Managing Projects in a Unix Environment

A brief workshop
by
John E. Kerrigan, Ph.D.

Table of Contents

Introduction. This workshop is aimed at individuals who are new to the unix environment and those folks who have some unix experience who might need to reinforce their skills. This workshop is not intended to be a system administration primer (i.e. there is no instruction here on how to use or configure the various unix shell environments.). The workshop is written for the students and faculty/staff of the University of Medicine and Dentistry of New Jersey. We begin with a brief overview of system structure at UMDNJ.

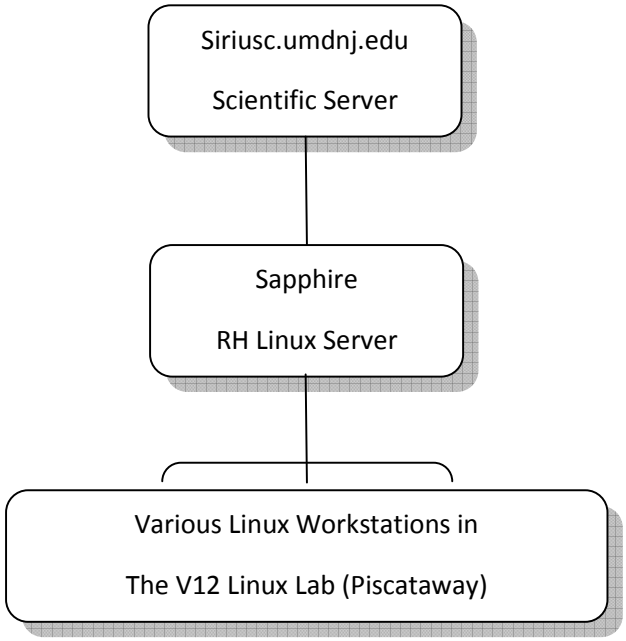


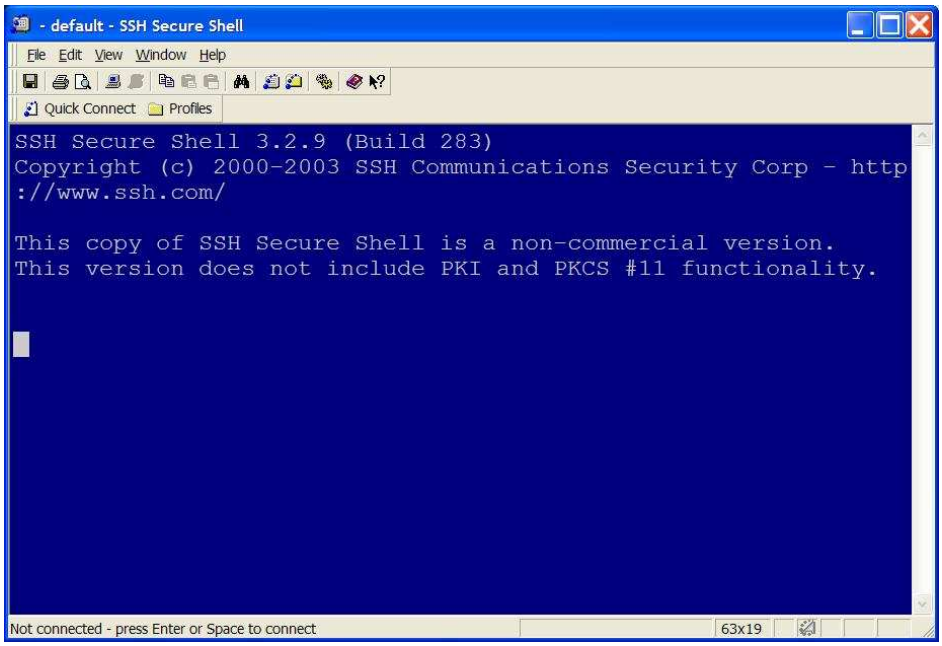
Figure 1. Relationship of RedHat linux workstations to campus servers. Sapphire is the legacy server which contains user accounts. Most accounts are now NFS mounted to the linux workstations from the scientific server.

You must have an umdnj e-mail account before you can obtain an account on the scientific server. Once you have an account on the scientific server, you can get account access to the linux machines in V12.

GCG/SeqWeb have been replaced by EMBOSS. You may access emboss from http://siriusc.umdnj.edu/emboss with your web browser.

PART 1. FUN WITH FILES AND DIRECTORIES.

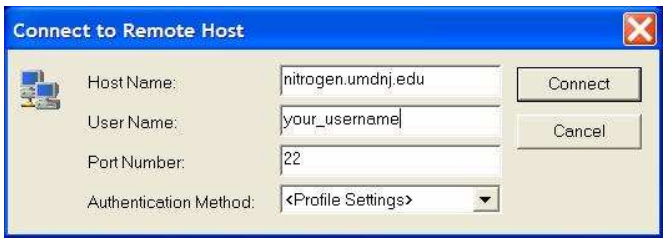
If you are in the V12 lab in Piscataway, go to a linux workstation and login using your account credentials (i.e. your username and password). Or if you are based in Newark, login to a lab PC using your UMDNJ account credentials and double-click on the Communication icon to reveal the communication program tools available. You should see a program called SSH Client. OpenSSH. You see the following window …



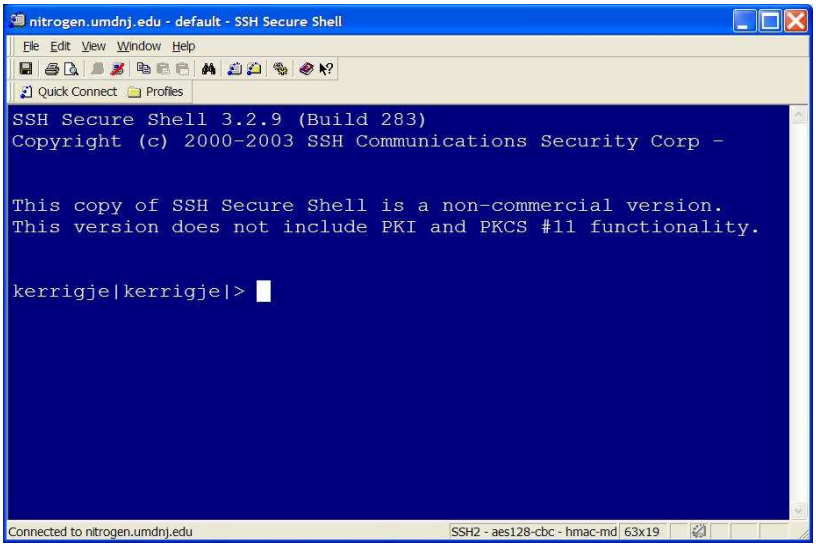
The linux machines in V12 are named for the first 14 elements in the periodic chart (i.e. hydrogen, helium … etc.).
http://en.wikipedia.org/wiki/Periodic_table_(big)

