Navigating Files and Directories

Overview

Teaching: 30 min
Exercises: 20 min
Questions

- How can I perform operations on files outside of my working directory?
- What are some navigational shortcuts I can use to make my work more efficient?

Objectives

- Use a single command to navigate multiple steps in your directory structure, including moving backwards (one level up).
- Perform operations on files in directories outside your working directory.
- Work with hidden directories and hidden files.
- Interconvert between absolute and relative paths.
- Employ navigational shortcuts to move around your file system.

Moving around the file system

We've learned how to use `pwd` to find our current location within our file system. We've also learned how to use `cd` to change locations and `ls` to list the contents of a directory. Now we're going to learn some additional commands for moving around within our file system.

Use the commands we've learned so far to navigate to the `dc_sample_data/untrimmed_fastq` directory, if you're not already there.

```
$ cd /pool/genomics/username
$ cd dc_sample_data
$ cd untrimmed_fastq
```

What if we want to move back up and out of this directory and to our top level directory? Can we type `cd dc_sample_data` ? Try it and see what happens.

```
$ cd dc_sample_data
```

```
-bash: cd: dc_sample_data: No such file or directory
```

Your computer looked for a directory or file called `dc_sample_data` within the directory you were already in. It didn't know you wanted to look at a directory level above the one you were located in.

We have a special command to tell the computer to move us back or up one directory level.

```
$ cd ..
```

Now we can use `pwd` to make sure that we are in the directory we intended to navigate to, and `ls` to check that the contents of the directory are correct.

```
$ pwd
```

```
/pool/genomics/username/dc_sample_data
```

```
$ ls
```

```
sra_metadata  untrimmed_fastq
```

From this output, we can see that `..` did indeed take us back one level in our file system.

You can chain these together like so:

```
$ ls ../../
```

prints the contents of `/pool/genomics`.

Finding hidden directories

First navigate to the `dc_sample_data` directory. There is a hidden directory within this directory. Explore the options for `ls` to find out how to see hidden directories. List the contents of the directory and identify the name of the text file in that directory.

Hint: hidden files and folders in Unix start with `.`, for example .my_hidden_directory

Solution

First use the `man` command to look at the options for `ls`.

```
$ man ls
```

The `-a` option is short for `all` and says that it causes `ls` to "not ignore entries starting with ." This is the option we want.

```
$ ls -a
```

```
.  ..  .hidden  sra_metadata  untrimmed_fastq
```

The name of the hidden directory is `.hidden`. We can navigate to that directory using `cd`.

```
$ cd .hidden
```

And then list the contents of the directory using `ls`.

```
$ ls
```

```
youfoundit.txt
```

The name of the text file is `youfoundit.txt`.

Examining the contents of other directories

By default, the `ls` commands lists the contents of the working directory (i.e. the directory you are in). You can always find the directory you are in using the `pwd` command. However, you can also give `ls` the names of other directories to view. Navigate to your `/pool/genomics` directory if you are not already there.

```
$ cd /pool/genomics/username
```

Then enter the command:

```
$ ls dc_sample_data
```

```
sra_metadata  untrimmed_fastq
```

This will list the contents of the `dc_sample_data` directory without you needing to navigate there.

The `cd` command works in a similar way.

Try entering:

```
$ cd /pool/genomics/username
$ cd dc_sample_data/untrimmed_fastq
```

This will take you to the `untrimmed_fastq` directory without having to go through the intermediate directory.

Navigating practice

Navigate to your `/pool/genomics` directory. From there, list the contents of the `untrimmed_fastq` directory.

Solution

```
$ cd /pool/genomics/username
$ ls dc_sample_data/untrimmed_fastq/
```

```
SRR097977.fastq  SRR098026.fastq
```

Full vs. Relative Paths

The `cd` command takes an argument which is a directory name. Directories can be specified using either a **relative** path or a full **absolute** path. The directories on the computer are arranged into a hierarchy. The full path tells you where a directory is in that hierarchy. Navigate to the home directory, then enter the `pwd` command.

```
$ cd
$ pwd
```

You will see:

```
/home/username
```

This is the full name of your home directory. This tells you that you are in a directory called `username`, which sits inside a directory called `home` which sits inside the very top directory in the hierarchy. The very top of the hierarchy is a directory called `/` which is usually referred to as the **root** directory. So, to summarize: `username` is a directory in `home` which is a directory in `/`.

