Unix/Linux CLI Tutorial

Authors: Mobeen Ludin and Aaron Weeden

Last Updated: May 14, 2015

PART 1: Introduction

The following is a very brief introduction to some useful Unix/Linux commands, including examples of how to use each command in a command-line interface (CLI). For more extensive information about any of these commands, use the **`man`** command as described below. Sources for more information appear at the end of this document.

What is Unix: Unix is a common operating system (a suite of programs that make the computer work) made up of three parts: the kernel, the shell, and programs.

What is the Kernel: The kernel of Unix is the hub of the operating system; it manages hardware, allocates time and memory to programs, and handles the filestore and communications in response to system calls (e.g. services to create a file, begin execution of a program, or open a network connection to another computer).

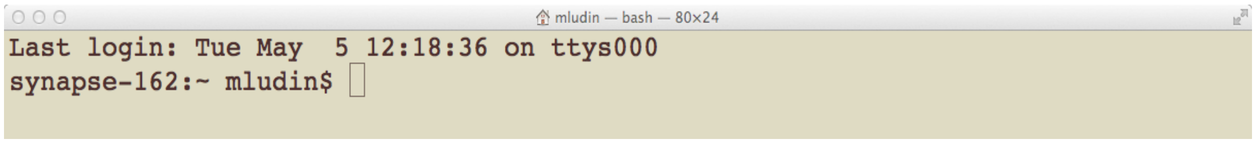
What is the shell: The shell is the interface between the user and the kernel. When a user logs in, the login program checks the username and password and then starts another program called the shell. The shell is a command-line interpreter (CLI).

What are commands: Once common thing about every Unix/Linux OS is that it comes with a bunch of tools that could be used to navigate the system, create files and directories, find files, move files around, monitor the system, run programs, etc. In short, it gives you the power to do things quickly and efficiently. Let's get started with learning some useful commands.

Note: Commands are in a **`bold, fixed-width font`**, and they start with a "`$`" (sometimes called a prompt). The "`$`" is not part of the command; only the text after it is.

Starting Terminal/Cygwin: We will be using **Terminal** (on Mac) or **Cygwin** (on Windows) to login to a remote server and practice Unix commands. We will refer to both Terminal and Cygwin as "terminal" moving forward.

a. Launch a terminal. It will look something like this on Mac:



```
Last login: Tue May  5 12:18:36 on ttys000
synapse-162:~ mludin$
```

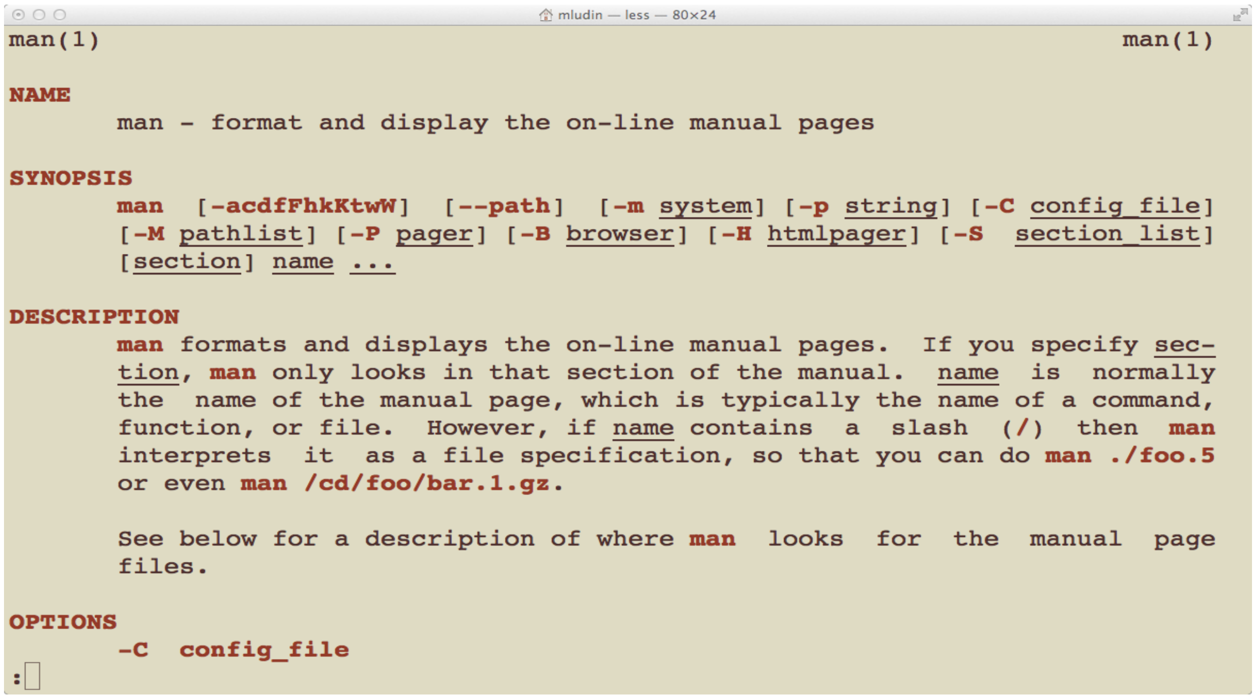
Note the "$", which indicates the prompt, where commands can now be entered.

PART 2: Unix/Linux Command Line

`$ man`: This stands for "manual page". Basically every Unix command has a manual page associated with it, which describes what the command is and how to use it. If you wanted to learn more about a command, at the prompt you could enter **`man`** followed by a space and then the name of the command. For example, let's learn more about the man command:

```
$ man man
```

Output:



```
man(1)                                                                  man(1)

NAME
       man - format and display the on-line manual pages

SYNOPSIS
       man  [-acdfFhkKtwW]  [--path]  [-m system] [-p string] [-C config_file]
       [-M pathlist] [-P pager] [-B browser] [-H htmlpager] [-S  section_list]
       [section] name ...

DESCRIPTION
       man formats and displays the on-line manual pages.  If you specify sec-
       tion, man only looks in that section of the manual.  name  is  normally
       the  name of the manual page, which is typically the name of a command,
       function, or file.  However, if name contains  a  slash  (/)  then  man
       interprets  it  as a file specification, so that you can do man ./foo.5
       or even man /cd/foo/bar.1.gz.

       See below for a description of where man  looks  for  the  manual  page
       files.

OPTIONS
       -C  config_file
:
```

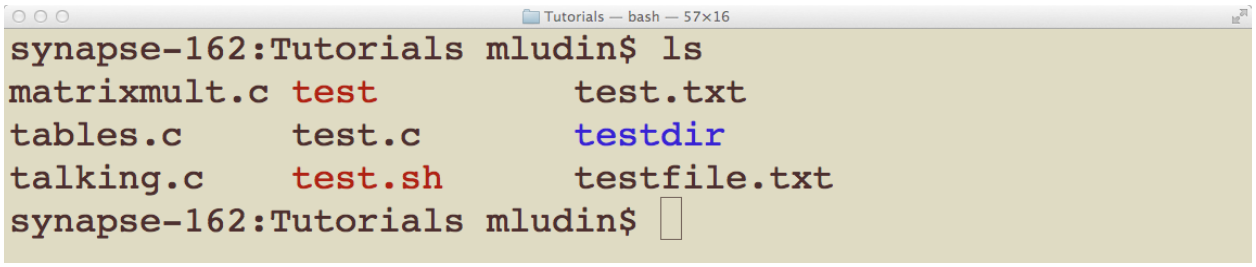
To scroll down/up, use the arrow keys. To quit out of the manual page, press q.

