

Introduction to Unix

Part 1: Navigating directories

First we download the directory called "Fisher" from [Carmen](#).

This directory contains a sample from the Fisher corpus. The Fisher corpus is a collection of short phone conversations between people who typically do not know each other. Participants are assigned a random topic to talk about.

We will explore the data at the command line. To do so, we open a shell.

- Open the Terminal application
- Go to "Shell" > "New Window" > "New Window with Settings - Basic"

This gives you a shell in which we can type. Here is the prompt I see (yours will be different):

```
dhcp48:~ mcdm$
```

Let's explore some Unix commands.

ls

What does this command do? Try it. You need to type it and hit "return".

It lists the contents of the directory we are in. By default, the Unix login session begins in your home directory.

man

If you want more information about a command, you can look in the manual. For example, to get information about the "ls" command, just type

```
$ man ls

NAME
  ls -- list directory contents

SYNOPSIS
  ls [-ABCFGHLOPRSTUW@abcdefghiklmnopqrstuvwxyz1] [file ...]

DESCRIPTION
  For each operand that names a file of a type other than directory, ls displays its name as well as any requested, associated information. For each operand that names a file of type directory, ls displays the names of files contained within that directory, as well as any requested, associated information.

  If no operands are given, the contents of the current directory are displayed.
```

To quit the manual, type "q" (for quit).

cd

This command changes your current directory location. Using it, we can navigate the directory hierarchy. Our Fisher directory has been downloaded into the "Downloads" folder, so let's go there.

```
$ cd Downloads
$ ls
```

Remember, "ls" lists the directory content. You should see the Fisher directory there. What's in it?

```
$ cd Fisher
$ ls
```

What do we see?

We see that "Fisher" contains two directories, named "058" and "065".

Go to one of these directories.

```
$ cd 058
$ ls
```

We see a bunch of files in there. What's in the other directory? We can go there directly. We know that the directory hierarchy looks like this:

```
Downloads
  Fisher
    058
      fe_03_05851.txt
      fe_03_05852.txt
      etc.
    065
```

Right now we are **in** the directory called "058". To go to the other directory ("065"), we need to go back up to "Fisher" and then go into "065". Conveniently "cd .." moves to the parent of the current directory. So we can do

```
$ cd ..
$ cd 065
```

Or in one command:

```
$ cd ../065
```

What follows the command is the *path*. "pwd" will give you the whole path of the directory you are currently in.

Just typing "cd" makes you go back to your home directory.

Go into the "065" directory again (if you are not there anymore). How do we go back to the Downloads folder from there?

```
$ cd ../../
```

Part 2: Looking into a file

Now we want to see how the files are structured. Let's look into one.

more

```
$ more fe_03_06596.txt

0.59 1.92 A-f: hello
1.96 2.97 B-m: (( hello ))
2.95 3.98 A-f: hello
3.71 5.43 B-m: my name is kevin gonzales
5.49 7.30 A-f: hi this is carol
6.95 8.35 B-m: carol okay carol
8.42 9.44 A-f: [laughter]
9.56 11.27 A-f: well that was fast
12.48 13.79 B-m: uh hello
13.47 14.85 A-f: yeah i'm here
14.58 15.53 B-m: okay
16.81 25.87 A-f: um so do you think that public or private school
```

To see more of the file, hit the "return" key.

It seems that we have one utterance per line. Participants are coded as A and B, and we have gender information (as *-f* or *-m* tagged to the participant code). The numbers at the beginning of the lines are the start and end time of the utterance.

Ok, we have a pretty decent understanding of how the file is structured. How do we quit this?

```
"q" -- easy :-)
```

Let's look at the second file. People are lazy... Using the "tab" key, Unix will fill the rest of the command for you. So let's try this.

```
$ more f [Then hit "tab"]
```

Magical, isn't it?

Now we just type "01", and hit "tab" again, and we are seeing the content of the file "fe_03_06501.txt".

Another small trick: Using the up and down arrows, we can repeat commands we just typed (there is a *history* of the commands we typed).

less

We can also use the "less" command to look into the file.

```
$ less fe_03_06596.txt
```

What's the difference between "more" and "less"? How can I find the answer to this question?

Look at the manual!

```
$ man less
```

DESCRIPTION

```
Less is a program similar to more (1), but which allows backward move-
ment in the file as well as forward movement. Also, less does not have
to read the entire input file before starting, so with large input
files it starts up faster
```

WC

Can we roughly quantify the amount of data that's available here? E.g., how many files do we have? How many words? We can count stuff easily with the "wc" command.

I. What happens when we type the "wc" command?

```
$ wc fe_03_06501.txt
173      2729   14055 fe_03_06501.txt
```

Tell me what these numbers mean. How do you find the answer?