Linux boxes names in V12:
1: hydrogen – dedicated to network administration and testing of new software
2: helium – dedicated to network administration and testing of new software
3: lithium
4: beryllium
5: boron
6: carbon
7: nitrogen
8: oxygen
9: fluorine
10: neon
11: sodium
12: magnesium
13: aluminum
14: silicon

Click on Quick Connect and enter the name of one of the linux hosts for Host Name: and your username for User Name:

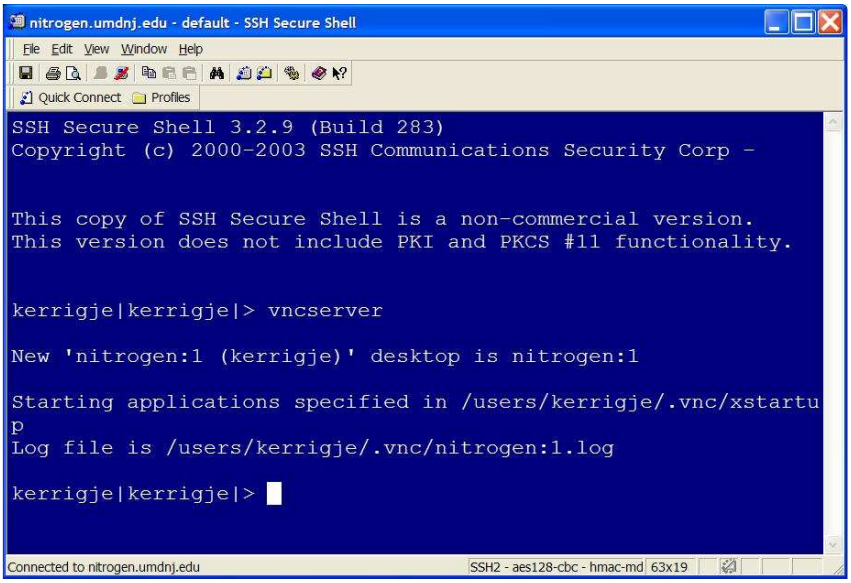


You will be asked to accept the host key (click on Yes) and then you will be prompted for your password. When you are logged in, you will see a command prompt similar to …

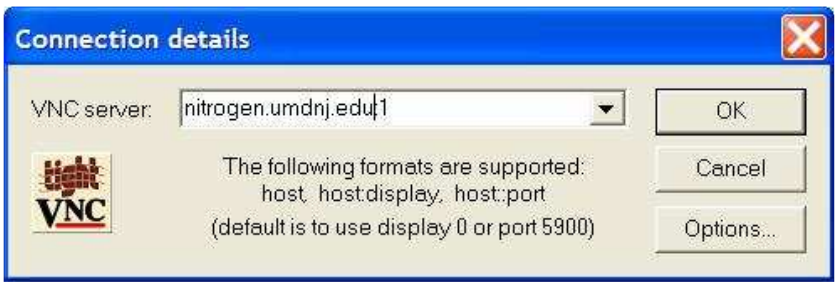


To setup a VNC session, we must first run vncserver command on the host we intend to use.

So, in your SSH shell, type the command "vncserver" and hit <Enter> key. You will be asked to establish a vncserver password for this and all future sessions if this is your first use of vnc. You will notice as in the example below, you are assigned a display # In the example below we get "nitrogen:1" where "nitrogen" is the name of the linux workstation and the number 1 is the display #. We will need this information and the vnc password to login to this workstation with VNC client from the Windows desktop.
[Note: You may use the –geometry flag to set the size of your VNC window to match the size of your display in pixels. For example, **vncserver –geometry** 1920x1200 would setup the VNC window for a 1920x1200 pixel display.]

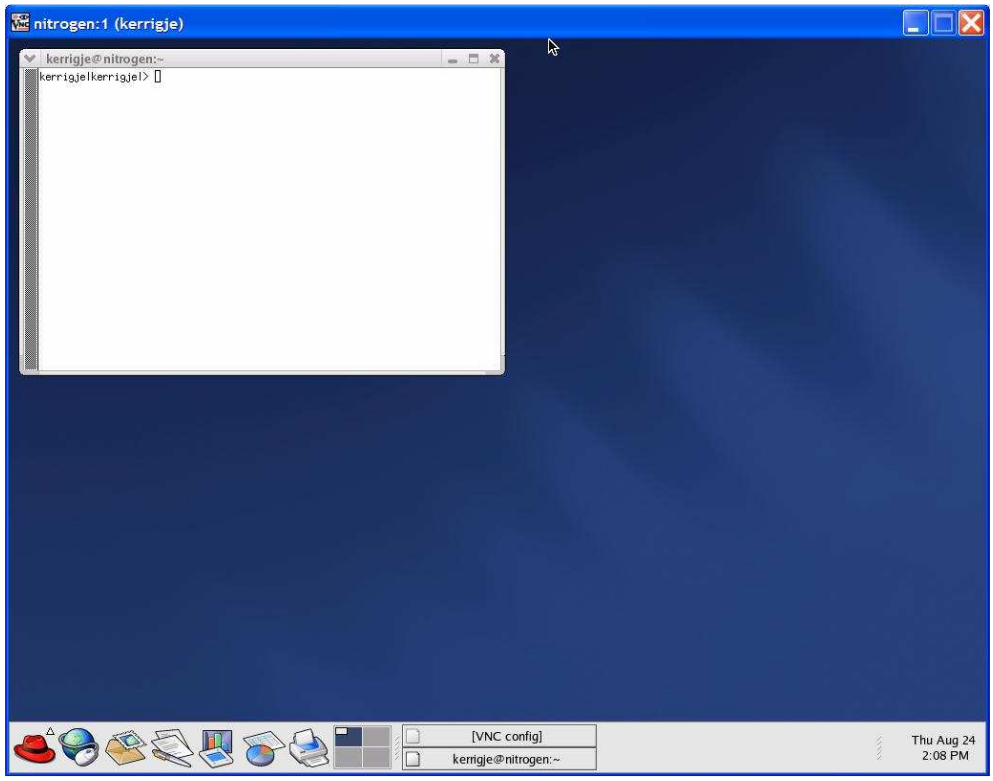


On your Windows machine, open VNC client from the Communications folder. You will see



Under VNC server: enter the linux hostname.umdnj.edu:[display #] as shown above, then click OK. Your linux hostname and display # might be different! You will next be prompted for your password …

Enter your vnc password. Now you will see a reproduction of the RedHat linux desktop …



The example above has a unix shell open already. To open a unix command shell, using your

mouse, right-click anywhere on the desktop screen open space and select New Terminal to open a new unix shell. You will need the unix shell to work this tutorial. When you are finished with the tutorial, click on the red X in the upper right-hand corner of the window to close the VNC client session.
[**VERY IMPORTANT**: Always remember to log back into the workstation using SSH to kill your vnc server session! For our example you would ssh to nitrogen and use the following command: **`vncserver -kill :1`** to kill the vncserver instance you created on it.]

