Introduction to Unix Tutorial

Topics covered in this Tutorial

1. CSIF Computer Network
2. Local Logging in.
3. Remote computer access: **`ssh`**
4. Navigating the UNIX file structure: **`cd, ls,`** and **`pwd`**
5. Making and removing directories: **`mkdir`**, **`rmdir`**
6. Copying, moving, and removing files: **`cp`**, **`mv`**, and **`rm`**
7. Command Line Editing
8. The Terminate Key: **`^C`**
9. Displaying Files: **`cat`**, **`more`**, and **`head`**.
10. Transfering files: **`sftp`**
11. File submission: **`handin`**
12. Printing Files: **`lpr`**, **`lpq, lprm,`** and **`enscript`**.
13. Text Editors: **`vi`**, **`emacs`**, and **`gedit.`**
14. Help command: **`man`**.
15. Logging out: **`exit`**

The following conventions will be used:

Instructions will be in: this normal typeface and *this italics typeface.*

Things which you will type will be in: **`this bold typeface.`**

CSIF Computer Network

Students taking computer science classes work on the machines run by CSIF (Computer Science Instructional Facility), located in the basement labs of Kemper. If you are enrolled in a class this quarter, you will automatically have an account on this system (see below, "Local Logging in"). Currently the bulk of the system consists of about eighty PC (Intel) workstations, named pc1 to pc83.

Local Logging In

If you are taking a computer science class, you will automatically get an account on the CSIF machines. This means you can log in to any of the machines in the basement labs. To access an account, you will need two things: 1) your Username and 2) your Password. Your user name is your UCD Kerberos username, and your password is your kerebos password.

Type your login name, then your password. Do this now. If all goes well, the monitor will display the UNIX graphic user interface (GUI) of the computer for you.

The PCs do not start with a terminal window. To create a terminal window on a PC, click on activites at the left end of the toolbar, and then click on the Search Bar. Type in either "terminal" or "konsole" and select on the application. For the rest of this tutorial will only be typing in one of these terminal windows.

Once you have created a terminal window, you will see a “command line prompt” indicating that Unix is ready to accept your instructions. This prompt will usually have your username, computer name, and end with a "$", but may be terminated by a "%". The prompt is generated by a program known as a *shell*. This is simply a program that accepts your commands and gets UNIX to execute them.

Remote Computer Access from Home

If you are in the CSIF, then please skip to the following section, “Using ssh in the CSIF”. If you are at home, then we need you to connect to the CSIF before you continue. The first step is determining the name of CSIF computer with which to connect. The CSIF computers are named pc1 to pc68, with about five missing. You may simply choose any number in that range, say 34. Then the computer name would be pc34.cs.ucdavis.edu. For this class, choose one between pc33 - pc60.

You can remotely log in to a CSIF machine from some other computer by using secure telnet software. The ssh software is already installed on a Mac, but must be download for a Windows computer. If you are on Windows machine please skip down to the topic “ssh on a Windows computer.”

ssh on a Mac

For Macs, there is a built-in UNIX terminal like on the CSIF computers. If you are on a Mac you will need to Open your root volume(usually called Macintosh HD), then Applications, then Utilities, and then Terminal. If you are on a Mac, then open a Terminal now.

The syntax for secure shell is: ssh *computer_name* So in our example you would type: ssh pc34.cs.ucdavis.edu.
If you have never connected to the computer before, ssh will ask if you are sure of the remote computer's authenticity:
The authenticity of host 'pc34 (169.237.5.14)' can't be established.
RSA key fingerprint is f0:ad:72:d2:2e:84:ab:53:2e:06:60:7d:3e:38:17:3c.
Are you sure you want to continue connecting (yes/no)?
In such a case, just type "yes".

Once the authenticity of the remote computer is established, ssh will prompt you for your username and password on the remote computer.

Now you are ready to make a real ssh connection.
Type: **ssh *computer_name,*** where ***computer_name*** is a pc computer that you chose.