Navigation Commands (`ls, ls -l, pwd, mkdir, cd, cd ..`):

`$ ls`: This command will list the files stored in the current working directory. To see a brief, multi-column list of the files in the current directory, enter:

```
$ ls
```

Output: Depending on your system, there may be a different coloring shown (generally, red fonts are for executable files and blue for directories).

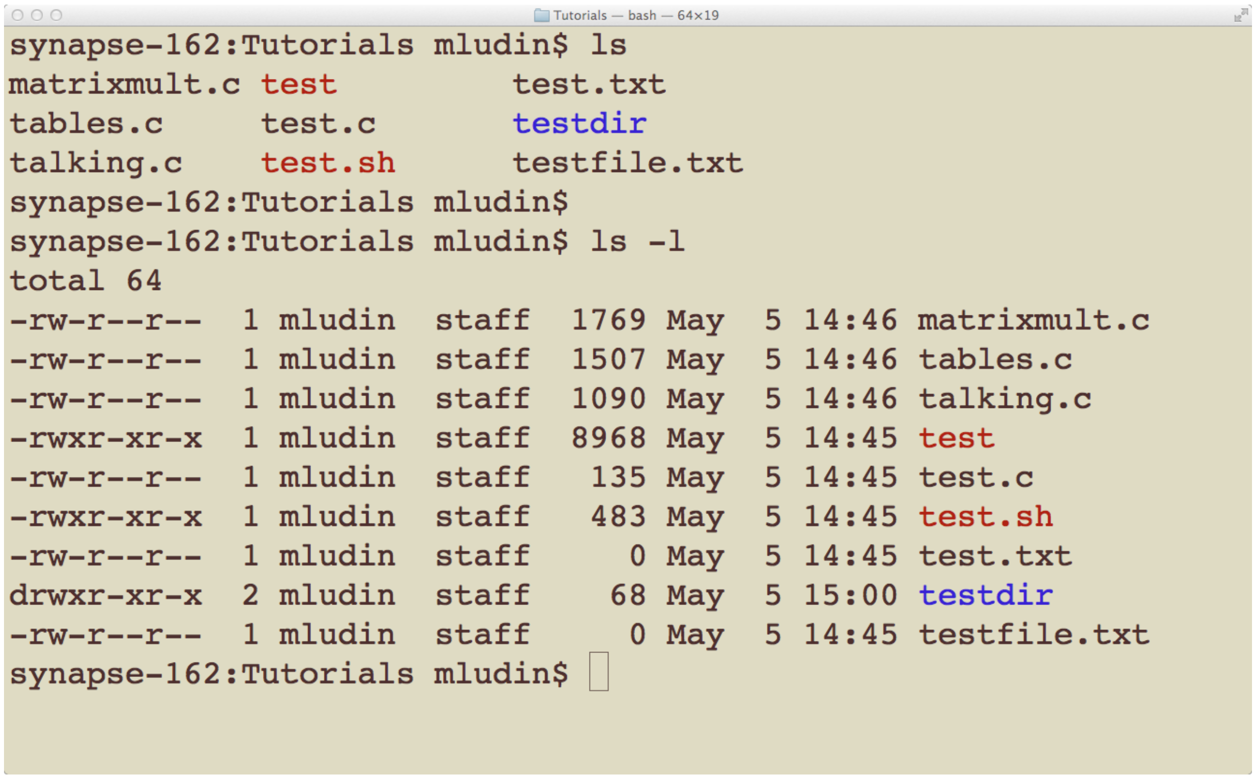


```
synapse-162:Tutorials mludin$ ls
matrixmult.c test         test.txt
tables.c     test.c       testdir
talking.c    test.sh      testfile.txt
synapse-162:Tutorials mludin$
```

`$ ls -l`: This command will list the files and folders in the current directory with more details, such as date last modified, owner, permissions, etc.

```
$ ls -l
```

Output:

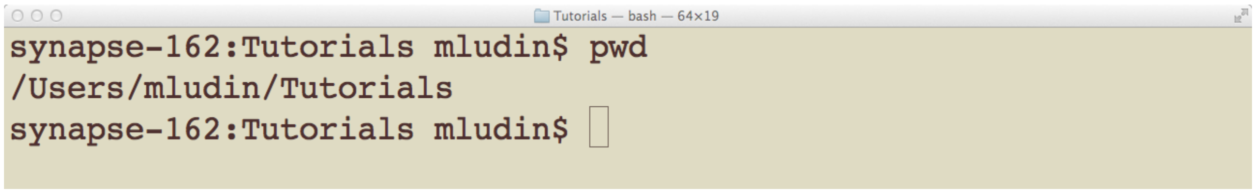


```
synapse-162:Tutorials mludin$ ls
matrixmult.c test         test.txt
tables.c     test.c       testdir
talking.c    test.sh      testfile.txt
synapse-162:Tutorials mludin$
synapse-162:Tutorials mludin$ ls -l
total 64
-rw-r--r--  1 mludin  staff  1769 May  5 14:46 matrixmult.c
-rw-r--r--  1 mludin  staff  1507 May  5 14:46 tables.c
-rw-r--r--  1 mludin  staff  1090 May  5 14:46 talking.c
-rwxr-xr-x  1 mludin  staff  8968 May  5 14:45 test
-rw-r--r--  1 mludin  staff   135 May  5 14:45 test.c
-rwxr-xr-x  1 mludin  staff   483 May  5 14:45 test.sh
-rw-r--r--  1 mludin  staff     0 May  5 14:45 test.txt
drwxr-xr-x  2 mludin  staff    68 May  5 15:00 testdir
-rw-r--r--  1 mludin  staff     0 May  5 14:45 testfile.txt
synapse-162:Tutorials mludin$
```

$ pwd (Print Working Directory): This command reports the current directory path. When you first login, your current working directory is usually your home directory. Your home directory has the same name as your username. For example, it is `mludin` for me.

```
$ pwd
```

Output:



```
synapse-162:Tutorials mludin$ pwd
/Users/mludin/Tutorials
synapse-162:Tutorials mludin$ 
```