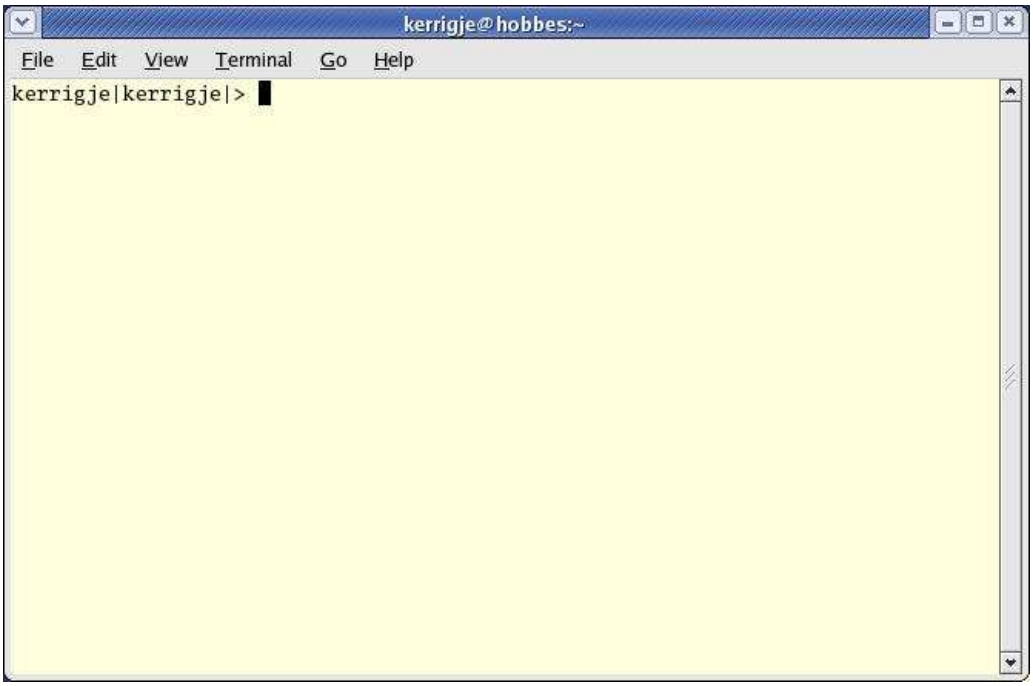


Figure 2. The linux bash shell (Fedora).

An important note about unix shells in graphical user environments. Notice that the cursor in the *unix shell* illustrated in Figure 2 above is a filled-in box. This means that the window is active. If the box were open, this would mean that the window is inactive. You can type commands from your keyboard until you drop and nothing will happen! To activate a unix shell in a graphical user environment, place your mouse cursor inside the shell's window and/or click inside of it. You will notice that the shell (winterm) cursor becomes a filled-in box. Once the shell cursor is filled-in, it is activated and ready to receive input from the keyboard. This cursor behavior is consistent between unix and linux platforms.

Prompts: For example in the bash linux sample given in Figure 4, you will notice the kerrigje|kerrigje| > prompt is different from prompts you may have encountered on other systems. Shell prompts are customizable! We will not go into this detail here. The prompt given in the linux bash sample provides the "username|working directory| > ". The format of the prompt is a matter of user preference.

The Man Pages: To obtain more information about any of the commands discussed in this workshop, just type "**man** *command_name*". Type "q", to quit the man page program. For example, to obtain more information about the "whoami" command discussed below, type "**man** whoami" followed by the <enter> key.

Who am I? Where am I?

Okay, easy enough. Unix has some really cool commands. Let's play!

So, who are you, anyway? Type the following command in the unix shell.

```
kerrigje|kerrigje|> whoami
kerrigje
kerrigje|kerrigje|>
```

your_username (unix should respond with your username)

The **whoami** command is a good command to use to identify your login id if you happen to use multiple logins with multiple unix shells open at once.

If you need to find more about other user based on his login id you can run **finger** command.

finger username

Who is logged-in to the system? Use the **w** command .

```
kerrigje|kerrigje|> w
 09:28:32  up 10 days, 23:08,  1 user,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU  WHAT
kerrigje pts/0    130.219.4.161     9:28am  0.00s  0.07s  0.01s  w
kerrigje|kerrigje|>
```

Will produce a tabular output of those individuals logged-in to your particular linux workstation. This is a useful command to use to monitor who is on the workstation along with what jobs he/she might be running *interactively* or in *batch*.

What processes are running? Use the **top** program. The top program will give a listing of all running processes (system and user).

```
kerrigje|kerrigje> top
14:22:10  up 80 days,  4:34,  5 users,  load average: 0.00, 0.00, 0.00
83 processes: 78 sleeping, 5 running, 0 zombie, 0 stopped
CPU states:  cpu    user    nice  system    irq  softirq  iowait    idle
           total    0.0%    0.0%    0.0%   0.0%     0.0%    0.0%  100.0%
Mem:  1018880k av,  968732k used,   50148k free,       0k shrd,  136680k buff
                    666964k actv,  133184k in_d,   17588k in_c
Swap: 2040244k av,  107160k used, 1933084k free                   493204k cached

  PID USER     PRI  NI  SIZE  RSS SHARE STAT %CPU %MEM   TIME CPU COMMAND
    1 root      15   0   108  108    56 S     0.0  0.0   0:03   0 init
    2 root      15   0     0    0     0 SW    0.0  0.0   0:00   0 keventd
    3 root      15   0     0    0     0 SW    0.0  0.0   0:00   0 kapmd
    4 root      34  19     0    0     0 SWN   0.0  0.0   0:00   0 ksoftirqd/0
    7 root      15   0     0    0     0 SW    0.0  0.0   0:00   0 bdflush
    5 root      15   0     0    0     0 SW    0.0  0.0   0:45   0 kswapd
    6 root      15   0     0    0     0 SW    0.0  0.0   0:30   0 kscand
    8 root      15   0     0    0     0 SW    0.0  0.0   0:01   0 kupdated
    9 root      25   0     0    0     0 SW    0.0  0.0   0:00   0 mdrecoveryd
   17 root      25   0     0    0     0 SW    0.0  0.0   0:00   0 scsi_eh_0
```

Top gives a table of running processes and updates the information on the screen in real time. The top program is very useful for tracking programs (your own and others) that are running on the system. Gives cpu usage stats, memory use, etc. To quit top, hit the "q" key.