Now enter the following command:

```
$ cd /pool/genomics/username/dc_sample_data/.hidden
```

This jumps forward multiple levels to the `.hidden` directory. Now go back to the `/pool/genomics` directory.

```
$ cd /pool/genomics
```

You can also navigate to the `.hidden` directory using:

```
$ cd dc_sample_data/.hidden
```

These two commands have the same effect, they both take us to the `.hidden` directory. The first uses the absolute path, giving the full address from the home directory. The second uses a relative path, giving only the address from the working directory. A full path always starts with a `/`. A relative path does not.

A relative path is like getting directions from someone on the street. They tell you to “go right at the stop sign, and then turn left on Main Street”. That works great if you’re standing there together, but not so well if you’re trying to tell someone how to get there from another country. A full path is like GPS coordinates. It tells you exactly where something is no matter where you are right now.

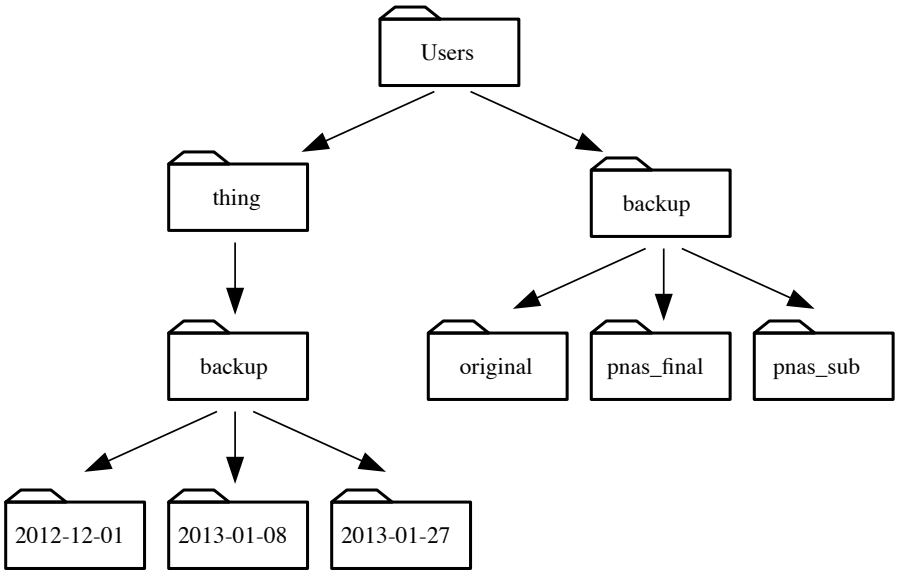
You can usually use either a full path or a relative path depending on what is most convenient. If we are in the home directory, it is more convenient to enter the relative path since it involves less typing.

Over time, it will become easier for you to keep a mental note of the structure of the directories that you are using and how to quickly navigate amongst them.

Relative path resolution

Using the filesystem diagram below, if `pwd` displays `/Users/thing`, what will `ls ../backup` display?

1. `../backup: No such file or directory`
2. `2012-12-01 2013-01-08 2013-01-27`
3. `2012-12-01/ 2013-01-08/ 2013-01-27/`
4. `original pnas_final pnas_sub`



Solution

1. No: there was a directory `backup` in `/Users`.
2. No: this is the content of `Users/thing/backup`, but with `..` we asked for one level further up.
3. No: see previous explanation. Also, we did not specify `-F` to display `/` at the end of the directory names.
4. Yes: `../backup` refers to `/Users/backup`.

Navigational Shortcuts

There are some shortcuts which you should know about. Dealing with the home directory is very common. The tilde character, `~`, is a shortcut for your home directory.

```
$ ls ~
```

```
bio
```

This prints the contents of your home directory, without you needing to type the full path.

There is a shortcut to quickly return to your home directory. typing `cd` with no directory name will bring you back to your home (~):

```
$ cd
$ ls
```

```
bio
```

The commands `cd` , and `cd ~` are very useful for quickly navigating back to your home directory. We will be using the `~` character in later lessons to specify our home directory.

Key Points

- The `/` , `~` , and `..` characters represent important navigational shortcuts.
- Hidden files and directories start with `.` and can be viewed using `ls -a` .
- Relative paths specify a location starting from the current location, while absolute paths specify a location from the root of the file system.

‹
(../01-introduction/index.html)

›
(../03-working-with-files/i