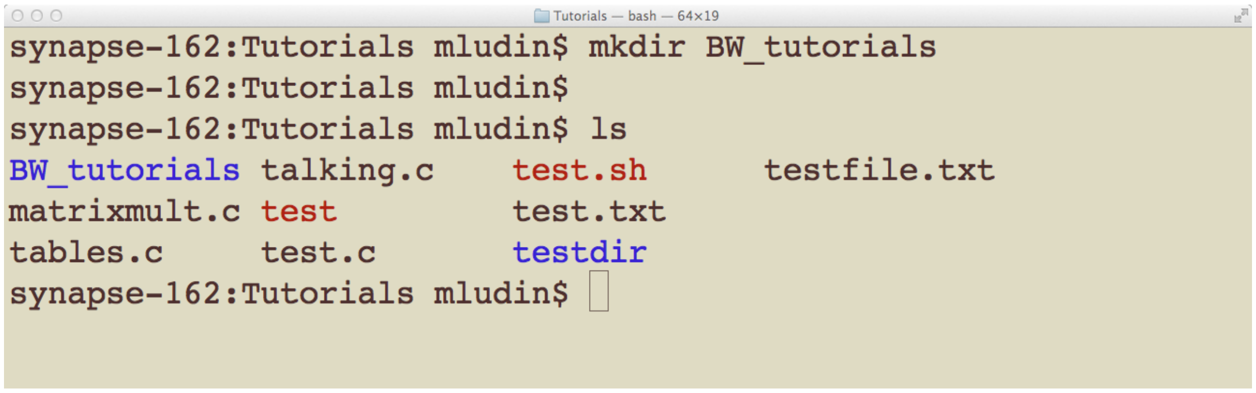
Unix directories are arranged in a hierarchy with the root (`/`, slash) at the top. When you have a directory inside another directory (e.g. the `Tutorials` directory inside your home directory), in the path name these directories will be separated by slashes (`/`).

$ mkdir: This command will create a new subdirectory in the current location. Let's create a new directory and call it `BW_tutorials`. We will use the same directory to save all our work during this tutorial.
NOTE: Unix is case-sensitive, so `BW` is different from `bw`; also, it is recommended NOT to use spaces in the names of the files or directories - this makes them very difficult to work with in Unix. Lastly, whenever you create a new directory, it is recommended to use the **`ls`** or **`ls -l`** commands to make sure the new directory was created successfully.

```
$ mkdir BW_tutorials
```

```
$ ls
```

Output:

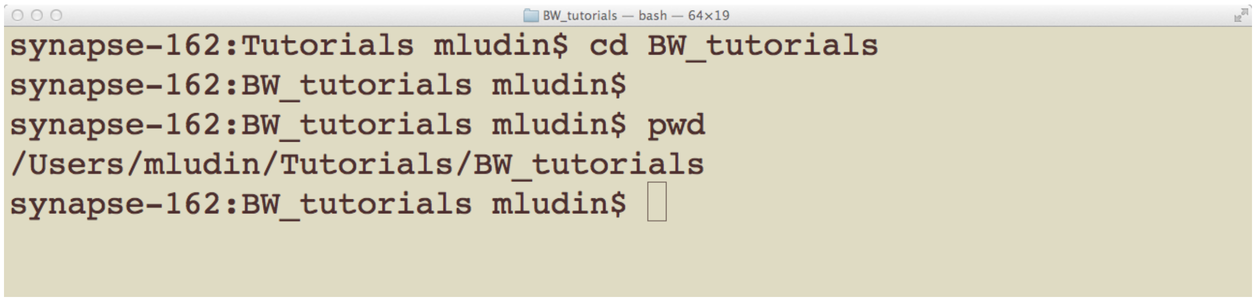


```
synapse-162:Tutorials mludin$ mkdir BW_tutorials
synapse-162:Tutorials mludin$
synapse-162:Tutorials mludin$ ls
BW_tutorials talking.c     test.sh       testfile.txt
matrixmult.c test          test.txt
tables.c     test.c        testdir
synapse-162:Tutorials mludin$ 
```

$ cd (change directory): This command changes your current directory location. To go to the directory we just created (`BW_tutorials`), enter:

```
$ cd BW_tutorials
```

Output:



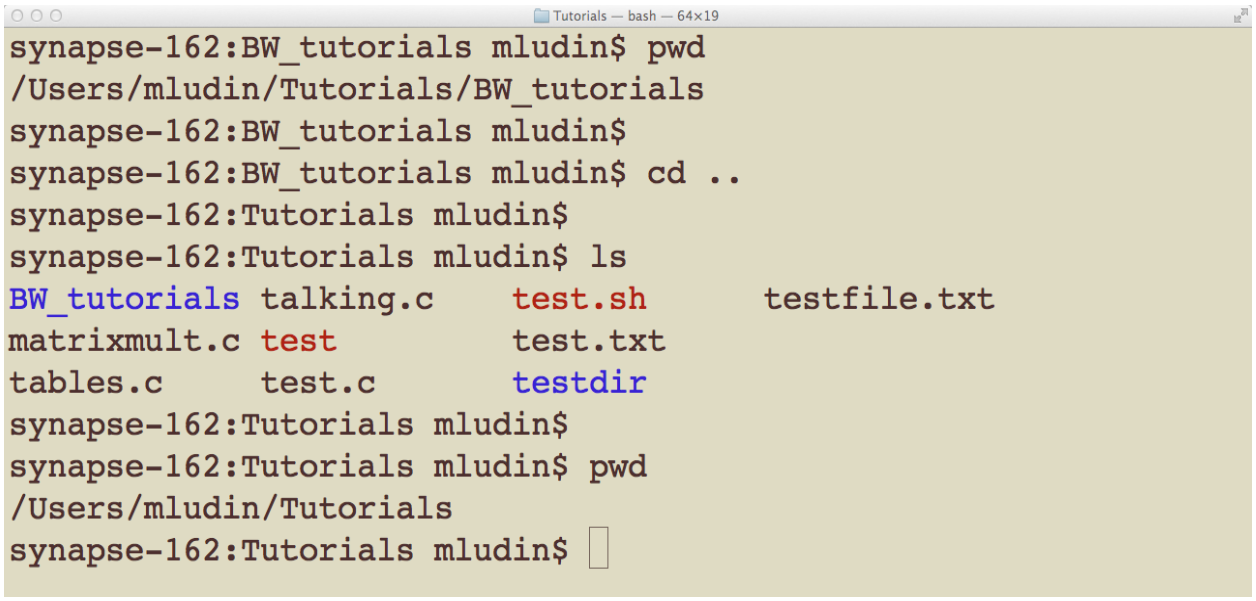
```
synapse-162:Tutorials mludin$ cd BW_tutorials
synapse-162:BW_tutorials mludin$
synapse-162:BW_tutorials mludin$ pwd
/Users/mludin/Tutorials/BW_tutorials
synapse-162:BW_tutorials mludin$
```

Note: To make sure you successfully changed your current working directory to `BW_tutorials`, you can use the **`pwd`** command as shown in the above picture. You could also use the **`ls`** command to see its content (but it should be empty).