Once you have entered your Kerebos name and password, you should now see a prompt that reflects your username and the remote computer, e.g. [ssdavis@pc34 ~]. Please skip down to the next topic: “Using ssh in the CSIF.”
For grahics forwarding, use: ssh pc34.cs.ucdavis.edu -X

ssh on a Windows computer

Since Windows does not come with ssh installed, you will need to download an ssh program. For this class, will be using MobaXTerm, available for download at:
http://download.cnet.com/MobaXterm/3000-7240_4-10890137.html

Once you have downloaded sofware, you will need to choose a CSIF with which to connect. You must randomly choose a computer name between pc33 and pc60. The following example assumes you chose pc34
Start MobaXterm.
The syntax for secure shell is: ssh computer_name So in our example you would type: ssh pc34.cs.ucdavis.edu.
If you have never connected to the computer before, ssh will ask if you are sure of the remote computer's authenticity:
The authenticity of host 'pc34 (169.237.5.14)' can't be established.
RSA key fingerprint is f0:ad:72:d2:2e:84:ab:53:2e:06:60:7d:3e:38:17:3c.
Are you sure you want to continue connecting (yes/no)?
In such a case, just type "yes".

Once the authenticity of the remote computer is established, ssh will prompt you for your username and password on the remote computer.

Now you are ready to make a real ssh connection.
Type: **ssh computer_name**, where **computer_name** is a pc computer that you chose.

Once you have entered your Kerebos name and password, you should now see a prompt that reflects your username and the remote computer, e.g. [ssdavis@pc34 ~].
For graphics forwarding, use ssh pc34.cs.ucdavis.edu -X

Using ssh in the CSIF

The syntax for secure shell is: ssh *computer_name* When you are already logged into a CSIF computer (as everyone should be at this point in the tutorial), you should now type the command **ruptime**. This results in a printout of

all the hostnames (computers) in the network, which ones are up and down, and the number of users for each host that is up. Look through the list, and choose a computer that is "up," and has few users on it. We will assume you chose pc27 for this example. We will walk through the steps of using ssh before having you actually issue the ssh command.

When specifying a computer name for ssh, you may omit the remote computer's domain (cs.ucdavis.edu) if your local computer is in its domain, i.e. in the CSIF, so in this example you could type: "ssh pc27" rather than "ssh pc27.cs.ucdavis.edu".

If you have never connected to the computer before, ssh will ask if you are sure of the remote computer's authenticity:

The authenticity of host 'pc27 (169.237.5.98)' can't be established.
RSA key fingerprint is a6:73:a3:d5:97:ae:e3:f5:65:8c:5b:a6:37:8c:bb:79.
Are you sure you want to continue connecting (yes/no)?

In such a case, just would type "yes".

Once the authenticity of the remote computer is established, ssh will prompt you for password on the remote computer. Note since you have already logged into a CSIF computer to get to this point, ssh will not ask for your username.

Now you are ready to make a real ssh connection.

Type: **ssh *computer_name,*** where ***computer_name*** is a pc computer that you chose from the ruptime list.

Once you have entered your password, please make sure that you are logged into a different computer from the original CSIF you started with by looking at the prompt in the terminal window, e.g "[ssdavis@pc27 ~]$". For the next few exercises you will use this remote computer.

Navigating the UNIX file system: pwd, cd, ls

Each directory in the file system has a *path name* that uniquely specifies the location of that directory within the computer's file structure. As you move within the file structure, you can use the command pwd to display the path name of current location. pwd stands for "print working directory". Type **`pwd`** now.

The shell should display */home/* followed by your username, e.g. */home/ssdavis*. This is your *home directory*. This is the place where you "live" in the Unix file system. Each user has their own *home directory* that is completely under their control.

The cd change directory) command allows you to move to any other public directory in the file system. The syntax for cd is: cd *path_name*. Now change to the */bin* directory by typing **`cd /bin`**.

After executing the directory change, use **`pwd`** to see that you really are in the */bin* directory.

The ls (list) command allows you to see the contents of a directory. The syntax for ls is: ls [-al] [*filename*] , where the "[ ]" indicate optional values. Now simply type **`ls`**.

You will see a list of the names of the files in the */bin*. The */bin* directory holds many of the basic UNIX utility programs. Note that "ls" is listed, but not "cd" or "pwd". This is because "ls" is an independent program, but "cd" and "pwd" are commands built into the shell program.

Before you look into the options for ls, I want you back in your home directory. You could type *cd /home/* followed by your username, e.g. *cd /home/ssdavis*, but there is a short cut. Simply type **`cd`**

After the shell has executed the change directory command, type **`pwd`**. You can see that simply typing "cd" without a path name will automatically return you to your home directory.

Now type **`ls`**.

If you have not created any files or directories in your account, then nothing will be displayed.

Many shell commands have *options* that may follow the command after a space and a dash. The options change the behavior of the command. For ls, the options change what you see -- giving you specialized types of information which you wouldn't see using the basic command.

