc = admin stuff. Ignore.

/usr = more programs, not user files. Hands off!

/mnt = 'mount point' for floppies, cd roms etc.
If you put a cd rom in, it is in /mnt/cdrom

/tmp = temporary files. Ignore.

/var = more temporary files. Ignore.

/home
/home/fred where ALL my files are.
/home/george where George's files are.
/home/ginny where Ginny's files are.
(I can't see them unless she lets me.)

A UNIX workstation is usually set up like this; Windows and MacOSX are different (although Windows is much more different than Mac).

# Your Home Directory

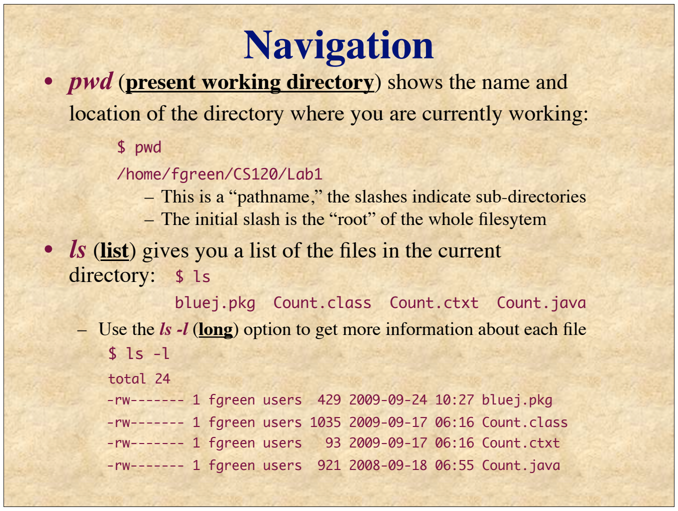
- When you login to the file server ("younger", or "csgateway"), you always start in your **Home** directory.
- Create sub-directories to store specific projects or groups of information, just as you would place folders in a filing cabinet.
- Do **not** accumulate thousands of files with cryptic names in your Home directory

# File & Directory Commands

- This is a minimal list of Unix commands that you must know for file management:

```
ls (list)        mkdir (make directory)
cd (change directory)  rmdir  (remove directory)
cp (copy)        pwd (present working directory)
mv (move)        more (view by page)
rm (remove)       cat (view entire file on screen)
```

- All of these commands can be modified with many options. Learn to use Unix '***man***' pages for more information.



## Navigation

- ***pwd*** (**<u>present working directory</u>**) shows the name and location of the directory where you are currently working:

```
$ pwd
/home/fgreen/CS120/Lab1
```

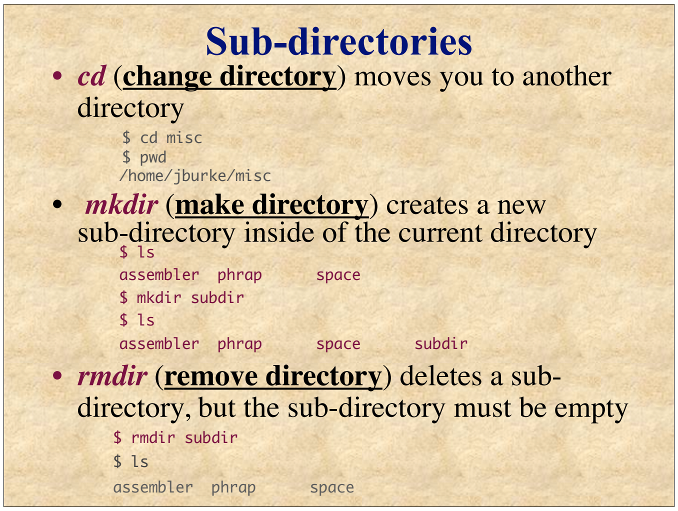
  - This is a "pathname," the slashes indicate sub-directories
  - The initial slash is the "root" of the whole filesytem

- ***ls*** (**<u>list</u>**) gives you a list of the files in the current directory: `$ ls`

```
          bluej.pkg  Count.class  Count.ctxt  Count.java
```

