Introducing the Shell

Overview

Teaching: 20 min
Exercises: 10 min
Questions

- What is a command shell and why would I use one?
- How can I move around on my computer?
- How can I see what files and directories I have?
- How can I specify the location of a file or directory on my computer?

Objectives

- Describe key reasons for learning shell.
- Navigate your file system using the command line.
- Access and read help files for `bash` programs and use help files to identify useful command options.
- Demonstrate the use of tab completion, and explain its advantages.

What is a shell and why should I care?

A *shell* is a computer program that presents a command line interface which allows you to control your computer using commands entered with a keyboard instead of controlling graphical user interfaces (GUIs) with a mouse/keyboard combination.

There are many reasons to learn about the shell:

- Many bioinformatics tools can only be used through a command line interface, or have extra capabilities in the command line version that are not available in the GUI. This is true, for example, of BLAST, which offers many advanced functions only accessible to users who know how to use a shell.
- The shell makes your work less boring. In bioinformatics you often need to do the same set of tasks with a large number of files. Learning the shell will allow you to automate those repetitive tasks and leave you free to do more exciting things.
- The shell makes your work less error-prone. When humans do the same thing a hundred different times (or even ten times), they're likely to make a mistake. Your computer can do the same thing a thousand times with no mistakes.
- The shell makes your work more reproducible. When you carry out your work in the command-line (rather than a GUI), your computer keeps a record of every step that you've carried out, which you can use to re-do your work when you need to. It also gives you a way to communicate unambiguously what you've done, so that others can check your work or apply your process to new data.
- Many bioinformatic tasks require large amounts of computing power and can't realistically be run on your own machine. These tasks are best performed using remote computers or cloud computing, which can only be accessed through a shell.

In this lesson you will learn how to use the command line interface to move around in your file system.

How to access the shell

On a Mac or Linux machine, you can access a shell through a program called Terminal, which is already available on your computer. If you’re using Windows, you’ll need to download a separate program to access the shell.

We will spend most of our time learning about the basics of the shell by manipulating some experimental data. Some of the data we’re going to be working with is quite large, and we’re also going to be using several bioinformatic packages in later lessons to work with this data. To avoid having to spend time downloading the data and downloading and installing all of the software, we’re going to be working with data on a remote server.

You can log-in to the remote server using the instructions here (http://www.datacarpentry.org/cloud-genomics/02-logging-onto-cloud/#logging-onto-a-cloud-instance). Your instructor will supply the `ip_address` and password that you need to login.

Each of you will have a different `ip_address`. This will prevent us from accidentally changing each other’s files as we work through the exercises. The password will be the same for everyone.

After logging in, you will see a screen showing something like this:

Output

```
Welcome to Ubuntu 14.04.3 LTS (GNU/Linux 3.13.0-48-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

  System information as of Sat Feb  2 00:08:17 UTC 2019

  System load: 0.0                Memory usage: 5%   Processes:       82
  Usage of /:  29.9% of 98.30GB   Swap usage:   0%   Users logged in: 0

  Graph this data and manage this system at:
    https://landscape.canonical.com/

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

597 packages can be updated.
444 updates are security updates.

New release '16.04.5 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


Last login: Fri Feb  1 22:34:53 2019 from c-73-116-43-163.hsd1.ca.comcast.net
```

This provides a lot of information about the remote server that you’re logging in to. We’re not going to use most of this information for our workshop, so you can clear your screen using the `clear` command.

Bash

```
$ clear
```

This will scroll your screen down to give you a fresh screen and will make it easier to read. You haven’t lost any of the information on your screen. If you scroll up, you can see everything that has been output to your screen up until this point.

Navigating your file system

The part of the operating system responsible for managing files and directories is called the **file system**. It organizes our data into files, which hold information, and directories (also called “folders”), which hold files or other directories.

Several commands are frequently used to create, inspect, rename, and delete files and directories.

Preparation Magic

If you type the command: `PS1='$ '` into your shell, followed by pressing the Enter key, your window should look like our example in this lesson.
This isn't necessary to follow along (in fact, your prompt may have other helpful information you want to know about). This is up to you!

Bash

```
$
```

The dollar sign is a **prompt**, which shows us that the shell is waiting for input; your shell may use a different character as a prompt and may add information before the prompt. When typing commands, either from these lessons or from other sources, do not type the prompt, only the commands that follow it.

Let's find out where we are by running a command called `pwd` (which stands for "print working directory"). At any moment, our **current working directory** is our current default directory, i.e., the directory that the computer assumes we want to run commands in, unless we explicitly specify something else. Here, the computer's response is `/home/dcuser`, which is the top level directory within our cloud system:

Bash

```
$ pwd
```

Output

```
/home/dcuser
```

Let's look at how our file system is organized. We can see what files and subdirectories are in this directory by running `ls`, which stands for "listing":

Bash

```
$ ls
```

Output

```
R  r_data  shell_data
```

`ls` prints the names of the files and directories in the current directory in alphabetical order, arranged neatly into columns. We'll be working within the `shell_data` subdirectory, and creating new subdirectories, throughout this workshop.