$ cd ..: This will take us one level up in the directory hierarchy. If we want to go to the parent directory of our current location, enter:

```
$ cd ..
```

Output:



```
synapse-162:BW_tutorials mludin$ pwd
/Users/mludin/Tutorials/BW_tutorials
synapse-162:BW_tutorials mludin$
synapse-162:BW_tutorials mludin$ cd ..
synapse-162:Tutorials mludin$
synapse-162:Tutorials mludin$ ls
BW_tutorials talking.c    test.sh      testfile.txt
matrixmult.c test         test.txt
tables.c     test.c       testdir
synapse-162:Tutorials mludin$
synapse-162:Tutorials mludin$ pwd
/Users/mludin/Tutorials
synapse-162:Tutorials mludin$
```

Note: I have used the `pwd` and `ls` commands again to verify I know what I am doing.

Exercise: Find out the result for the following commands. What output is given? What does the command list, or where does the command move you in the directory hierarchy?

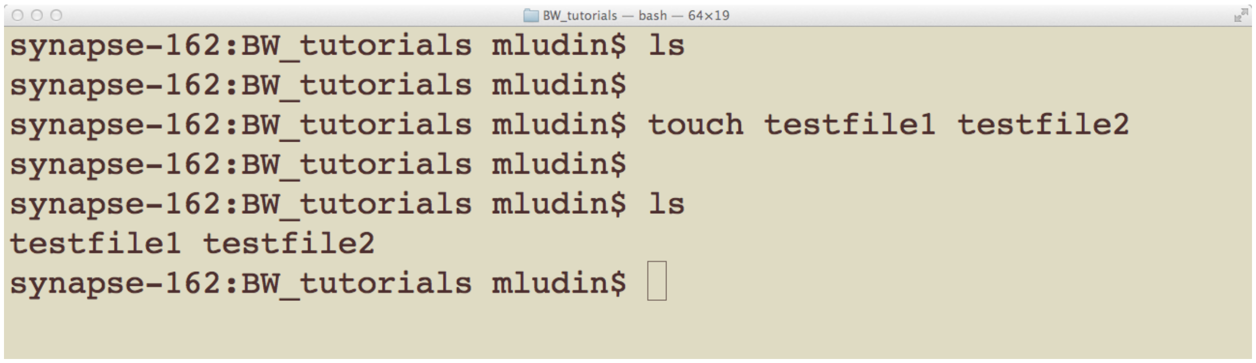
1.
```
$ cd ~
```
2.
```
$ ls -a
```
3.
```
$ cd ~mludin
```
4.
```
$ cd
```
5.
```
$ ls ~mludin
```

Create an empty directory called `testdir` under `BW_tutorials`. Remember you can use **`ls`** and **`pwd`** to help you verify it worked.

File manipulation commands (`touch`, `cat`, `cp`, `mv`, `rm`, `tail`, `head`, `less`, `more`):
These commands are used to rename, move, delete, or read files without using a text editor. Read the commands' **`man`** pages for more details. We will only cover a few of them here.
`$ touch filename`: This command will create an empty file called `filename` in your current directory. To create two empty files with names `testfile1` and `testfile2`, enter:

```
$ touch testfile1 testfile2
```

Output:



```
synapse-162:BW_tutorials mludin$ ls
synapse-162:BW_tutorials mludin$
synapse-162:BW_tutorials mludin$ touch testfile1 testfile2
synapse-162:BW_tutorials mludin$
synapse-162:BW_tutorials mludin$ ls
testfile1 testfile2
synapse-162:BW_tutorials mludin$ 
```

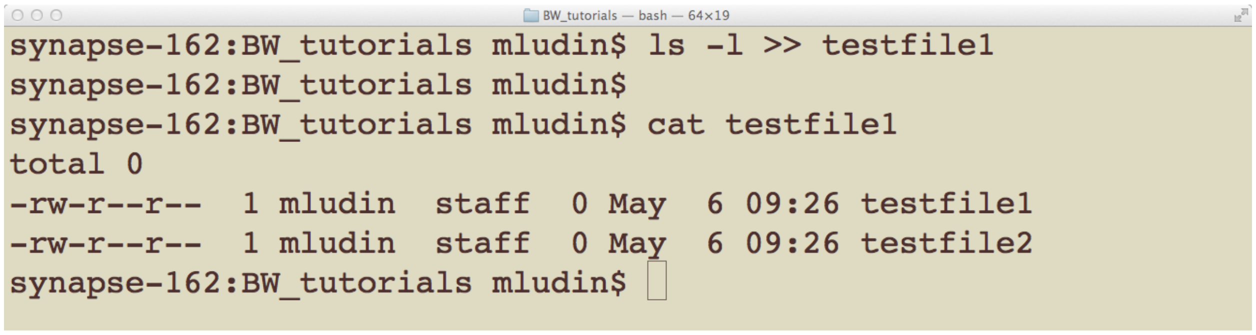
How to append the output of a command to another file?
`>>`: You can use two greater-than signs (>>) to take the output of a command and append it to (add it to the bottom of) a file. For example, to take the output of the **`ls -a`** command and append it to `testfile1`,

```
$ ls -a >> testfile1
```

$ cat: This command outputs the entire contents of a text file:

```
$ cat testfile1
```

Output:

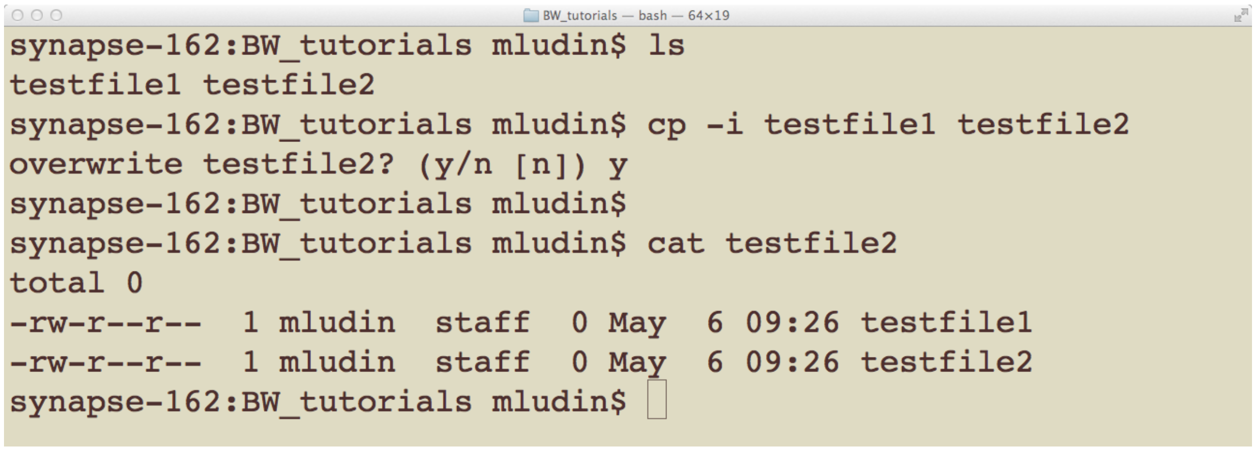


```
synapse-162:BW_tutorials mludin$ ls -l >> testfile1
synapse-162:BW_tutorials mludin$
synapse-162:BW_tutorials mludin$ cat testfile1
total 0
-rw-r--r--  1 mludin  staff  0 May  6 09:26 testfile1
-rw-r--r--  1 mludin  staff  0 May  6 09:26 testfile2
synapse-162:BW_tutorials mludin$
```

