<u>Basic commands</u>

history
    review the most recent commands

cd
    change the working directory to the home directory

cd -
    change the working directory to the next-to-most recent directory

head -1 file.txt
    print the first line of file.txt to the screen

tail -13 file.txt
    print the last 13 lines of file.txt to the screen

tail -f file.txt
    print the file file.txt to the screen continuously. This is useful if some other program continuously appends stuff to file.txt and you want to watch file.txt as it grows. Another way to do this is with **tee**.

wc -l file.txt
    count the number of lines in file.txt

grep "carrot" file.txt
    print out all lines of file.txt containing the string "carrot"

grep "^314" file.txt
    print all lines of file.txt that start with the string "314"

grep "^314.*carrot$" file.txt
    print all lines of file.txt that start with the string "314", then have 0 or more other characters, then end with the string "carrot"

tar cvzf mydir.tar.gz mydir
    tar and gzip the directory "mydir" into an archive called mydir.tar.gz

tar tvzf mydir.tar.gz
    just print what's in mydir.tar.gz without actually extracting the files from the archive

tar xvzf mydir.tar.gz
    extract the files from mydir.tar.gz

ssh -X susie@woof
    log in to the computer woof as user susie. The -X allows you to open graphical things on the local computer. Use -Y instead of -X if you are using a mac.

scp file.txt susie@woof.as.arizona.edu:/home/susie/
    copy the file file.txt over the network into susie's home directory on the computer "woof"

scp susie@woof.as.arizona.edu:/home/susie/file.txt .
    copy the file file.txt located in susie's home directory on the remote computer "woof" to the current working directory.

scp -r mydir susie@woof.as.arizona.edu:/home/susie/stuff/
    copy the entire directory "mydir" into the subdirectory "stuff" of susie's home directory on the remote computer "woof"

pwd
    print the current working directory to the screen (eg where am I)

whoami
    print your user id to the screen

who
    print the list of people who are currently using this computer

ls -lah mydir

print the contents of mydir:
- including permissions, timestamp, and file size
- including files whose names start with a period
- print file sizes in human-readable units (eg K,M) instead of bytes

ls -lrt mydir

print the contents of directory mydir:
- including permissions, timestamp, and file size
- sort in order of decreasing timestamp (-t) then reverse (-r)

locate file.txt

Output the location of file.txt on the current computer. This only works if updateb is being regularly run. If not, you can always do:

find $HOME -name apple.txt 2>/dev/null

Starting in the home directory, march through every subdirectory tree and output any files called apple.txt. If a directory is off-limits because I don't have read permissions, never mind.

diff file1.txt file2.txt

Compare file1.txt and file2.txt and output the lines that differ

awk '{print $1}' file.txt

print the first column of file.txt to the screen

rsync -avur mydir susie@woof.as.arizona.edu:/home/susie/

update the directory "mydir" in susie's home directory on the remote computer woof so that it's synchronized with the "mydir" subdirectory of the current directory.

rsync -avurn mydir susie@woof.as.arizona.edu:/home/susie/

Don't actually do the synchronizing; just print what would get done (dry-run mode).

du -h mydir

Print out how much disk space is taken up by the stuff in directory mydir in human-readable units; if mydir has subdirectories this will recursively march through those as well

df -h /home/

Print out how much disk space is still free on the partition where the /home/ directory lives

top

Print out all the processes that are currently running. Type "q" to exit top.

ps -ef

Print out all processes that are running; this is useful for finding out process ids for processes that you want to kill. How to kill them? Use:

kill 12345

Terminate the process whose process id is 12345. (Sometimes you have to insist by doing kill -9 12345 instead of just kill 12345.)

sftp susie@woof

Log in to the computer woof as user susie. Use commads "get file.txt" or "mget file.txt" to get file from computer woof; use "put file.txt" or "mput file.txt" to put file from your home computer onto computer woof. "exit" to close connection.

ping woof.as.arizona.edu

Bounce packets off the remote computer "woof" to see if it's up.

Fun with output redirection

One of the most useful things about unix is the ability to use the | and > tokens to reroute the output from one command into being to the input to another command. For example, you can use **less** (or **more**) **file.txt** to look at the contents of file.txt. However, you can also use **less** to read the output of another command. Examples:

history | less

Review the most recent commands with **less**

head -100 file.txt | grep "carrot"

Inspect the first 100 lines of file.txt and print the lines that contain the string "carrot"

grep "^314.*carrot$" file.txt | wc -l

Count the number of lines that start with "314" and end with "carrot" by the previous command rather than printing them to the screen.

find $HOME -name "*.pro" |xargs grep readcol |less

You've forgotten the syntax for the idl command "readcol", but you're sure you have a program file somewhere that uses readcol. If only you could find it! Starting in your home directory, this command marches through every subdirectory and identifies any files whose suffix is "pro." Then it searches those files for lines containing the string "readcol" and outputs those lines so you can inspect them with "less".

ssh susie@woof "cat /home/susie/file.tar.gz" | tar xvzf -

Extract the contents of the archive file.tar.gz, located in susie's home directory on the remote computer "woof", to the current directory without first scp-ing the file to the current computer.

echo "Hello world" > file.txt

Write the string "Hello world" to the file file.txt. If file.txt already exists, this will overwrite it, so if you'd prefer to append "Hello world" to the end of file.txt in the event that file.txt exists, use >>:

echo "Hello world" >> file.txt

awk '{print 3.0 * $1}' file.txt >file2.txt

Take the first column of file.txt, multiply by 3, and write the result to a new file called file2.txt (again, overwriting if it already exists).

awk '{if(NR>1) sum+=$1/$2;n++}END{print sum/n}' file.txt

Skip the first line of file.txt, then compute the average of the ratio of the numbers of column 1 to column 2, then print that ratio to the screen

awk '{if($1!=2 && $3>max) max=$3}END{print max}' file.txt

Find the maximum value of the third column in file.txt, excluding lines where the first column equals 2, and print that maximum to the screen