

Chapter 2:

Ten Things to Know About UNIX Commands

1. how commands work

All UNIX commands work in what is called a "verb-noun" syntax. This means that you first type in the verb, followed by the noun. For example, to move into a subdirectory called `datafiles`, type:

```
cd datafiles
```

The "cd" command is the verb (it means "change directory"), and "datafiles" is the noun.

(This is all in contrast to the "noun-verb" syntax of windows, where you select an object--such as a file on the desktop--and then use a verb--like a delete command--to perform an action.)

2. directories

Just like in Windows, UNIX keeps your files in directories (or folders). Just like in Windows, there can be directory within directories. To find out what directory you are currently in, type:

```
pwd
```

which stands for "present working directory". To change to a different directory, use the "cd" command. For example, to change to a directory called "data", type:

```
cd data
```

There are a couple of points about the `cd` command that should be clarified. Firstly, you can only `cd` over to directories that are "inside" the current directory. (The same is true in Windows, of course: in Windows Explorer, you can only open folders that are inside the current folder.) Secondly, you can only `cd` to directories that *exist*. (Again, the same is true in Windows—you can't click on folders that don't exist.)

To change directory upwards to the directory that *contains* the current directory, type:

```
cd ..
```

The symbol “..” (pronounced “dot dot”) is a short-hand for “the parent directory”, which is the directory that contains the current directory. (The same notation works on Windows and DOS computers, too, by the way.)

3. listing the files

To get a list of everything that is in a directory, type "ls".

```
ls
```

To get to see all of the "details" about the files, including their sizes, type "dir" or "ls -last".

```
dir
```

The dir command is an *alias* for “ls –last”, and it might not work on all computers that you come across. If not, just go ahead and type out the entire “ls –last” command.

4. wildcards

Use a "?" symbol as a wildcard, meaning that ANY CHARACTER can appear in this spot. For example, to list all of the files that have names 5 characters long, type:

```
ls ??????
```

To list all of the files that start with "test.da" but have one more letter, type:

```
ls test.da?
```

A more powerful wildcard is *. * matches EVERYTHING, including NOTHING. For example, to list all of the files in the directory, type:

```
ls *
```

To list all of the files that start with "test.", type:

```
ls test.*
```

5. copying files

To copy a file, use the "cp" command:

```
cp source target
```

You can combined cp with the wildcards. For example, to copy all files that start with "test." to a subdirectory called myfiles, type:

```
cp test.* myfiles
```

6. erasing files

Files are removed using the "rm" command. To remove a file called test.dat, type:

```
rm test.dat
```

The rm command can be combined with the wildcards to remove many files at once. For example, to remove all files starting with the word "test.", type:

```
rm test.*
```

To remove all files in the directory, type:

```
rm *
```

WARNING: There is no "undelete" in UNIX. What gets erased *stays* erased! Combining the rm command with wildcards can be extremely dangerous!

7. making your own subdirectories

A subdirectory is created inside the current directory using the "mkdir" command. To create a subdirectory called myfiles, type:

```
mkdir myfiles
```

8. removing subdirectories

A subdirectory is removed using the following command:

```
rmdir myfiles
```

In most versions of UNIX, the subdirectory has to be empty before you can remove it.

9. seeing the contents of a file

For most types of files that we will use in this class, the contents of the file can be *seen* (but not *changed*) using the "more" command. To see the contents of a file called test.dat type:

```
more test.dat
```

10. editing the contents of a file

Ah, well, that's what editors like vi are for. And we'll learn about that in the next chapter.