Now type: **`ls -a`**. The **`-a`** option stands for "all". You should see something like this:

```
.                .cshrc          .login          .pine-debug2  .wmrc
..               .cshrc.COPY4U   .login.COPY4U   .pine-debug3  .xmotd
.DECkeymaprc     .dt             .mailrc         .pine-debug4  .xsession-errors
.FVWM2-errors    .dtprofile      .mh_profile     .pinerc
.SGIkeymaprc     .emacs          .msgsrc         .rhosts
.Xauthority      .emacs~         .netscape       .sh_history
.addressbook     .f              .newsrc         .ssh2
.addressbook.lu  .forward        .oldnewsrc      .tin
.bash_history    .history        .pine-debug1    .wm_style
```

Where did all these files come from? Note that all of them start with a period. UNIX hides system files that are normally not needed by the user by starting them with a period. There are two unusual files listed, "**.**", and "**..**". These have special meanings in path names. "**.**" stands for the current directory.

Now type **cd .** (with just one period after cd).

Now type **pwd**. You haven't moved within the directory structure!

"**..**" stands for the parent directory of the current directory. Now type **cd ..** (with two periods after cd).

Now type **pwd**. You should see "*/home*".

Now return to your home directory by simply typing **cd**

Now type **ls –al** The **–l** option stands for long. The shell will display a line of information for each file. Here is an excerpt from my home directory:

```
drwxr-xr-x   27 ssdavis    users        4096 Jan  4 21:29 .
drwxr-xr-x  4185 root    users      147456 Jan  7 07:56 ..
-rw-r--r--    1 ssdavis    users        1648 Oct  5  1993 .DECkeymaprc
-rw-------    1 ssdavis    users           0 Apr  3  2000 .ICEauthority
-rw-r--r--    1 ssdavis    users         698 Oct  5  1993 .SGIkeymaprc
-rwxr--r--    1 ssdavis    users        2481 Oct  5  1993 .X11Startup
-rw-r--r--    1 ssdavis    users        2285 Feb 26  1997 .addressbook.lu
-rwx------    1 ssdavis    users        2546 Oct 13 10:30 .cshrc
-rw-------    1 ssdavis    users        1869 Sep 25  1996 .cshrc.COPY4U
-rwx------    1 ssdavis    bin          1864 Mar 26  1998 .cshrc.OLD
```

Here is a brief introduction to the information displayed in the long format.

- File permissions and file types (leftmost field)

  `drwxr-xr-x`

  The `d` at the very left says the file is a *directory*.
  The next letters `rwx` say that the owner (myself) has *read*, *write* and *execute* permissions for the file. For a directory, execute permission means I can cd to that directory. Read permission means I can use the command `ls` to see the contents of the directory. Write permission means I can create additional files *in* that directory.
  The next two groups of three show that other users (*group* and *world*) are only allowed read and execute permissions.

  `-rwxr--r--`

  The dash at the very left indicates an *ordinary file*, as opposed to a directory. Here the read, write and execute permissions mean the ability to look at the file (using commands like `more`, see below), to *edit* the file (using a text editor, see below), or *execute* the file (if it is an executable program -- otherwise this permission doesn't mean much).

  `-rw-r--r--`

  This set of permissions shows an ordinary file which can only be read and edited by the owner, and only read by everyone else.

- Links field (second from left): number of short cuts to this file from other directories.
- Owner field (third from left)
- Group field (fourth from left)
- Size field (fifth): shows the file size in bytes
- Date field: when the file was last modified
- Name fields (last).

Notice there were no pathnames after any of the ls commands. That means that you saw the contents of the current working directory. If you do put a pathname afterwards, then you will see the contents of the directory identified by that pathname. Now type **`ls /`**

In this example, the pathname you gave ls was the *root* directory (indicated by a sole backslash), which is the top directory in the Unix file system. All the files you see listed there are subdirectories of the root. Notice the subdirectory *home*. That is the subdirectory in which the home directories of everyone having an account are located.

Creating and Deleting Directories : mkdir, rmdir

To create a directory use the mkdir command. It's syntax is mkdir {*directory_name}*+. The "{}+" indicate that there must be at least one directory name given.

Now type **`mkdir first`**

Now type **`ls -l`** You should have a directory named *first* now in your file list.

Now type **`cd first`** to move into the *first* directory.

Use **`pwd`** to see where you are.

Now type **`mkdir second third`**.