  - Use the ***ls -l*** (**<u>long</u>**) option to get more information about each file

```
$ ls -l
total 24
-rw------- 1 fgreen users  429 2009-09-24 10:27 bluej.pkg
-rw------- 1 fgreen users 1035 2009-09-17 06:16 Count.class
-rw------- 1 fgreen users   93 2009-09-17 06:16 Count.ctxt
-rw------- 1 fgreen users  921 2008-09-18 06:55 Count.java
```



## Sub-directories

- ***cd*** (**<u>change directory</u>**) moves you to another directory

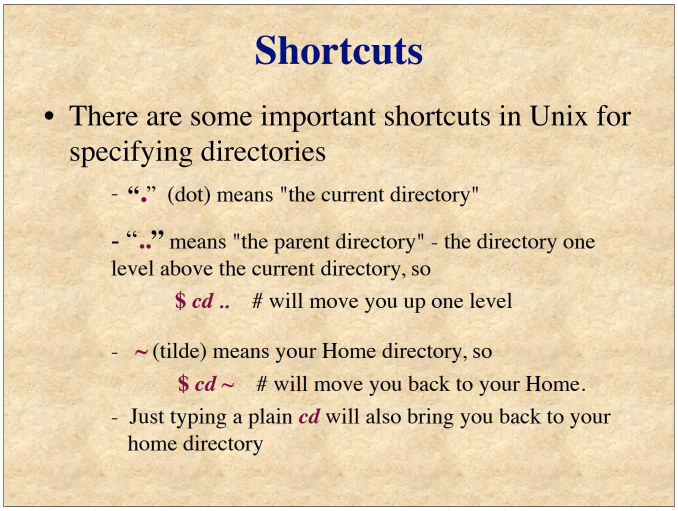
```
$ cd misc
$ pwd
/home/jburke/misc
```

- ***mkdir*** (**<u>make directory</u>**) creates a new sub-directory inside of the current directory

```
$ ls
assembler  phrap      space
$ mkdir subdir
$ ls
assembler  phrap      space      subdir
```

- ***rmdir*** (**<u>remove directory</u>**) deletes a sub-directory, but the sub-directory must be empty

```
$ rmdir subdir
$ ls
assembler  phrap      space
```



## Shortcuts

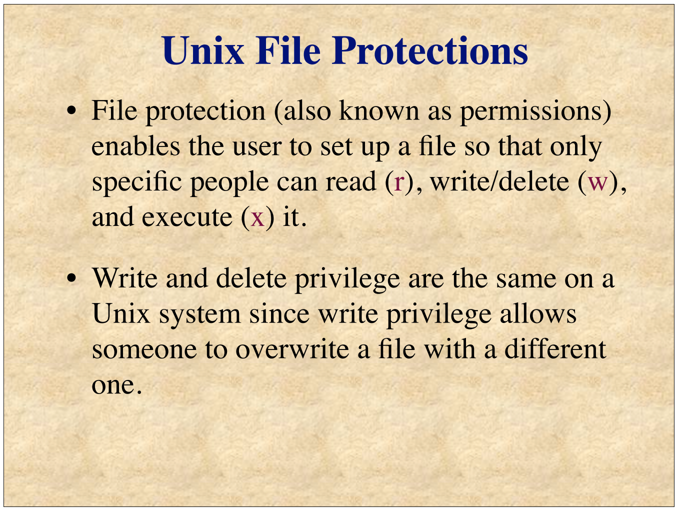
- There are some important shortcuts in Unix for specifying directories
  - "**.**" (dot) means "the current directory"
  - "**..**" means "the parent directory" - the directory one level above the current directory, so