What if I just want to track a specific job running in background on my machine? Use the **ps** and **grep** commands. For example, the following command will track any running sybyl.exe jobs.

```
ps –ef | grep sybyl.exe
```

The **ps –ef** command gives you all processes. We pipe (|) this output to **grep**, which searches for the "sybyl.exe" string. The **grep** command is a useful string search tool.

```
kerrigje|kerrigje> ps -ef | grep sybyl.exe
kerrigje      64664      64624  0 14:09:25 pts/1    0:02 /products/tripos/sybyl6.9/bin/sybyl.exe
kerrigje      64744      64457  0 14:09:52 pts/0    0:00 grep sybyl.exe
```

The first number in the output is the process ID number (PID). The sybyl.exe process above is interactive. Processes that are running in background will be indicated by a question mark (?) in the output. You may stop your own processes using the "kill" command. For example …

```
kill 64664
```

… will kill the sybyl.exe process given in the example above.

Where are you in the directory structure of the system? Use the **pwd** command, which stands for **p**rint **w**orking **d**irectory.

```
kerrigje|kerrigje> pwd
/users/kerrigje
```

What is in your directory? Use the **ls** command (**ls** stands for **l**i**s**t). The ls command has a number of useful flags associated with it. We will cover a few of the most useful flags. Try the following for fun.

ls with no flags:

```
kerrigje|kerrigje > ls
Desktop              dumpster            mail                ppe_icu.mol2
SPDBV                kerrigje.outbox     modeling            public_html
course               look_output.log     nsmail
s2003_students.txt
dos                  mac                 pcpine
```

The **–F** flag helps us distinguish between directories (indicated with a / character) and files:

```
kerrigje|kerrigje > ls -F
Desktop/             kerrigje.outbox     nsmail/
SPDBV/               look_output.log     pcpine/
course/              mac/                ppe_icu.mol2
dos/                 mail/               public_html/
dumpster/            modeling/           s2003_students.txt
```

The **–lF** flag (the l stands for "long" list) gives us more information about the individual files and directories.

```
kerrigje|kerrigje > ls -lF
total 932
drwxr-xr-x   2 kerrigje   gcg             96 Feb  4  2002 Desktop/
drwxr-xr-x   7 kerrigje   gcg             96 Sep  6  2002 SPDBV/
drwxr-xr-x   2 kerrigje   gcg           1024 May 14  2002 course/
drwx--x--x   2 kerrigje   gcg             96 Jan 15  2002 dos/
drwxr-xr-x   2 kerrigje   gcg             96 Feb  4  2002 dumpster/
-rw-r--r--   1 kerrigje   gcg             63 Feb  4  2002 kerrigje.outbox
-rw-r--r--   1 kerrigje   gcg          16017 Mar  6  2002 look_output.log
drwx--x--x   2 kerrigje   gcg             96 Jan 15  2002 mac/
drwx------   2 kerrigje   gcg             96 Jan 15  2002 mail/
drwx------   2 kerrigje   gcg           1024 Mar 20 08:52 modeling/
drwx------   2 kerrigje   gcg             96 Feb  4  2002 nsmail/
drwxr-x---   3 kerrigje   gcg             96 Feb  4  2002 pcpine/
-rw-r--r--   1 kerrigje   gcg         455122 Jul 30 14:30 ppe_icu.mol2
drwxr-xr-x   4 kerrigje   gcg           1024 Jul 30 13:50 public_html/
-rw-r--r--   1 kerrigje   gcg             52 Jan 14  2003 s2003_students.txt
```

(Special Note
ll is a common *alias* for **ls –l** on many unix systems.)

The meaning of the **–l** output :

```
-rw-r--r--   1 kerrigje   gcg             52 Jan 14  2003 s2003_students.txt
```

permissions **links** **owner** **group** **size** **last access date** **filename**

Reading access permissions strings:

-	rw-	r--	r--
file type	**owner**	**group**	**everyone else (other)**

file type
A dash (-) means that this is just a file.
A "d" means that this is a directory.

`rw-` The *owner* (aka **u**ser) has **r**ead and **w**rite access to this file, but no e**x**ecute (**x**) access.
`r--` The *group* has read access only.
`r--` Everyone else (i.e. the outside world) has read access only to this file.

File access permissions are an important aspect of security on all unix machines. We will discuss file permissions/ownership and how to change these permissions in a moment.

One last useful **ls** flag. The **–a** flag will display all files (the .dot or hidden files).

```
kerrigje|kerrigje > ls -aF
./                              .neditdb
../                             .mozilla/
.Latitude/                      .noninteractive_login
.Xauthority                     .profile
.addressbook                    .rhosts
.addressbook.lu                 .seqlab-helios/
.cshrc                          .seqlab-history
.desktop-kerrigje|kerrigje.UMDNJ.EDU/     SPDBV/
.desktophost/                   dumpster/
.dtEnvPref                      kerrigje.outbox
.flexlmrc*                      look_output.log
.forward                        mac/
.history*                       mail/
.kshrc                          modeling/
.last_login                     nsmail/
.login                          pcpine/
.look/                          ppe_icu.mol2
.look_history                   public_html/
.moe-rc                         s2003_students.txt
```

The output above is listed in two columns (the –F portion of the flag just adds the / for the directories). Some of these hidden files are very important for the management of shell behaviour and configuration. The more important files are:

.login
.profile
.kshrc (.cshrc, .tshrc, etc.)
.history (command history)
.mozilla/ (Your Netscape directory)

The Netscape directory is an important directory to keep track of. If you use Netscape on the workstation, you should periodically check the .mozilla/profile_name/Cache directory to make sure that it is empty. Webpage content can consume your disk quota in a hurry!

Let's check the contents of our .mozilla/profile_name/Cache directory.

```
kerrigje|kerrigje > ls -F .mozilla/profile_name/Cache
0F/        10/        11/        1B/        1C/        index.db
```

If you just get a prompt with no other information, then the directory is empty. If you see a bunch of subdirectories as in my .mozilla/profile_name/Cache directory above, you may already have a quota problem! The subdirectories in my .mozilla/profile_name/Cache directory above were produced from one visit to http://www.cnn.com

How do I remove those unwanted files and subdirectories clogging up my disk space?

The **rm** command (**rm** stands for remove) is the answer to your file management woes! **CAUTION**: Make certain that you back-up your valuable files before removing them. In unix, removing files is irreversible! (Unless they were backed-up the day before by system administration)

First, use the **cd** command to change directory into the .mozilla/profile_name/Cache directory. Type "**cd** *dirname*"

```
kerrigje|kerrigje > cd
.mozilla/profile_name/Cache
kerrigje|kerrigje > pwd
/users/kerrigje/.mozilla/profile_name/Cache
```

We used **cd** followed by **pwd** just to make sure we moved into the correct directory.

Removing files is easy. Just type **rm** *filename* and it's gone! You may use the asterisk (*) wildcard to remove multiple files all at once. Removing whole directories along with their contents (even subdirectories!) requires a **R**ecursive remove.

NOTE: In Unix/Linux environment file/directories names are case sensitive. Thus ***Filename*** *and* ***filename*** *are two different files.*

To remove a subdirectory, type **rm –R** *dirname*.

To remove all subdirectories and files in your current working directory, type "**rm –R ***" **CAUTION**: Very dangerous! Never use this command in your home or root directory!