Now type **`ls –a`** . The display should be:

```
.  ..  second  third
```

Now type **`ls -l`**. The display should be like:

```
total 8
drwxr-xr-x    2 ssdavis    users        4096 Jan  7 22:29 second
drwxr-xr-x    2 ssdavis    users        4096 Jan  7 22:29 third
```

The syntax for the remove directory command is rmdir {*directory_name}*+.
Now type **`rmdir second`** to remove the directory named *second.*
Use **`ls`** to check your work.
You should see only *third* listed.

***

Copying, Moving, and Deleting Files : cp, mv, rm

Use **`pwd`** to make sure that you are still in */home/user_name/first***`.`** There are two syntaxes for the copy file command: 1) cp *source_file_name destination_file_name*, and 2) cp {*source_file_name*}+ *destination_directory* Now you will copy a file from user ssdavis' home directory into the current directory.

Type **`cp ~yourName/add.c myfile.c`**

Use **`ls`** to make sure that you were successful. You should see only *myfile.c*, and *third* listed.

Now copy this file down into the *third* directory by typing **`cp myfile.c third`**

You could check your work by cding into the *third* directory and then using ls, but there is simpler way. Just type **`ls third`** to display the contents of the *third* directory. You should see only *myfile.c* listed.

The **`mv`** command has three syntaxes: 1) mv *old_file_name new_file_name*, to change the name of a file, 2) mv {*filename}*+ *directory_name*, to move file(s) into a directory, and 3) mv *old_directory_name new_directory_name* to change the name of a directory. You will try each one in order.

Now type **`mv myfile.c wombat`** to rename *myfile.c* to *wombat*.

Use **`ls`** to check your work. You should see only *third* and *wombat* listed.

Now type **`mv third/myfile.c .`** (note the period indicates the current directory) to move *myfile.c* to the current directory.

Again use **`ls`** to check your work. You should now have *myfile.c*, *third*, and *wombat* in the current directory.

Now type **`mv third fourth`**, to rename the *third* directory to *fourth*.

Use **`ls`** to check your work. You should now see *fourth*, *myfile.c*, and *wombat* listed.

The syntax of the command to remove files is rm {*file_name*}+.

Now type **`rm myfile.c`**

Use **`ls`** check your work.

Your current directory, *first*, should now contain only one directory, *fourth*, and one file, *wombat*.

Shell Command Line Editing

Our UCD CSIF system accounts are set up to use the tcsh extension of the popular shell called the C-shell. A nice feature of tcsh is *command line editing*. Here is how it works:

Let's say you wish to copy *wombat* into the *fourth* directory, but you type **`cp woombat forth/snake`**, and, BEFORE YOU HIT THE RETURN KEY, you notice your errors. Then you can go back to make the appropriate changes by using the left-arrow key to go to the `o' in *forth* and the typing 'u'; then use the left-arrow key to an 'o' in *woombat* and hit the Delete key, which will remove the extra 'o'; and then hit the return key to process the command. Try it now.

Type **`ls fourth`** to check your work. You should see only snake listed.

On the other hand, if you have already hit the return key when you notice your typing error, you can use ctrl-p (or up-arrow) to go back to my previous command. (Use ctrl-p to go back through several previous commands, and ctrl-n or down-arrow to go forward.) Note that you can also use this to repeat (without change) a previous command.

The tcsh (and some other shells) also allows you to do *file name completion*, again a great saver of typing and time. Let's say you wish to move *wombat* into the *fourth* directory. The command would be "mv wombat fourth".

But rather than typing the whole thing, you can type **`mv w`** then hit the Tab key.

Since *wombat* is the only file in the current directory that starts with "w", the shell would complete the file name (wombat) for you, you would then just type **`f`** and again hit the Tab key.

Since *fourth* is the only file that begins with an f the shell will complete the file name for your again, giving you "mv wombat fourth/" on the command line!

Hit the then Enter key, and again type **`ls fourth`**, and you should see only *snake* and *wombat*.

The shell will complete a file name as long as there is only one file with a name that matches what you have typed so far. This means that you may have to type several letters at the beginning of a file name if there are other files with identical beginnings of their names.

Logging Out : logout

It is time for you to end your ssh session and return to your original CSIF computer. Now type **`logout`** You should receive the message "Connection to pc23 closed.", where pc23 was the name of the remote computer.

Now type **`ls`**. You should see your *first* directory on this computer.

Now type **`cd first/fourth`** to change into the subdirectory named *fourth* in the directory named *first*.

Type **`ls`** again. *snake* and *wombat* should be listed.

You can now see that you can log on to any one of the CSIF hosts, and you will see your account exactly as you left it. The reason for this is that each of these machines gets its information about your account from a central *server* that has a hard drive on which your account is stored. Thus, each of these hosts is like a "window" into your account: you will see the same thing from any window.

