1. Commands that display information.

a. `cat` *filename*

Displays the content of file *filename*.

b. `file` *filename*

Descibes the kind of file that *filename* is.

c. `finger`

This shows who is currently logged-on.

d. `finger` *username*

Prints information about the user.

e. `ls`

Lists files and subdirectories of the current directory.

f. `ls -l`

The above, but long format, more information. The first column (actually 10 character positions) lists permissions. If there is a 'd' in the first place the item listed is a directory. The remaining 9 positions come in groups of 3. The first three deal with the owner's permissions. (Note: all files and directories are owned by someone. You own your home directory.) The next 3 deal with permissions of others in your "group" and the last 3 deal with with permissions of all other users. In each group of 3, the first position deals with reading (r), the second position deals with writing (w), and the third deals with executing (x). Files in your directory should have rw- in the first group indicating that you

have the right to read (look at the file, copy it),
have the right to write (change, destroy the file),
do not the right to execute the file (run the file as a unix command).

g. `ls` *dirname*

Lists files, subdirectories of directory *dirname*.

h. `man` *command*

Displays information about the command *command*. Press spacebar for next screenful. Type q to quit.

i. `more` *filename*

Displays content of file *filename* a screenful at a time. To see the next screenful, press the spacebar. To see one more line press the enter key. Quit by typing a q.

j. `ps`

Lists all processes still alive started during your current login. You will see a column headed with PID. PID stands for process identifier. Each process on the system is identified by an integer. You can use a process identifier to signal the corresponding process with the `kill` command.

k. `ps -aux`

See all the processes on the system.

l. `ps -x`

Lists all live processes belonging to you, even those started at an earlier login.

m. `pwd`

Displays current directory path.

m. `wc` *filename*

Displays the number of lines, words, and characters in a the file.

2. Current Directory, Pathnames. Try the `pwd` command. If you have just logged on you should see:

```
/u1/class/cs170dd
```

where the letter d's stand for the digits you see. The first slash stands for the root directory. The other slashes are just separators. A path name for a file or directory specifies the location of the file or directory in cs server's file system. Our path tells us that `cs170dd` is contained in directory `/u1/class` which is contained in directory `/u1` which is contained in the root directory, `/`. The root directory is not contained in any other directory. It is the top directory. All absolute paths start there. The path name of your file `pg1.html` is

```
/u1/class/cs170dd/public_html/pg1.html
```

All commands that take file names expect full (absolute) pathnames of files. If you leave off the first slash in a file or directory name it is assumed that it is in your current directory (relative pathname) If you use a command followed by a relative pathname, like:

```
cd public_html
```

unix puts the pathname of the current directory, say

```
/u1/class/cs170dd
```

in front of the relative pathname to make the absolute path name,

```
/u1/class/cs170dd/public_html
```

Note the slash (`/`) separator between the two parts.

When you logon, your current directory is the home directory that goes with your login. The short hands `~` and `~sternfl` stand respectively for the path to your directory and the path to my directory. For example the expression `~sternfl/examples` stands for the `examples` subdirectory in my home directory.

3. Commands that make changes.

a. `cd`

Makes the current directory be your home directory.

b. `cd` *directoryname*

Makes the current directory be *directoryname*. Examples: `cd ..` makes the new current directory be the parent (containing) directory of the current directory. `cd ~sternfl` makes your current directory be my home directory. `cd /` makes the root directory be your current directory.

c. `cp` *file1 file2*

Copies *file1* to *file2*. If *file2* already exists, its old contents are replaced by a copy *file1*.

d. `cp` *file1 directory*

Copies *file1* to a file of the same "last" name in directory *directory*. By "last" name I mean the right most segment of the pathname *file1* after the last slash. Example: `cp ~sternfl/E1.java ~` . The last name is `E1.java`. The command will copy the file to a file in your home directory with the name `E1.java`. If you already have a file by that name in your home directory its contents will be replaced by the contents of my file.

e. kill -9 *processIdNumber*

Sends a signal 9 to the process with process idNumber *processIdNumber*. This **kills** the process.

f. `mkdir` *pathname*

Makes a new directory. The new directory will be the "last" name (see above) in the pathname.

g. `rm` *file*

Removes the file.

h. `rmdir` *dir*

Removes the directory. The directory must be empty (contain no files or subdirectories).

i. `chmod ooo` *file*

j. `chmod ooo` *dir*

Changes the permissions. The `ooo` are three octal digits. The first `o` sets your permissions, the second your group's permissions, and the last everyone elses'. 4 = read permission; 2 = write permission, 1 = execute permission. 6 = 4+2 = read and write permission. 0 = no permission.