To move up a directory, type **cd ..** (The cd dot dot)

```
kerrigje|kerrigje > cd ..
kerrigje|kerrigje > pwd
/users/kerrigje/.mozilla
```

If you ever get lost in your directory structure, just type **cd** <enter> and this command will return you to your home directory!

```
kerrigje|kerrigje > pwd
/users/kerrigje/.mozilla/profi
le_name/Cache
kerrigje|kerrigje > cd
kerrigje|kerrigje > pwd
/users/kerrigje
```

Use the **mkdir** command to make your own directories. Let's make a directory called"myproject".

```
kerrigje|kerrigje > mkdir
myproject kerrigje|kerrigje > ls
-F
Desktop/              look output.log      pcpine/
SPDBV/                mac/                ppe_icu.mol2
course/               mail/               public_html/
dos/                  modeling/           s2003_students.txt
dumpster/             myproject/
kerrigje.outbox       nsmail/
```

NOTE: The **rmdir** command will **remove** directories; however, the directory must be empty before it will work!

Let's make a file that we can store in myproject. Type the following command.

```
kerrigje|kerrigje > ls -aF > dirlist.out
```

You have just learned a new concept! Whenever you type most commands in unix, the output (STDOUT) goes to the terminal display by default (some commands output to a file or device by default). You can always direct STDOUT to a text file using the redirection operator ">". (The opposite of this is the "<" operator, which directs STDIN to the program) In this command, we have issued the familiar request for the listing of all files and directories in our home directory and directed the output to the file "dirlist.out", which is created automatically. This is why you will not see any output on the screen. Lets examine the contents of this new file using the "**more**" program.

```
kerrigje|kerrigje > more dirlist.out
./
../
.Latitude/
.Xauthority
.addressbook
.addressbook.lu
.cshrc
```
… etc

The space bar (or the "**f**" key) advances the output. Use the "**b**" key to go backwards. Use the "**q**" key to quit the **more** program.

This file contains a lot of information. Let's say I want to just look at the first ten lines of data in this file. We can use the "**head**" command.

```
kerrigje|kerrigje > head dirlist.out
./
../
.Latitude/
.Xauthority
.addressbook
.addressbook.lu
.cshrc
.desktop-130.219.171.132/
.desktop-130.219.34.66/
```

The head program prints out the first ten lines by default. To change this behavior, just use the - # flag (where # is the number of lines you wish to examine).

```
kerrigje|kerrigje > head -5 dirlist.out
./
../
.Latitude/
.Xauthority
```

Use the **tail** command to view the last ten lines of a data or log file. Like head, ten lines is the default for tail. Use the -# flag to view more lines or fewer lines. If you are viewing a log file from a running computation, you will find yourself using this command very often to check onthe progress of your job.

```
kerrigje|kerrigje > tail -15 dirlist.out
course/
dirlist.out
dos/
dumpster/
kerrigje.outbox
look_output.log
mac/
mail/
modeling/
myproject/
nsmail/
pcpine/
ppe_icu.mol2
public_html/
s2003_students.txt
```

Now that we have created a file, let's copy the file to our new directory. Use the **cp** command to copy files.

```
kerrigje|kerrigje > cp dirlist.out myproject
```

Copies the file "dirlist.out" to the subdirectory "myproject". The file dirlist.out is present in both our home directory */users/your_username* and in myproject *(/users/your_username/myproject*). There is no need to use the full path in the command because myproject resides in */users/your_username* which happens to be our current working directory.

How to get a listing of other subdirectories without having to "cd" into them. Let's say we want to check the contents of our "myproject" directory from home.

```
kerrigje|kerrigje > ls -F myproject
dirlist.out
```

Now let's add a new directory to myproject called *newstuff* and move our dirlist.out file to this newstuff directory. We will use the **mv** command to move the file. Study the commands below.

```
kerrigje|kerrigje > mkdir myproject/newstuff
kerrigje|kerrigje > mv dirlist.out myproject/newstuff
kerrigje|kerrigje > ls -F
Desktop/              look output.log      pcpine/
SPDBV/                mac/                ppe_icu.mol2
course/               mail/               public_html/
dos/                  modeling/           s2003_students.txt
dumpster/             myproject/
kerrigje.outbox       nsmail/
kerrigje|kerrigje > ls -F myproject/newstuff
dirlist.out
kerrigje|kerrigje > ls -F myproject
dirlist.out   newstuff/
```

Notice what happens when we use the **mv** command. The dirlist.out no longer resides in our home directory because we have moved it myproject/newstuff. The dirlist.out file now resides in myproject and myproject/newstuff. You may also move files in the reverse direction (a directory back) using "**../**". For example, to move dirlist.out from newstuff to myproject do the following…

cd myproject/newstuff
mv dirlist.out ../
You may use the * wildcard with the **cp** and **mv** commands to copy and move multiple files. For example, "**cp** *.pdb *myproject*" would copy all files with the .pdb extension into the directory myproject.

You may use the **cp** command to copy files to different volumes; however, you may not use the **cp** command to copy directories.

Compressing files and archiving data
Perhaps the most important section of this workshop!
You should get into the habit of compressing data and archiving data. We will use two very nice unix tools: **gzip** (compression) and **tar** (stands for **T**ape **AR**chive). Lets cd into our myproject directory.

```
kerrigje|kerrigje > cd /users/your_username/myproject
```

Get a long listing of the directory contents.

```
kerrigje|kerrigje > ls -lF
total 3
-rw-r--r--    1 kerrigje gcg          825 Aug 13 08:58 dirlist.out
drwxr-xr-x    2 kerrigje gcg           96 Aug 13 09:08 newstuff/
```

The file dirlist.out is 825 kb in size. Your size might be slightly different depending upon your filesystem.

```
kerrigje|kerrigje > gzip -c dirlist.out > dirlist.out.gz
kerrigje|kerrigje > ls -lF
total 4
-rw-r--r--    1 kerrigje gcg          825 Aug 13 08:58 dirlist.out
-rw-r--r--    1 kerrigje gcg          443 Aug 13 10:01 dirlist.out.gz
drwxr-xr-x    2 kerrigje gcg           96 Aug 13 09:08 newstuff/
```

The **gzip –c** command directs the compressed file to STDOUT, which means that we must use the "**>**" operator to direct STDOUT to a file (dirlist.out.gz). Note that the file has been reduced to about half its size. This whole process can be automated w/o the –c flag! For maximum file compression use the -9 flag in gzip. For example:

```
gzip -9 myfile
```