The Terminate Key : ^C,

On the left side of the keyboard is a key labeled Ctrl. Use this key in combination with others to send control signals to Unix. In UNIX literature, the Ctrl key is specified by using the ^. For example, ^C means "hold down the Ctrl key, then type c". Notice I don't mean capital C, even though by convention that is how it's written. **^C** usually causes instant termination of any running program.

For example, the ping utility checks the quality of the network connection between your computer and a remote computer. It will continue to check the quality until you terminate it by typing **^C**. The syntax of ping is ping *computer_name*.

Now type **ping pc3**.

Your screen will start filling with lines indicating the quality of the connection to pc3.

Terminate ping by typing ^**C**.

ping should stop, and you should see the command prompt again.

You can use **^C** in most situations when a command or program is causing the computer to perform in an unsatisfactory manner.

Displaying Files: cat, more, head

The syntax of the command to display a file is: cat {*file_name}*+**.** Now type **cat wombat**.

The contents of the file just flew by on the screen. For short files this might be fine, but usually you will wish to see the file a little at a time.

You can use more {*file_name*}**+** to scroll through a file. Now type **more wombat**.

You will see the first screen-worth of the file displayed. To scroll forward one line, press the Enter key. To scroll forward an entire screen, press the Space bar. To quit in the middle of displaying a file, type the letter **q.** There is no way to scroll backwards with more.

Sometimes you will just wish to see the first few lines of a file. You can then use the head command. The syntax is head [-n] {*file_nameI}*+, where "n" is the number of lines to be displayed, e.g. head -3 snake. If you leave out n, then head defaults to the first 10 lines. Now type **head –4 snake**.

Transfering Files: sftp

You can use the sftp (Secure File Transfer Protocol) command to retrieve or place files in remote locations, provided you have permission to access the location where the files are. WinSCP is a free Windows sftp program with a GUI. You can download WinSCP from http://winscp.net/eng/index.php. You can download s GUI version of sftp for Macs by following UCD Resources->Software->File Transfer tabs within your MyUCDavis web page to get Fugu for Macs.

The syntax of sftp is: sftp *computer_name*. As with ssh, you do not have to include the domain of the computer if you are ftping from a computer in the same domain. You are now going to go through the process of getting your wombat file from another computer.

Before continuing, type **cd** to get you in your home directory.

Now type **sftp *computer_name***, where ***computer_name*** is some CSIF computer (remember ruptime?).

As in ssh, if you are trying to access a computer that is new to you, sftp will ask you to OK its authenticity; do so. You will be asked for your username. In this case you can simply hit enter, because the username it has detected is proper for the destination computer. If your account is under a different name, then you have to enter that name. Typically, many ftp sites let you access them under the user name anonymous, using your e-mail address as the password. In the CSIF you

will still need to type in your CSIF password.

You can navigate in ftp using the cd, pwd, and ls**.** ftp also has a dir command which works like ls -l to show more information about files.

Now type **pwd**, and you will see that you are in your own home directory.

Now type **cd first/fourth** to move the remote ftp session into the *fourth* directory.

You can use **lcd** to change your local directory.

Now type **lcd first** to move to your *first* directory on the computer in front of you.

Now type **lpwd** to see where you are on the computer in front of you.

Now type **ls** to see the files available to you on the remote computer.

You should see *snake* and *wombat* listed.

The "get" command retrieves files from the remote computer. Now type **get snake** to retrieve *snake* into your *first* directory.

If you had wanted to move multiple files you could use mget which has the syntax of mget {*file_name}*+.

If you had wanted to transfer a file from the local computer to the remote computer you would use put, or mput.

Now that you are done transfering files type **bye** to end the ftp session.

To see if *snake* is really in your *first* directory you can type **ls first**.

Submitting Files: handin

The handin program for electronic submission. handin provides a secure means of submitting files to another user, recounting what has already been submitted, and listing what subdirectories exist for containing submissions.

The syntax is: handin *to_user_name* [*directory_name*] {*file_name}**

To determine the available directories in an account you omit the *directory_name* and *file_names*. For ECS 30, the to_user_name would be cs30. You should now type **handin cs30**.

The program should tell you that the lab, and Proj1 to Proj5 directories are available.

To determine the files you have already submitted to a specific directory you would use the following syntax: handin *to_user_name* [*directory_name*]**.**

Now type **handin cs30 lab**.

No files should appear.

Now, use cd change to your */first/fourth* directory.