That's right: we look at the manual.

```
$ man wc

NAME
  wc -- word, line, character, and byte count

SYNOPSIS
  wc [-clmw] [file ...]

DESCRIPTION
  The wc utility displays the number of lines, words, and bytes contained in each input file, or standard input (if no file is specified) to the standard output. A line is defined as a string of characters delimited by a <newline> character. Characters beyond the final <newline> character will not be included in the line count.
```

So what are the different numbers we see?

- 173 is the number of lines in the file "fe_03_06501.txt"
- 2,729 is the number of words in the file "fe_03_06501.txt"
- 14,055 is the number of bytes in the file "fe_03_06501.txt"

Now I want to know how many words I have in this whole directory "065", and not only for this particular file. We will use * which serves as a wildcard character. So *.txt will mean "anything that ends with .txt"

How is a word defined?

```
$ wc *.txt
19178  252415 1324595 total
```

II. Can we know how many files we have in this "065" directory?

We can list the directory content and count the output.

We use the vertical bar "|" to connect two commands together so that the output from one command becomes the input of the next command. Two (or more) commands connected in this way form what's called a *pipe*.

```
$ ls | wc
100      100     1600
```

So we have 100 files in the "065" directory.

Now count the number of files in the "058" directory.

Redirection

What if we want to capture the output of a command in a more permanent way? We can use the command line to create a file with our results using the ">" character. This is just like the pipe, except that the output goes to a file instead of another command.

Try the command:

```
$ wc *.txt > lineCounts.txt
```

What is in the lineCounts.txt file now? (Which command do you use to view the file contents?)

Be very careful with output redirection... if the file after the ">" already exists, its contents are overwritten with no warning!

Exercise

How many files and how many words do we have in total in the Fisher directory?

Part 3: Going further

There are other useful Unix commands, and you might want to take a look at [Unix for Poets](#).

But for our purpose, we just want to be able to navigate the directory tree and have quick peeks into files. Sometimes a file might be really big, and you don't want to open it in Word ;-)

The "grep" command might be quite useful to you but we will not insist too much on this. Regular expressions are quite handy but tricky too (see [xkcd comic](#)). Just for the fun of it, I will quickly illustrate what you can do with regular expressions.

"grep" stands for *Globally search a Regular Expression and Print*. It searches plain-text data for **lines** matching a regular expression. The syntax is the following:

```
grep regular_expression filename
```

We can first try regular expressions in an interactive mode. We will type "grep" followed by the regular expression we want to look for, then "enter". We can then type some text (one line), and hit enter. If the regular expression matches the text, the line will be echoed back to us. If not, nothing will happen.

```
grep "hello"
hello
hello    #echoed back to us

hello dave
hello dave #echoed back to us

dave
#nothing is happening
```

Use ctrl-C to quit.

It can be useful to use some colors, to see what exactly gets matched. Do to this, we add the option "--color=auto":

```
grep --color=auto "hello"
```

Using "grep", we can have an idea of how much people laugh in these conversations. How would we do this?

Remember the first file we looked at:

```
$ cd
$ cd Downloads/Fisher/065/
$ more fe_03_06596.txt

0.59 1.92 A-f: hello
1.96 2.97 B-m: (( hello ))
2.95 3.98 A-f: hello
3.71 5.43 B-m: my name is kevin gonzales
5.49 7.30 A-f: hi this is carol
6.95 8.35 B-m: carol okay carol
8.42 9.44 A-f: [laughter]
9.56 11.27 A-f: well that was fast
12.48 13.79 B-m: uh hello
13.47 14.85 A-f: yeah i'm here
14.58 15.53 B-m: okay
16.81 25.87 A-f: um so do you think that public or private school
```

Laughter is coded explicitly in the transcriptions as [laughter]. Using "wc", we know how many lines a file contains. If we can count how often [laughter] appears in these lines, then we will have an idea of the proportion of utterances containing laughter.

Let's make sure we are working on the same directory here. Let's start with 065.

```
$ grep "\[laughter\]" *.txt
```

We need to use the backslash character to *escape* the bracket character which has a special meaning in regular expressions. But here we want its literal meaning as a character. (If we type [xyz] without escaping the brackets, it will match any line that contains the letter x, y or z.)

Now how do we get the line count?

```
$ grep "\[laughter\]" *.txt | wc

1482    23609   161959
```

How many total lines do we have in the "065" directory?

```
$ wc *.txt

19178   252415  1324595 total
```

So roughly 8% of the utterances in the files from the "065" directory contain laughter.

Perhaps we want to do this on all the files in all the directories.

```
$ cd ../
$ grep -R "\[laughter\]" ./ | wc

1681    26608   186119
```

-R stands for *recursive*

./ refers to the current directory

Getting the number of lines of all the files in all the directories is a bit trickier, and I will not enter into the details, but here is a command to do it.

```
$ find . -name "*.txt" | xargs wc
```

```
29734 362020 1902592 total
```