Gives … myfile.gz

Use the **gun zip** command to uncompress the file. You will compress files so that you can move them to another location for backup. For example, study the following sequence of commands where we move our compressed file to another volume then decompress it with gunzip.

```
kerrigje|kerrigje > mv dirlist.out.gz /users3/kerrigje
kerrigje|kerrigje > cd
kerrigje|kerrigje > ls -F
Desktop/             kerrigje.outbox      pcpine/
SPDBV/               look_output.log      ppe_icu.mol2
course/              mac/                 public_html/
dirlist.out.gz       mail/                s2003_students.txt
dos/                 modeling/
dumpster/            nsmail/
kerrigje|kerrigje > gunzip dirlist.out.gz
kerrigje|kerrigje > ls -lF
total 939
drwxr-xr-x    2 kerrigje gcg           96 Feb  4  2002 Desktop/
drwxr-xr-x    7 kerrigje gcg           96 Sep  6  2002 SPDBV/
drwxr-xr-x    2 kerrigje gcg         1024 May 14  2002 course/
-rw-r--r--    1 kerrigje gcg          825 Aug 13 10:01 dirlist.out
drwx--x--x    2 kerrigje gcg           96 Jan 15  2002 dos/
drwxr-xr-x    2 kerrigje gcg           96 Feb  4  2002 dumpster/
-rw-r--r--    1 kerrigje gcg           63 Feb  4  2002 kerrigje.outbox
```

```
-rw-r--r--    1 kerrigje gcg        16017 Mar  6  2002 look_output.log
drwx--x--x    2 kerrigje gcg           96 Jan 15  2002 mac/
drwx------    2 kerrigje gcg           96 Jan 15  2002 mail/
drwx------    2 kerrigje gcg         1024 Mar 20 08:52 modeling/
drwx------    2 kerrigje gcg           96 Feb  4  2002 nsmail/
drwxr-x---    3 kerrigje gcg           96 Feb  4  2002 pcpine/
-rw-r--r--    1 kerrigje gcg       455122 Jul 30 14:30 ppe_icu.mol2
drwxr-xr-x    4 kerrigje gcg         1024 Jul 30 13:50 public_html/
-rw-r--r--    1 kerrigje gcg           52 Jan 14  2003 s2003_students.txt
```

It's that simple! See! The dirlist.out file is restored to its original grandeur!

Using the **–c** flag gave us more control. You will notice that the original file is still intact. Had we used "**gzip** dirlist.out" with no flags or redirection, the original file would have been replaced by dirlist.out.gz!

Lets say you have a finished project that you want to backup and store in an archive. Wouldn't it be nice to archive and compress everything all in one command? Here is the answer to your prayers!

Go into your project directory (e.g. myproject). Issue the **tar** command as shown below.

```
kerrigje|kerrigje > cd myproject
kerrigje|kerrigje > tar -clf - . | gzip > ../myproject.tar.gz
kerrigje|kerrigje > cd ..
kerrigje|kerrigje > ls -lF
total 7
drwx------    2 kerrigje gcg         2048 Jun 10 09:56 gcg/
drwxr-xr-x    3 kerrigje gcg           96 Aug 13 10:05 myproject/
-rw-r--r--    1 kerrigje gcg          639 Aug 13 10:12 myproject.tar.gz
```

The **tar** command creates (the **–c** flag) the archive. The "-" passes the filename requirement for the **tar** command along to **gzip** to assign the filename there. The "|" (pipe) pipes the output from **tar** to **gzip**, which compresses the data and sends it to the file *myproject.tar.gz* in the next directory up (hence the "../"). What you have done is create a special file called a .tar.gz file (a compressed tape archive file). These files go by the .tgz extension as well. CAUTION: Check your quota to make sure that you have enough quota to accommodate the archive file that you are creating!

If you want to preserve the full directory structure, use the following 2-step procedure to create your compressed archive:

First, cd .. out of myproject.

```
tar cvf myproject2.tar myproject
gzip -9 myproject2.tar
```

Or, to do this in one step use the 'pipe' "|"…

tar cvf – myproject **|** **gzip** > myproject2.tgz

To decompress or open a tape archive that has been compressed, just use **gunzip**. To open a tape archive use the following tar command:

gunzip `file.tar.gz` (gives file.tar)

tar –xvf `file.tar` (**-x** *extract* releases the archive)

To combine these two elements into one command line, use the 'pipe' "|"

gunzip –dc `file.tar.gz` | **tar –xvf -**

The **–d** flag to **gunzip** is for "decompress" and the **–c** means stdout (standard output). The dash "-" after the **tar** command tells tar to accept the filenames (& directory names if any) from the pipe.

Special Note for Linux Users

The tar command in linux has gzip/gunzip capability built-in! In linux, the sequence above can be accomplished with one short command using the **–z** flag.

tar –zxvf file.tar.gz

Likewise, creating .tar.gz (.tgz) files is a breeze in linux.

tar –zcvf - . > ../file.tgz

Accomplishes the same task as the **tar –clf - . | gzip** > ../file.tgz command!

Use the **–t** flag to get a listing (tabulation) of files in your tar file. You must gunzip .tar.gz or .tgz archives.

```
kerrigje|kerrigje > tar -tvf myproj.tar
rwx------ 5623/302 dir           Aug 14 09:12 2003 ./
rw-r--r-- 5623/302           443 Aug 13 08:58 2003 dirlist.out.gz
rw-r--r-- 5623/302             0 Aug 14 09:12 2003 myproj.tar
rwxr-xr-x 5623/302 dir           Aug 13 09:08 2003 newstuff/
rw-r--r-- 5623/302           825 Aug 13 08:05 2003 newstuff/dirlist.out
```

The compressed files that you have made are readable by the **WinZip** program on Microsoft Windows PC's (or **StuffIt** on the Mac). You may obtain **WinZip** from ISTWEB's FTP archives (www.winzip.com)

Protecting your data using file permissions.

You should get into the habit of protecting your data and directories from intrusion by using file permissions. Remember file permissions? A brief overview of what was discussed earlier:

```
-rw-r--r--   1 kerrigje   gcg              52 Jan 14  2003 s2003_students.txt
```
permissions **links** **owner** **group** **size** **last access date** **filename**

Reading access permissions strings:

```
-          rw-          r--          r--
```
file type **owner** **group** **everyone else (other)**

file type
A dash (-) means that this is just a file.
A "d" means that this is a directory.

`rw-`	The *owner* (aka **u**ser) has **r**ead and **w**rite access to this file, but no e**x**ecute (**x**) access.
`r--`	The *group* has read access only.
`r--`	Everyone else (i.e. the outside world) has read access only to this file.

We use the **chmod** command to alter file permissions for any given file or directory.

```
kerrigje|kerrigje > ls -lF
total 7
drwx------    2 kerrigje gcg         2048 Jun 10 09:56 gcg/
drwxr-xr-x    3 kerrigje gcg           96 Aug 13 10:31 myproject/
-rw-r--r--    1 kerrigje gcg          639 Aug 13 10:12 myproject.tar.gz
```

Looking at the contents of my directory above, I see that my gcg directory gives me ***r**ead*, ***w**rite* and ***ex**ecute* permission, the *groups* have no permissions and *other* has no permissions. The permissions set for the gcg directory is nice and secure.