$ cp: This command copies a file, preserving the original and creating an identical copy. If you already have a file with the new name, ***`cp` will overwrite and destroy the duplicate***. For this reason, it's safer to add **`-i`** after the **`cp`** command to force the system to ask for your approval before it destroys any files. The general syntax for **`cp`** is:

```
$ cp -i testfile1 testfile2
```

Output:



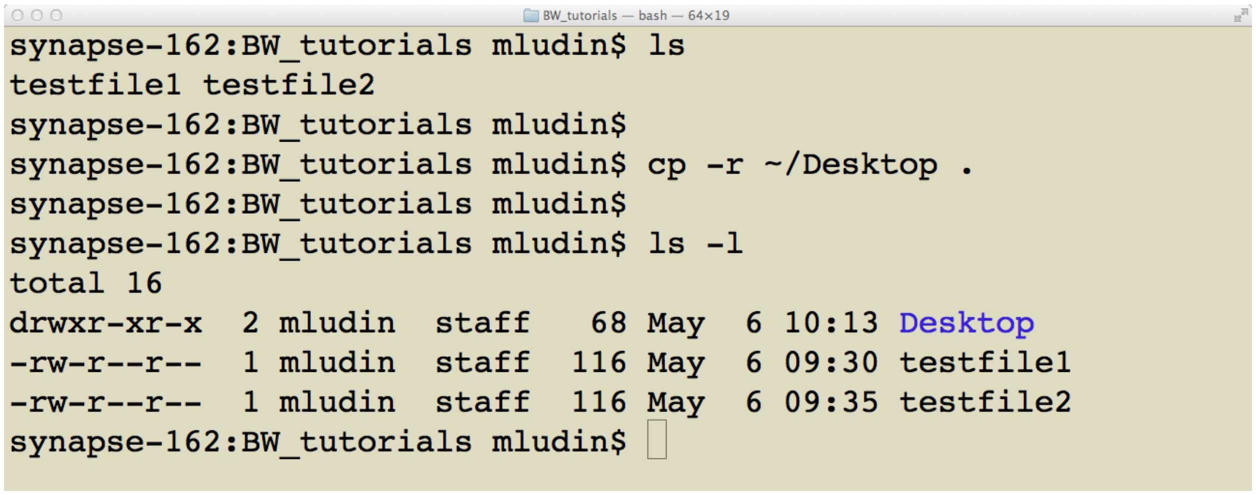
```
synapse-162:BW_tutorials mludin$ ls
testfile1 testfile2
synapse-162:BW_tutorials mludin$ cp -i testfile1 testfile2
overwrite testfile2? (y/n [n]) y
synapse-162:BW_tutorials mludin$
synapse-162:BW_tutorials mludin$ cat testfile2
total 0
-rw-r--r--  1 mludin  staff  0 May  6 09:26 testfile1
-rw-r--r--  1 mludin  staff  0 May  6 09:26 testfile2
synapse-162:BW_tutorials mludin$
```

$ cp -r: If you want to make a copy of a directory and all of the files and directories under it, use the "`-r`" option for recursive copy. For example, to copy the `Desktop` folder and all of its files and subfolders into our current directory, enter:

```
$ cp -r ~/Desktop .
```

NOTE: Whenever you copy a file or directory to a different location, you first provide the source, followed by the destination, separated by spaces. In the above command, the `Desktop` directory is the source, and the "dot" (`.`) is the destination. In Unix, a single dot means "here" or the current working directory.

Output:



BW_tutorials — bash — 64×19

```
synapse-162:BW_tutorials mludin$ ls
testfile1 testfile2
synapse-162:BW_tutorials mludin$
synapse-162:BW_tutorials mludin$ cp -r ~/Desktop .
synapse-162:BW_tutorials mludin$
synapse-162:BW_tutorials mludin$ ls -l
total 16
drwxr-xr-x  2 mludin  staff   68 May  6 10:13 Desktop
-rw-r--r--  1 mludin  staff  116 May  6 09:30 testfile1
-rw-r--r--  1 mludin  staff  116 May  6 09:35 testfile2
synapse-162:BW_tutorials mludin$
```

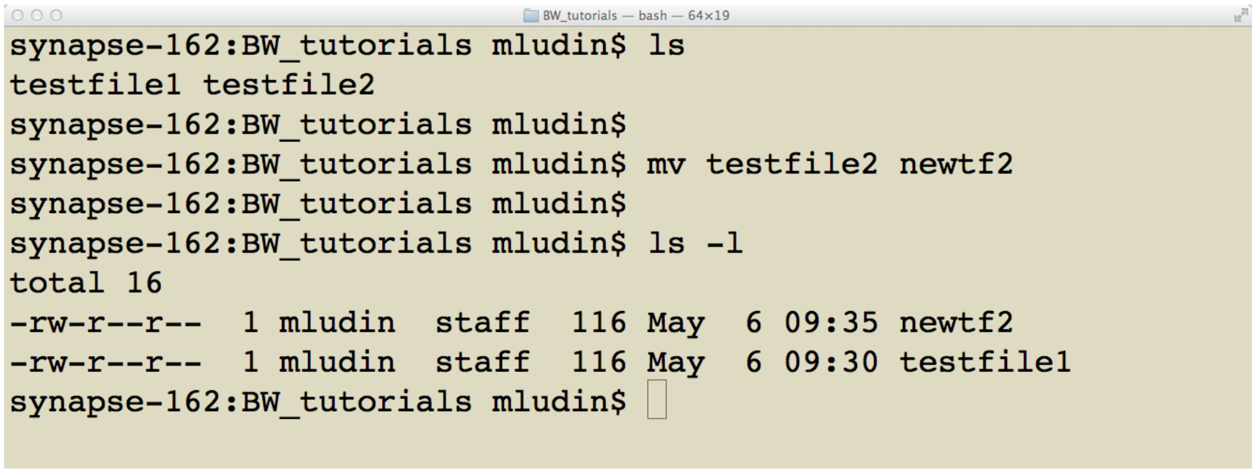
`$ mv`: This command will move a file or directory. Unlike the **`cp`** command, **`mv`** will not preserve the original file. You can also use **`mv`** to rename files or directories.

Note: As with the **`cp`** command, you can always use **`-i`** to make sure you do not overwrite an existing file.