The command to change locations in our file system is `cd`, followed by a directory name to change our working directory. `cd` stands for "change directory".

Let's say we want to navigate to the `shell_data` directory we saw above. We can use the following command to get there:

Bash

```
$ cd shell_data
```

Let's look at what is in this directory:

Bash

```
$ ls
```

Output

```
sra_metadata  untrimmed_fastq
```

We can make the `ls` output more comprehensible by using the **flag** `-F`, which tells `ls` to add a trailing `/` to the names of directories:

Bash

```
$ ls -F
```

Output

```
sra_metadata/  untrimmed_fastq/
```

Anything with a “/” after it is a directory. Things with a “*” after them are programs. If there are no decorations, it’s a file.

`ls` has lots of other options. To find out what they are, we can type:

Bash

```
$ man ls
```

Some manual files are very long. You can scroll through the file using your keyboard’s down arrow or use the `Space` key to go forward one page and the `b` key to go backwards one page. When you are done reading, hit `q` to quit.

Challenge

Use the `-l` option for the `ls` command to display more information for each item in the directory. What is one piece of additional information this long format gives you that you don’t see with the bare `ls` command?

Solution

Bash

```
$ ls -l
```

Output

```
total 8
drwxr-x--- 2 dcuser dcuser 4096 Jul 30  2015 sra_metadata
drwxr-xr-x 2 dcuser dcuser 4096 Nov 15  2017 untrimmed_fastq
```

The additional information given includes the name of the owner of the file, when the file was last modified, and whether the current user has permission to read and write to the file.

No one can possibly learn all of these arguments, that’s what the manual page is for. You can (and should) refer to the manual page or other help files as needed.

Let’s go into the `untrimmed_fastq` directory and see what is in there.

Bash

```
$ cd untrimmed_fastq
$ ls -F
```

Output

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

This directory contains two files with `.fastq` extensions. FASTQ is a format for storing information about sequencing reads and their quality. We will be learning more about FASTQ files in a later lesson.

Shortcut: Tab Completion

Typing out file or directory names can waste a lot of time and it's easy to make typing mistakes. Instead we can use tab complete as a shortcut. When you start typing out the name of a directory or file, then hit the Tab key, the shell will try to fill in the rest of the directory or file name.

Return to your home directory:

Bash

```
$ cd
```

then enter:

Bash

```
$ cd she<tab>
```

The shell will fill in the rest of the directory name for `shell_data`.

Now change directories to `untrimmed_fastq` in `shell_data`

Bash

```
$ cd shell_data
$ cd untrimmed_fastq
```

Using tab complete can be very helpful. However, it will only autocomplete a file or directory name if you've typed enough characters to provide a unique identifier for the file or directory you are trying to access.

If we navigate back to our `untrimmed_fastq` directory and try to access one of our sample files:

Bash

```
$ cd
$ cd shell_data
$ cd untrimmed_fastq
$ ls SR<tab>
```

The shell auto-completes your command to `SRR09`, because all file names in the directory begin with this prefix. When you hit Tab again, the shell will list the possible choices.

Bash

```
$ ls SRR09<tab><tab>
```

Output

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

Tab completion can also fill in the names of programs, which can be useful if you remember the beginning of a program name.

Bash

```
$ pw<tab><tab>
```

Output

```
pwck       pwconv    pwd       pwdx       pwunconv
```

Displays the name of every program that starts with `pw`.

Summary

We now know how to move around our file system using the command line. This gives us an advantage over interacting with the file system through a GUI as it allows us to work on a remote server, carry out the same set of operations on a large number of files quickly, and opens up many opportunities for using bioinformatic software that is only available in command line versions.

In the next few episodes, we'll be expanding on these skills and seeing how using the command line shell enables us to make our workflow more efficient and reproducible.

Key Points

- The shell gives you the ability to work more efficiently by using keyboard commands rather than a GUI.
- Useful commands for navigating your file system include: `ls`, `pwd`, and `cd`.
- Most commands take options (flags) which begin with a `-`.
- Tab completion can reduce errors from mistyping and make work more efficient in the shell.

(../)

(../02-the-filesys

Edit on GitHub (https://github.com/datacarpentry/shell-genomics/edit/gh-pages/_episodes/01-introduction.md) / Contributing (https://github.com/datacarpentry/shell-genomics/blob/gh-pages/CONTRIBUTING.md) / Source (https://github.com/datacarpentry/shell-genomics/) / Cite (https://github.com/datacarpentry/shell-genomics/blob/gh-pages/CITATION) / Contact (mailto:)

Using The Carpentries theme (https://github.com/carpentries/carpentries-theme/) — Site last built on: 2019-11-23 19:34:47 +0000.