    ***$ cd ..*** # will move you up one level

  - **~** (tilde) means your Home directory, so

    ***$ cd ~*** # will move you back to your Home.

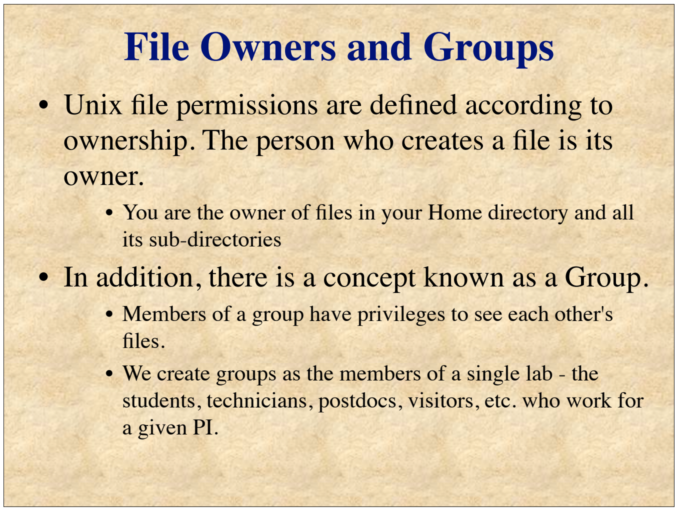
  - Just typing a plain ***cd*** will also bring you back to your home directory



## Unix File Protections

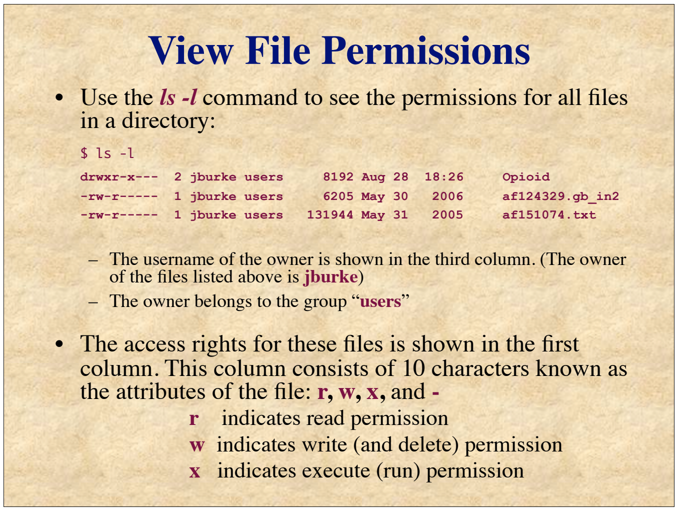
- File protection (also known as permissions) enables the user to set up a file so that only specific people can read (r), write/delete (w), and execute (x) it.

- Write and delete privilege are the same on a Unix system since write privilege allows someone to overwrite a file with a different one.



## File Owners and Groups

- Unix file permissions are defined according to ownership. The person who creates a file is its owner.
  - You are the owner of files in your Home directory and all its sub-directories
- In addition, there is a concept known as a Group.
  - Members of a group have privileges to see each other's files.
  - We create groups as the members of a single lab - the students, technicians, postdocs, visitors, etc. who work for a given PI.



## View File Permissions

- Use the ***ls -l*** command to see the permissions for all files in a directory:

```
$ ls -l
drwxr-x---  2 jburke users     8192 Aug 28  18:26     Opioid
-rw-r-----  1 jburke users     6205 May 30   2006     af124329.gb_in2
-rw-r-----  1 jburke users   131944 May 31   2005     af151074.txt
```

  - The username of the owner is shown in the third column. (The owner of the files listed above is **jburke**)
  - The owner belongs to the group "**users**"

- The access rights for these files is shown in the first column. This column consists of 10 characters known as the attributes of the file: **r**, **w**, **x**, and **-**

  **r** indicates read permission
  **w** indicates write (and delete) permission
  **x** indicates execute (run) permission

```
$ ls -l
drwxr-x---  2 jburke users     8192 Aug 28  18:26     Opioid
-rw-r-----  1 jburke users     6205 May 30   2006     af124329.gb_in2
-rw-r-----  1 jburke users   131944 May 31   2005     af151074.txt
```

- The first character in the attribute string indicates if a file is a directory (d) or a regular file (**-**).
- The next 3 characters (rwx) give the file permissions for the owner of the file.
- The middle 3 characters give the permissions for other members of the owner's group.
- The last 3 characters give the permissions for everyone else (others).
- The default protections assigned to new files on our system is: -rw-r----- (owner=read and write, group =read, others=nothing)

# Change Protections

- Only the owner of a file can change its protections
- To change the protections on a file use the ***chmod*** (**<u>change mode</u>**) command.