To rename a file named `testfile2` in the current directory to the new name `newtf2`, enter:

```
$ mv -i testfile2 newtf2
```

Output:



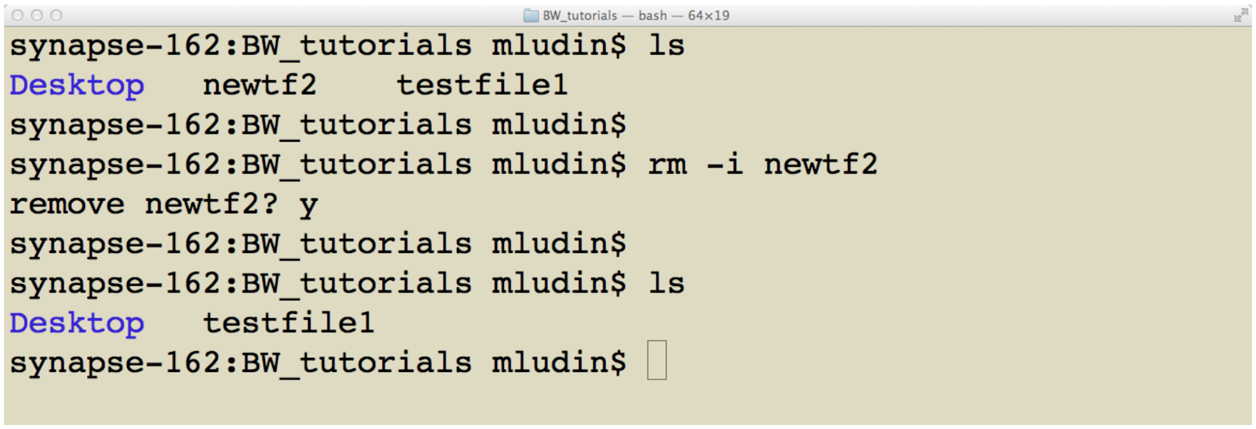
BW_tutorials — bash — 64×19

```
synapse-162:BW_tutorials mludin$ ls
testfile1 testfile2
synapse-162:BW_tutorials mludin$
synapse-162:BW_tutorials mludin$ mv testfile2 newtf2
synapse-162:BW_tutorials mludin$
synapse-162:BW_tutorials mludin$ ls -l
total 16
-rw-r--r--  1 mludin  staff  116 May  6 09:35 newtf2
-rw-r--r--  1 mludin  staff  116 May  6 09:30 testfile1
synapse-162:BW_tutorials mludin$
```

$ rm: This command will remove (destroy) a file. You can always enter this command with the **-i** option, so that you will be asked to confirm each file deletion. To remove the file named `newtf2`, enter:

```
$ rm -i newtf2
```

Output:



```
synapse-162:BW_tutorials mludin$ ls
Desktop    newtf2     testfile1
synapse-162:BW_tutorials mludin$
synapse-162:BW_tutorials mludin$ rm -i newtf2
remove newtf2? y
synapse-162:BW_tutorials mludin$
synapse-162:BW_tutorials mludin$ ls
Desktop    testfile1
synapse-162:BW_tutorials mludin$
```

Note: Using rm will remove a file permanently (there is NO trash can or recycle bin to pull it from!), so be sure you really want to delete a file before you use rm!
To remove a non-empty subdirectory, **rm** accepts the `-r` option. Note that this will remove the directory and everything in it, usually without warning you first, so you may want to include the **-i** option as well, just to be safe.

Exercise: Find out the result for the following commands:

6.
```
$ ls -al > testfile1
```
7.
```
$ cat testfile1 >> testfile2
```
8.
```
$ man less > testfile1
```
9.
```
$ less testfile1
```
10.
```
$ tail testfile1
```
11.
```
$ head testfile1
```
12.
```
$ rm -r testdir
```
It is recommended to use **ls**, **pwd**, and **cat** to verify the above commands work as you expect.

PART 3: Remote Access (Shodor Servers / BW System) Using SSH:

`$ ssh`: SSH stands for Secure Shell, and it provides us a secure and encrypted channel to login to a remote computer over the network. Before you log into a remote machine, you will need an Internet connection and to know the following information for that system:

Username: `mludin` (this is as an example. your username should be different)

Password: (You should know your password for each system)

Hostname: `login.shodor.org` (This is the Shodor server you will use)

Once you know the above information, start a terminal and enter the following command:

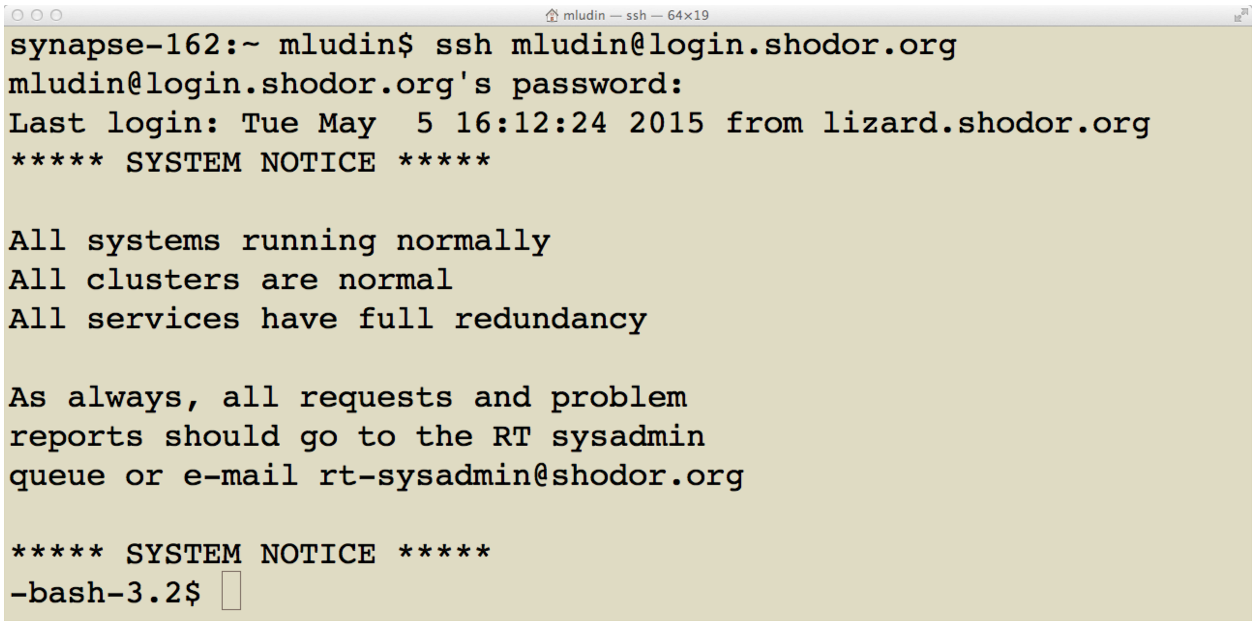
```
$ ssh username@hostname
```

NOTE: Make sure to replace the "username" and "hostname" your own correct info.