If you then use **ls** you should see both *wombat* and *snake* there.

To submit both files, now type: **handin cs30 lab wombat snake**.

Assuming all has gone well, the handin command will copy all the specified files to *cs30_account/handin/tutorial/from_user*, where *from_user* is username of the person doing the submitting, i.e. your username. The directory *from_user* is created if it doesn't already exist, and is named after the invoking user. Therefore, each user can submit files with the same names as other users and there will be no overwriting, because each user has his/her own subdirectory. If you resubmit a file with the same name as one you had previously submitted, the new file will overwrite the old file. This is handy when you suddenly discover a bug after you've used handin. But beware, the time stamps of the files in *cs30* are those of the time of submission, and not those of the time of file creation. A file you wrote yesterday, but handin today, will have today's date and submission time. DO NOT use handin after the due date time.

Now type handin cs30 tutorial to see that the files are indeed in that directory. You should see both snake and wombat.

Text Editors ***

There are many text editors available: 1) `vi`, 2) `emacs`, 3)`gedit`, `etc` . `We will be using vi, gvim`

The first two each require some effort to master. See the references for more details. The last one, gedit, is menu-driven and fairly self explanatory, though much less powerful than the others.

To create a new file, or edit an existing one, just type the name of your editor followed by the name of the file, e.g. vi myfile. You can also type any editor name without specifying a file, then save whatever you have done to a file later.

One feature of `emacs` which is expecially powerful is the ability to maintain multiple *buffers* of files. Each file that you open is stored by `emacs`, so that you can switch among them, working on each one or comparing them without having to open up another text editor to do so. In addition, you can run a *shell* from `emacs`, so that you can interact with the computer without leaving the text editor environment. This is especially handy if, for example, you are trying to debug a program. You can edit your program, then compile and run it to see if it works, all within `emacs`.
end ***

Printing files : lpr, lpq, lprm, enscript

The syntax of the standard printing command is: lpr [-h] {*file_name*}+ The printer for the CSIF is in room 86 Kemper. The -h option suppresses printing of the header page. To print *file1* without a header pager, you could type either lpr -h file1.

To check the status of your print order you would type lpq`.` This results in a listing of all print jobs (with their job numbers) on the print queue.

To remove a print job, the syntax is lprm *job_number* Notice you can get your *job number* from the lpq command. Without the job number, you would remove only the job of yours that is currently active. To remove *all* your jobs, put a dash in place of the job number, e.g. lprm -

The other print command that is useful for compressing two pages onto one has the syntax:

enscript [-2r] [-h] {*file_name*}+

As expected, the -h option is used to suppress printing of the header page. But more important, the option -2r says that you want to rotate the paper sheet by 90 degrees, and print two pages onto this one sheet. This saves paper and lets you read more of a long program at a glance.

Type **cd** to get to your home directory.

Now type **enscript -2r .cshrc** to print the your .cshrc file with a header page. (Note that in this particular example the name of the file happens to begin with a period.)

Go to 86 Kemper, and retrieve your printout, and discard it.

Help Command : man

You can get online documentation (which is not always easy to read!) for almost any command in Unix. You just type the command man followed by the name of the command. For example, here is a small portion of what you get when asking about ls:

```
% man ls
  Name
     ls - list and generate statistics for files

   Syntax
     ls [ options ] name ...

   Description
     For each directory argument, ls lists the contents of the directory.
     For each file argument, ls repeats the file name and gives any other
     information you request with the options available.  By default, the
     list is sorted alphabetically.  When no argument is given, the current
     directory is listed.  When several arguments are given, files are listed
     first, followed by directories and the files within each directory.
     Options are listed below.
```

Notice the line under "Syntax". The meaning is as follows:

- First is the command name
- The square brackets [] indicate that the *options* are optional! In other words, you don't have to type any options to run the basic command. If you *do* use options, then you must select from a list that comes after what is shown here.
- name ... means that zero or more names of files may be entered here. The three dots indicate that the number of names is variable. If you enter no name, then by default you'll get a listing of the current working directory.

The man page presentation is shown using more (see above, "Displaying Files"), so when you have read one full screen, you hit the spacebar.

Logging Out: exit

You are done with the tutorial! Type **`exit,`** to terminate the command line sessions.

To terminate your session, click on the System drop down menu of the toolbar, and then select "Log Out" near the bottom of the menu. You will then be presented with a dialog box to confirm that you wish to logout. Click the "Logout" or the "Continue logout" button to complete the logout process. Congratulations!! Your done!!