The permissions for the myproject directory give *read* and *execute* permission to group and other. This means that folks on the outside can view the contents of the myproject and even worse can steal the files contained in this directory. Although, outsiders cannot alter the files contained in the myproject directory as they do not have "write" permission.

Table 1. Permissions Strings and Numeric Equivalents.

Permissions string	*Decimal Equivalent*
---	0
--x	1
-w-	2
-wx	3
r--	4
r-x	5
rw-	6
rwx	7

When reading a file or directory permission, the order is always: user group other. We will make our myproject directory secure like the gcg directory by using chmod.

```
kerrigje|kerrigje > chmod 700 myproject
kerrigje|kerrigje > ls -lF
total 7
drwx------    2 kerrigje gcg         2048 Jun 10 09:56 gcg/
drwx------    3 kerrigje gcg           96 Aug 13 10:31 myproject/
-rw-r--r--    1 kerrigje gcg          639 Aug 13 10:12 myproject.tar.gz
```

Table 2. Common Permissions and Numeric Equivalents.

Permission	*Numeric equivalent*
rwx------	700
rw------	600
rw-r--r--	644
rwxr-xr-x	755

Use 700 for directories and programs, 600 and 644 for files. The 755 permission is commonly used for "public" websites.

We can secure the tar.gz file in the same manner.
You may use the * wildcard with **chmod**.

Part 2. Communication and File Transfer

Secure Shell (ssh)
Secure FTP (sftp)

Why secure shell and sftp?

The problem with the **telnet** and **ftp** programs is that your password is sent to the host in clear text. This means that anyone who is watching can intercept your password! (Exception: anonymous ftp) With secure shell and sftp, all communication is encrypted. **As a rule, never use telnet or ftp. Always use ssh and sftp!**

Checking Quotas from the linux machine.

The best way to login to the host to check your quota while logged into a linux workstation is to use ssh. Use ssh hostname. You may be asked to accept an encryption key if this is the first time that you have logged into the host from specific machine that you are using. Just answer "yes" and proceed. Then you will be asked for your password. Once you are logged into the host, type **quota –v** to check your quota.

```
kerrigje|kerrigje > ssh oxigen.umdnj.edu
Password:
kerrigje|kerrigje [24]> quota –v
Disk quotas for kerrigje (uid 502):
Filesystem     usage  quota  limit    timeleft  files  quota  limit
timeleft
/users       5129412 10000000 10500000              26804      0      0
```

Volume	kb	soft (kb)	hard (kb)	# files	soft	hard

Always check your quotas periodically! If you exceed your memory limits or file limits, you will notice that you can no longer write files. Programs that produce output will give you strange errors if they're nice or even worse give no error at all (In this case you end up with empty data files!). Remember, you can free-up space by clearing out your .mozilla/profile_name/Cache directory if you happen to use Netscape.

You exit an ssh session exactly like a telnet session. Just type, "**exit**" to quit your ssh session.

```
kerrigje|kerrigje > exit
logout
Connection to oxigen.umdnj.edu closed.
```

Being on the same network, we were able to login with just typing ssh siriusc to connect. What about remote systems? We need the full host DNS name. The usual syntax for ssh is ..

ssh –X username @host.school.edu

The –X permits X forwarding (the forwarding of your display address so you can run graphical (openGL) X interface programs from the remote host).

If I'm on the rwja server and want to download the myproject.tar.gz file to my rwja account, I can use the **sftp** program.

```
rwja [25]> sftp kerrigje@oxygen.umdnj.edu
Password:
sftp> ls
drwx------    4 kerrigje gcg          2048 Aug 13 10:12 .
drwxr-xr-x    8 root     scigrp       2048 Aug 12 16:41 ..

drwx------    3 kerrigje gcg            96 Aug 13 10:31 myproject
-rwx------    1 kerrigje gcg           639 Aug 13 10:12 myproject.tar.gz
drwx------    2 kerrigje gcg          2048 Jun 10 09:56 gcg
sftp> get myproject.tar.gz
Fetching /users/kerrigje/myproject.tar.gz to myproject.tar.gz
```

The command shell prompt changes to an `sftp>` prompt. The **sftp** program automatically distinguishes between ascii and binary formats. In fact, binary and ascii are invalid commands in sftp. Here is a summary of the most useful sftp commands:

ls – directory list (no flags supported!)
cd – change directory (**cd** dirname)
get – use to get a single file from the host. (Newer versions of sftp use the get command for single and multiple file transfer.)
mget – use to get multiple files from the host (used with the * wildcard; e.g. "**mget** *.pdb").
put – use to upload a single file to the remote host.
mput – use to upload multiple files to the remote host.
bye – use to exit the sftp program.

Secure Copy – Another useful unix tool that supports encrypted communication is **scp**. Here is the command syntax for scp:

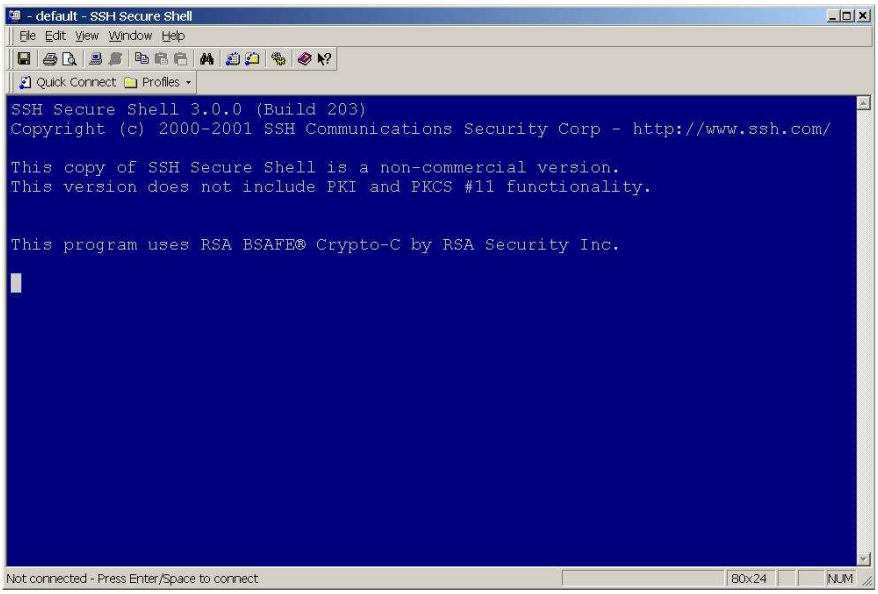
scp *myfile.txt* username@myhost.myschool.edu:/users/username

You will be prompted for your password on the host and *voila* your file will be copied to the directory given in the path following the colon.

You may also use the recursive option and copy whole directories and their contents at once.