For example:

```
$ ssh mludin@login.shodor.org
```

Output:



```
synapse-162:~ mludin$ ssh mludin@login.shodor.org
mludin@login.shodor.org's password:
Last login: Tue May  5 16:12:24 2015 from lizard.shodor.org
***** SYSTEM NOTICE *****

All systems running normally
All clusters are normal
All services have full redundancy

As always, all requests and problem
reports should go to the RT sysadmin
queue or e-mail rt-sysadmin@shodor.org

***** SYSTEM NOTICE *****
-bash-3.2$
```

Note: Unix usually hides the password, so it won't show anything as you type. If you think you have entered the wrong characters, just enter backspace a lot and then reenter the correct password.

Once you successfully enter your password and login to the Shodor server, you will see a welcome message that contains the information about last login and information about how to contact system administrators for issues.

Also, note that the shell prompt might look very different depending on the system and the version of the Linux/Unix distribution installed.

If you are planning on using X11 for remote display, add a "`-X`" to your ssh command:

```
$ ssh -X mludin@login.shodor.org
```

PART 4: Remote Copy (To/From Shodor Servers/BW System) Using SCP:
`$ scp`: SCP stands for Secure Copy, and it provides a secure and encrypted channel to copy files to/from a remote computer over the network. You will need to know the following information:
Username: `mludin` (this is as an example. your username should be different)
Password: (You should know your password for each system)
Hostname: `login.shodor.org` (this is the Shodor server you will use)
Source: `name_of_the_file` (where to copy from on the "source" system)
Destination: `name_of_directory` (where to copy to on the "destination" system)

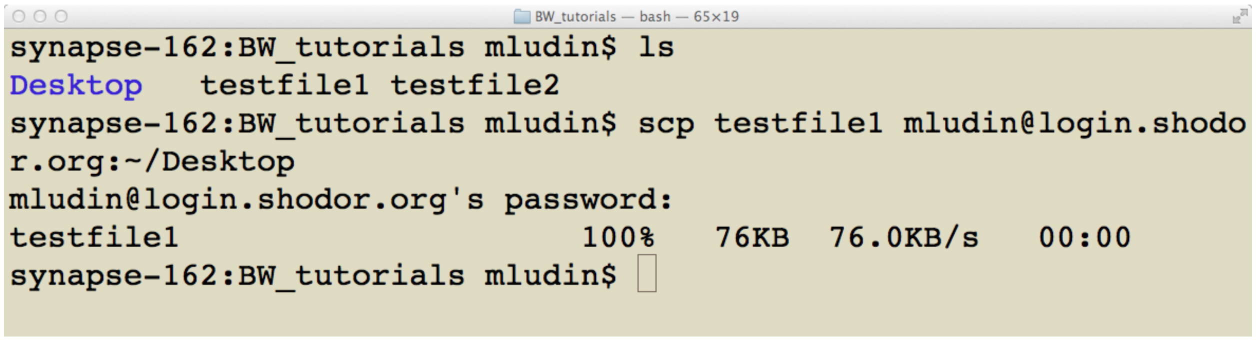
From a local source system to a remote destination system: Let's copy some files from our local computer (e.g. our laptop/desktop/a machine we `ssh`ed into) to a remote computer. The syntax looks like:

```
$ scp source username@hostname:~/destination
```

Example: Let's copy the file named `testfile1` (which we have been playing with) to the `Desktop` folder in your account on Shodor server (`login.shodor.org`):

```
$ scp testfile1 mludin@login.shodor.org:~/Desktop
```

Output:



```
synapse-162:BW_tutorials mludin$ ls
Desktop   testfile1 testfile2
synapse-162:BW_tutorials mludin$ scp testfile1 mludin@login.shodo
r.org:~/Desktop
mludin@login.shodor.org's password:
testfile1                      100%   76KB  76.0KB/s   00:00
synapse-162:BW_tutorials mludin$
```

Note: The "`~/`" is used to indicate your home directory. Also, the password will not be visible as you type it.

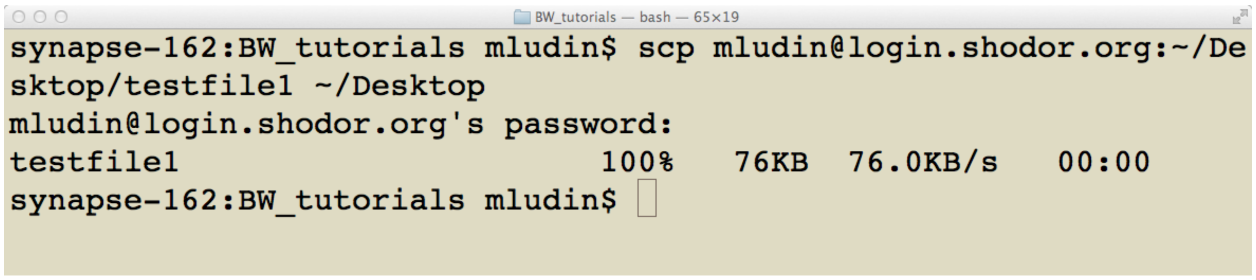
From a remote machine to our machine: Now let's copy a file from a remote machine to our local computer. The syntax is:

```
$ scp username@hostname:~/source destination
```

Example: Now let's copy back the file name `testfile1` from the `Desktop` directory in your account on the Shodor server to the `Desktop` folder on your local machine. If you are already on Shodor server, make sure to exit out by pressing Control-D or entering the command `exit` or `logout`.

```
$ scp mludin@login.shodor.org:~/Desktop/testfile1 ~/Desktop
```

Output:



```
synapse-162:BW_tutorials mludin$ scp mludin@login.shodor.org:~/De
sktop/testfile1 ~/Desktop
mludin@login.shodor.org's password:
testfile1                                    100%    76KB   76.0KB/s    00:00
synapse-162:BW_tutorials mludin$ 
```

Note: If you want to copy a file from a remote server to your current working directory, just replace the "`~/Desktop`" with "`.`".

Copying files to/from Blue Waters:

To login to BW system, make sure you have following information handy:

Your BW username

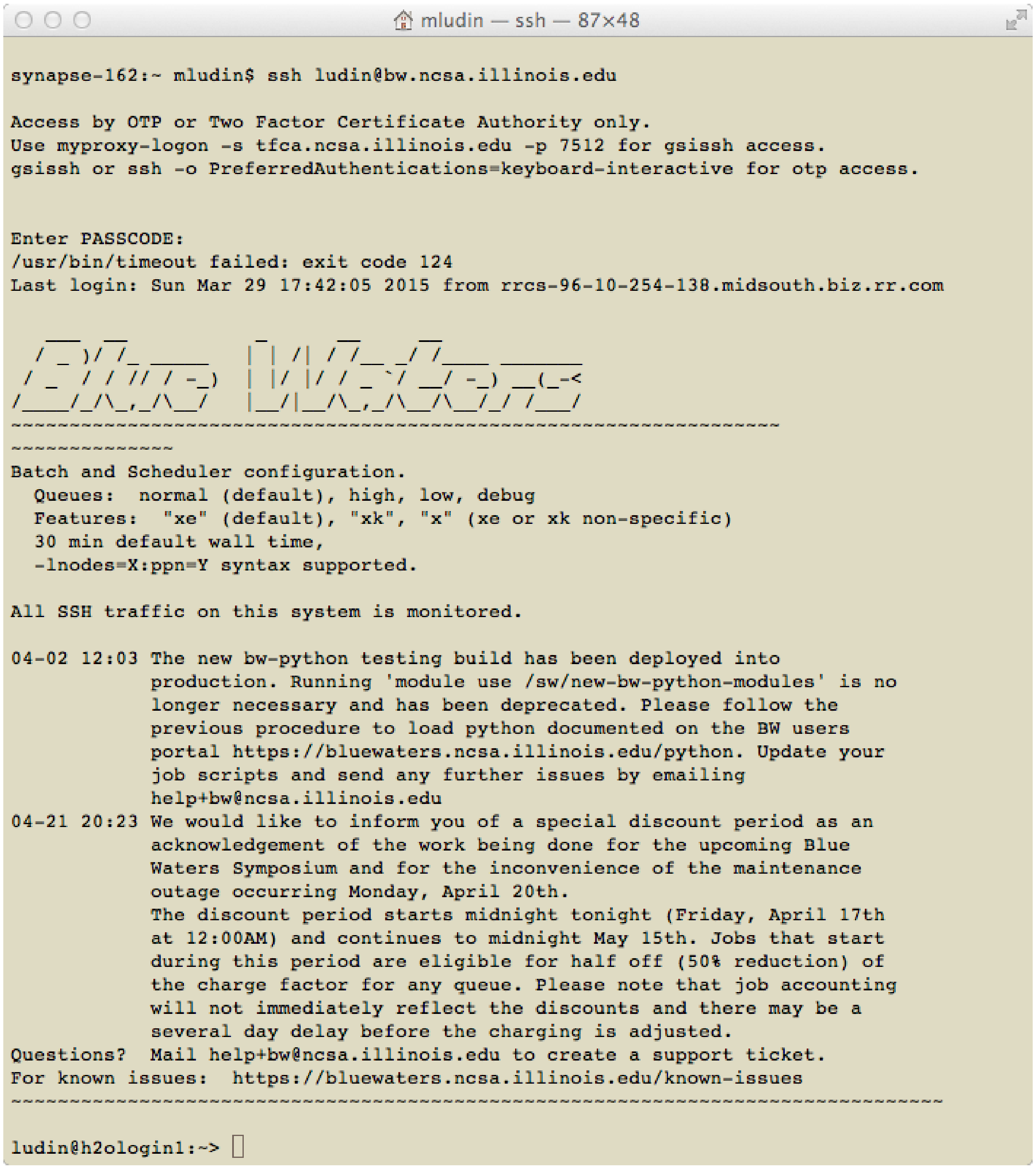
Your BW password

BW hostname: `bw.ncsa.illinois.edu`

Example:

```
$ ssh ludin@bw.ncsa.illinois.edu
```

Output: (its long, so see next page)



mludin — ssh — 87×48

```
synapse-162:~ mludin$ ssh ludin@bw.ncsa.illinois.edu

Access by OTP or Two Factor Certificate Authority only.
Use myproxy-logon -s tfca.ncsa.illinois.edu -p 7512 for gsissh access.
gsissh or ssh -o PreferredAuthentications=keyboard-interactive for otp access.


Enter PASSCODE:
/usr/bin/timeout failed: exit code 124
Last login: Sun Mar 29 17:42:05 2015 from rrcs-96-10-254-138.midsouth.biz.rr.com

   ___  __           _      __     __
  / _ )/ /_ _____   | | /| / /__ _/ /____ _______
 / _  / / // / -_)  | |/ |/ / _ `/ __/ -_) __(_-<
/____/_/\_,_/\__/   |__/|__/\_,_/\__/\__/_/ /___/
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~
Batch and Scheduler configuration.
  Queues:  normal (default), high, low, debug
  Features:  "xe" (default), "xk", "x" (xe or xk non-specific)
  30 min default wall time,
  -lnodes=X:ppn=Y syntax supported.

All SSH traffic on this system is monitored.

04-02 12:03 The new bw-python testing build has been deployed into
            production. Running 'module use /sw/new-bw-python-modules' is no
            longer necessary and has been deprecated. Please follow the
            previous procedure to load python documented on the BW users
            portal https://bluewaters.ncsa.illinois.edu/python. Update your
            job scripts and send any further issues by emailing
            help+bw@ncsa.illinois.edu
04-21 20:23 We would like to inform you of a special discount period as an
            acknowledgement of the work being done for the upcoming Blue
            Waters Symposium and for the inconvenience of the maintenance
            outage occurring Monday, April 20th.
            The discount period starts midnight tonight (Friday, April 17th
            at 12:00AM) and continues to midnight May 15th. Jobs that start
            during this period are eligible for half off (50% reduction) of
            the charge factor for any queue. Please note that job accounting
            will not immediately reflect the discounts and there may be a
            several day delay before the charging is adjusted.
Questions?  Mail help+bw@ncsa.illinois.edu to create a support ticket.
For known issues:  https://bluewaters.ncsa.illinois.edu/known-issues
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

ludin@h2ologin1:~> 
```

Note: Now you could use commands such as `ls`, `cd`, `mkdir`, `pwd`, etc. on Blue Waters.

Copying files to/from Blue Waters:

Copying files **from** Blue Waters is straightforward and follows the pattern established above. Copying files **to** Blue Waters is a bit complicated. To copy anything to Blue Waters you must first `ssh` into BW, then use the `scp` command there. One reason you have your accounts on the Shodor server is to use it as a gateway between your local computer and Blue Waters. Here is an example for copying a file from your local machine to Blue Waters:

1. On your local machine (or on `login.shodor.org`), copy the file from your local machine to `login.shodor.org` using `scp`.
2. On Blue Waters, copy the file from `login.shodor.org` to Blue Waters using `scp`.

Note that the following will NOT work:

1. (Will not work) On your local machine, copy the file from your local machine to Blue Waters.

Note that the following will also NOT work:

1. (Will not work) On `login.shodor.org`, copy the file from `login.shodor.org` to Blue Waters.

Additional Resources:
Useful Unix intro+tutorial (web tutorial): highly recommended for Unix noobs
http://www.ee.surrey.ac.uk/Teaching/Unix/unixintro.html

Software carpentry (video lectures): Highly recommended for scientist-programmers
http://software-carpentry.org/
Some sections are particularly relevant to this internship program, with links to videos:

- The Shell
- Version Control (using Subversion – svn)
- Testing
- Python
- Systems Programming (from Python)
- Classes and Objects (object oriented programming)
- Make
- Matrix programming (linear algebra, NumPy)
- Regular expressions

LinuxCommand.org:
http://linuxcommand.org/lc3_learning_the_shell.php

Unix tutorials point:
http://www.tutorialspoint.com/unix/unix-getting-started.htm