  [Beware, this is a confusing command.]

  - First you have to decide for whom you will change the access permissions:
    - » **the file owner (u)**
    - » **the members of your group (g)**
    - » **others (o) (ie. anyone with an account)**
  - Next you have to decide if you are adding (**+**), removing (**-**), or setting (**=**) permissions.
- Taken all together, it looks like this:

```
$ chmod   u=rwx o+rx    myfile.html
```

This will set the owner to have read, write, and execute permission; and add the permission for others to read and execute the file named **myfile.html**.

# Commands for Files

- Files are used to store information, for example, data or the results of some analysis.
  - You will mostly deal with text files
  - Files on the server are automatically backed up every night.
- *cat* dumps the entire contents of a file onto the screen.
  - For a long file this can be annoying, but it can also be helpful if you want to copy and paste (use the buffer of your SSH program).

# *more*

- Use the command ***more*** to view at the contents of a file one screen at a time:

```
$ more t27054_cel.pep
!!AA_SEQUENCE 1.0
P1;T27054 - hypothetical protein Y49E10.20 - Caenorhabditis elegans
Length: 534  May 30, 2000 13:49  Type: P  Check: 1278  ..
1  MLKKAPCLFG SAIILGLLLA AAGVLLLIGI PIDRIVNRQV IDQDFLGYTR
        51  DENGTEVPNA MTKSWLKPLY AMQLNIWMFN VTNVDGILKR HEKPNLHEIG
101  PFVFDEVQEK VYHRFADNDT RVFYKNQKLY HFNKNASCPT CHLDMKVTIP
t27054_cel.pep (87%)
```

- – Hit the spacebar to page down through the file
- – **Ctrl-U** moves back up a page
- – At the bottom of the screen, ***more*** shows how much of the file has been displayed

# Copy & Move

- ***cp*** lets you **<u>copy</u>** a file from any directory to any other directory, or create a copy of a file with a new name in one directory
  - `cp  filename.ext   newfilename.ext`
  - `cp  filename.ext   subdir/newname.ext`
  - `cp  /home/jdoe01/filename.ext   ./subdir/newfilename.ext`
- ***mv*** allows you to **<u>move</u>** files to other directories, but it is also used to rename files.
  - Filename and directory syntax for ***mv*** is exactly the same as for the ***cp*** command.
    - `mv filename.ext   subdir/newfilename.ext`
  - **NOTE:** When you use ***mv*** to move a file into another directory, the current file is deleted.

# Delete

- Use the command ***rm*** (**<u>remove)</u>** to delete files
- There is no way to undo this command!!!
  - We have set the machnie to ask if you really want to remove each file before it is deleted.
  - You must answer “Y” or else the file is not deleted.

```
$ ls
af151074.gb_pr5   test.seq
$ rm test.seq
rm: remove test.seq? y
$ ls
af151074.gb_pr5
```

# Moving Files between Computers

- You will often need to move files between computers - desktop to server and back
- There are several options
  - Memory stick (flash memory)
  - E-mail
  - Network filesharing
  - FTP, SFTP, SSH

# FTP is Simple

- **F**ile **T**ransfer **P**rotocol is standard for all computers on any network. (We use SFTP: “S” for “Secure”.)
- The best way to move lots of data to and from remote machines:
  - put raw data onto the server for analysis
  - get results back to the desktop for use in papers and grants
- Graphical FTP applications:
  - In MacOS X: Fugu, FileZilla (or, for adventurous souls, “sftp” at command line!), free; Transmit, Fetch, modestly priced
  - In Windows: WinSCP, FileZilla, PSFTP (companion to “PuTTY”), all free