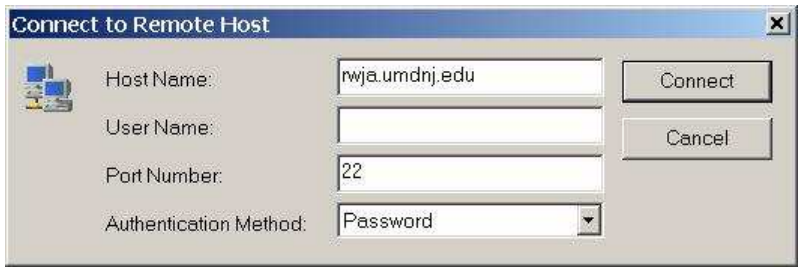
scp –r *dirname* username@myhost.myschool.edu:/users/username

How to use ssh/sftp for Windows. Download SSH from http://istweb.umdnj.edu/ftp/index.cfm and install on your PC. (MAC Users: There are similar programs for the MAC available at http://istweb.umdnj.edu/ftp/index.cfm)



Secure Shell Window.

Click on "Quick Connect". The following dialog appears.



Enter the desired hostname and username, and then click on Connect. Enter your password and you're logged in.

Click on the new file transfer window.
You can change the host directory by going to Operation > Go To Folder …

To download a file, just select it with your mouse and hit the download button. A dialog should appear where you may select a folder on your system to store the file. To upload a file to the remote host, just hit the up arrow and a dialog will appear from which you may select a file from your hard drive, CD-R, or Zip drive.

PART3. Quick and Dirty Editing and String Searching

String searching and the **grep** command . It is often said that you are not a true UNIX nerd until you learn how to use grep! The origin of the name for the **grep** command is a bit sketchy. In the early days of unix, folks did their string searching (strings or search patterns were called *regular expressions*) using an editor called "ed". The user would use the word "global" as a prefix to the ed command and then list the result with the "print" option. Putting this all together gives us **g**lobal/**r**egular **e**xpression/**p**rint. There you have it! The origin of the name for the command **grep**!

Lets use grep for some simple string searching using a new list file.

ls –F > maillist.out to make a new listing of our home directory contents.

```
kerrigje|kerrigje > grep mail maillist.out
maillist.out
```

Our search for the string mail turns up every line in the file that contains that string. The grep command is case sensitive!

```
kerrigje|kerrigje > grep Mail maillist.out
kerrigje|kerrigje >
 Mail/
```

Notice that our search for Mail above gave a different output. To perform a general string search ignoring case use the **–i** flag.

```
kerrigje|kerrigje > grep -i MAIL maillist.out
Mail/
maillist.out
```

Use the **–c** flag to obtain a count of how many matches you have for a string. Using **–c** alone will give case-sensitive matches. Using **–ic** will give the total # of matches ignoring case.

```
kerrigje|kerrigje > grep -c mail maillist.out
1
```

Use the **–n** flag to obtain the line number where each string is located. For example

```
kerrigje|kerrigje > grep -n mail maillist.out
33:maillist.out
```

Quick & Dirty File Editing using grep

Download 6PTI.PDB from the Protein Data Bank (http://rutgers.rcsb.org/pdb/).

Before using PDB files for molecular modeling calculations, we must split out the HETATM records and remove the crystallographic water atoms (There may be special instances where you may want to keep a few water molecules.). Here's a quick way to do it without using a text editor!

```
kerrigje|kerrigje > grep -v '^HETATM' 6PTI.pdb > 6PTI_protein.pdb
kerrigje|kerrigje > tail -20 6PTI_protein.pdb
ATOM    452  CB  CYS    55      11.841   0.436   8.634  1.00 10.51      6PTI 524
ATOM    453  SG  CYS    55      10.271  -0.018   7.900  1.00 10.18      6PTI 525
ATOM    454  N   GLY    56      13.939   0.557  11.358  1.00 14.61      6PTI 526
ATOM    455  CA  GLY    56      15.319   0.828  11.790  1.00 17.33      6PTI 527
ATOM    456  C   GLY    56      16.029  -0.385  12.375  1.00 18.91      6PTI 528
ATOM    457  O   GLY    56      15.443  -1.332  12.929  1.00 21.00      6PTI 529
ATOM    458  N   GLY    57      17.308  -0.138  12.617  1.00 21.95      6PTI 530
CONECT   43   42  453                                                    6PTI 609
CONECT  110  109  302                                                    6PTI 610
CONECT  242  241  421                                                    6PTI 611
CONECT  302  110  301                                                    6PTI 612
CONECT  421  242  420                                                    6PTI 613
CONECT  453   43  452                                                    6PTI 614
CONECT  469  470  471  472  473                                         6PTI 615
CONECT  470  469                                                        6PTI 616
CONECT  471  469                                                        6PTI 617
CONECT  472  469                                                        6PTI 618
CONECT  473  469                                                        6PTI 619
MASTER       38    0    1    2    3    0    0    6  536    0   11    5  6PTIA  7
END                                                                     6PTI 621
```

The **–v** flag tells **grep** to select lines not matching the specified pattern! The ^ means "the beginning of the line". We used the > redirection operator to direct the output to the filename specified.
Now let's say we want to extract the lines containing the water atom coordinates and put that information in a separate file. Here is how to do it.

```
kerrigje|kerrigje > grep 'HOH' 6PTI.pdb > 6PTI_water.pdb
kerrigje|kerrigje > tail -15 6PTI_water.pdb
HETATM  532  O   HOH   233      26.116   2.624  -4.884  0.85 39.87      6PTI 594
HETATM  533  O   HOH   234      27.600   0.000  -3.581  0.50 46.24      6PTI 595
HETATM  534  O   HOH   235       0.583   6.198  -9.923  0.70 45.41      6PTI 596
HETATM  535  O   HOH   236       0.000  19.100  -0.787  0.50 23.82      6PTI 597
HETATM  536  O   HOH   237      14.437  -4.813  13.018  1.00 31.94      6PTI 598
HETATM  537  O   HOH   238       2.772   3.327   8.117  0.80 28.06      6PTI 599
HETATM  538  O   HOH   239       0.679  -0.882   7.824  1.00 44.80      6PTI 600
HETATM  539  O   HOH   240      13.528 -11.124   8.347  0.80 41.45      6PTI 601
HETATM  540  O   HOH   241      13.090  -2.359  12.935  1.00 37.82      6PTI 602
HETATM  541  O   HOH   242       1.723  19.016   0.875  1.00 39.47      6PTI 603
HETATM  542  O   HOH   243      -7.327   6.121  -7.648  1.00 46.65      6PTI 604
HETATM  543  O   HOH   244      -3.312   1.700  -7.546  1.00 54.61      6PTI 605
HETATM  544  O   HOH   245       7.109   2.230  -6.398  1.00 29.94      6PTI 606
HETATM  545  O   HOH   246       6.873  16.063 -10.121  1.00 38.08      6PTI 607
HETATM  546  O   HOH   247       7.523  -8.995   6.523  1.00 43.87      6PTI 608
```

Isn't that just the coolest thing you ever did? Okay, so it's not the coolest thing you ever did. However, it sure beats opening a text editor, highlighting all those lines with the mouse, etc. …