Data Engineer Technical Documentation

Table of Contents

[Introduction to Shell 2](#_Toc171603924)

[Manipulating files and directories 2](#_Toc171603925)

[How does the shell compare to a desktop interface? 2](#_Toc171603926)

[Where am I? 3](#_Toc171603927)

[How can I identify files and directories? 4](#_Toc171603928)

[How else can I identify files and directories? 5](#_Toc171603929)

[How can I move to another directory? 6](#_Toc171603930)

[How can I move up a directory? 7](#_Toc171603931)

[How can I copy files? 8](#_Toc171603932)

[How can I rename files? 10](#_Toc171603933)

[How can I delete files? 11](#_Toc171603934)

[How can I create and delete directories? 12](#_Toc171603935)

[Wrapping up 13](#_Toc171603936)

[Manipulating data 15](#_Toc171603937)

[Combining tools 15](#_Toc171603938)

[Batch processing 15](#_Toc171603939)

[Creating new tools 15](#_Toc171603940)

# Introduction to Shell

Description

The Unix command line has survived and thrived for almost 50 years because it lets people do complex things with just a few keystrokes. Sometimes called "the universal glue of programming," it helps users combine existing programs in new ways, automate repetitive tasks, and run programs on clusters and clouds that may be halfway around the world. This Introduction will introduce the key elements and show how to use them efficiently.

## Manipulating files and directories

This is a brief introduction to the Unix shell. You'll learn why it is still in use after almost 50 years, how it compares to the graphical tools you may be more familiar with, how to move around in the shell, and how to create, modify, and delete files and folders.

### How does the shell compare to a desktop interface?

An operating system like Windows, Linux, or Mac OS is a special kind of program. It controls the computer's processor, hard drive, and network connection, but its most important job is to run other programs.

Since human beings aren't digital, they need an interface to interact with the operating system. The most common one these days is a graphical file explorer, which translates clicks and double-clicks into commands to open files and run programs. Before computers had graphical displays, though, people typed instructions into a program called a command-line shell. Each time a command is entered, the shell runs some other programs, prints their output in human-readable form, and then displays a prompt to signal that it's ready to accept the next command. (Its name comes from the notion that it's the "outer shell" of the computer.)

Typing commands instead of clicking and dragging may seem clumsy at first, but as you will see, once you start spelling out what you want the computer to do, you can combine old commands to create new ones and automate repetitive operations with just a few keystrokes.

|  |
| --- |
| **Question:** What is the relationship between the graphical file explorer that most people use and the command-line shell?  **Answer:** They are both interfaces for issuing commands to the operating system.  **Note:** Both take the user's commands (whether typed or clicked) and send them to the operating system.  **Wrong answer:**   * The shell is part of the operating system, while the file explorer is separate. * The file explorer is built on top of the shell. * The file explorer lets you view and edit files, while the shell lets you run programs. |

### Where am I?

The **filesystem** manages files and directories (or folders). Each is identified by an **absolute path** that shows how to reach it from the filesystem's **root directory**: /home/repl is the directory repl in the directory home, while /home/repl/course.txt is a file course.txt in that directory, and / on its own is the root directory.

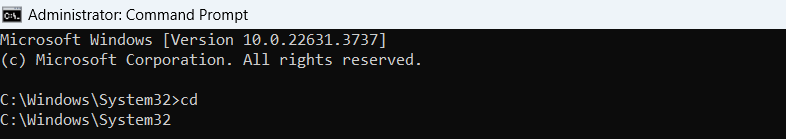
To find out where you are in the filesystem, run the command pwd (short for "**p**rint **w**orking **d**irectory"). This prints the absolute path of your **current working directory**, which is where the shell runs commands and looks for files by default.

Run pwd. Where are you right now?

**Note:** pwd is a linux command, won't work in windows. In windows just run cd without passing any arguments

**Linux:**

**Windows:**



### How can I identify files and directories?

pwd tells you where you are. To find out what's there, type ls (which is short for "listing") and press the enter key. On its own, ls lists the contents of your current directory (the one displayed by pwd). If you add the names of some files, ls will list them, and if you add the names of directories, it will list their contents. For example, ls /home/repl shows you what's in your starting directory (usually called your **home directory**).

**Question:** Use ls with an appropriate argument to list the files in the directory /home/repl/seasonal (which holds information on dental surgeries by date, broken down by season). Which of these files is not in that directory?

**Possible answers:**

autumn.csv

fall.csv - that file is not in the seasonal directory.

spring.csv

winter.csv

**Note:** Use the command dir to list all the directories and files in Windows; ls is a unix command.

### How else can I identify files and directories?

An absolute path is like a latitude and longitude: it has the same value no matter where you are. A **relative path**, on the other hand, specifies a location starting from where you are: it's like saying "20 kilometers north".

As examples:

* If you are in the directory /home/repl, the **relative** path seasonal specifies the same directory as the **absolute** path /home/repl/seasonal.
* If you are in the directory /home/repl/seasonal, the **relative** path winter.csv specifies the same file as the **absolute** path /home/repl/seasonal/winter.csv.

The shell decides if a path is absolute or relative by looking at its first character: If it begins with /, it is absolute. If it *does not* begin with /, it is relative.

**Question:** You are in /home/repl. Use ls with a **relative path** to list the file that has an absolute path of /home/repl/course.txt (and only that file).

**Answer:**



**Question:** You are in /home/repl. Use ls with a **relative** path to list the file /home/repl/seasonal/summer.csv (and only that file).

**Answer:**



**Question:** You are in /home/repl. Use ls with a **relative** path to list the contents of the directory /home/repl/people.

**Answer:**



### How can I move to another directory?

Just as you can move around in a file browser by double-clicking on folders, you can move around in the filesystem using the command cd (which stands for "change directory").

If you type cd seasonal and then type pwd, the shell will tell you that you are now in /home/repl/seasonal. If you then run ls on its own, it shows you the contents of /home/repl/seasonal, because that's where you are. If you want to get back to your home directory /home/repl, you can use the command cd /home/repl.

**Question:** You are in /home/repl/. Change directory to /home/repl/seasonal using a relative path.

**Answer:**

**Question:** Use pwd to check that you're there.

**Answer:**

**Question:** Use ls without any paths to see what's in that directory.

**Answer:** ****

### How can I move up a directory?

The **parent** of a directory is the directory above it. For example, /home is the parent of /home/repl, and /home/repl is the parent of /home/repl/seasonal. You can always give the absolute path of your parent directory to commands like cd and ls. More often, though, you will take advantage of the fact that the special path .. (two dots with no spaces) means "the directory above the one I'm currently in". If you are in /home/repl/seasonal, then cd .. moves you up to /home/repl. If you use cd .. once again, it puts you in /home. One more cd .. puts you in the root directory /, which is the very top of the filesystem. (Remember to put a space between cd and .. - it is a command and a path, not a single four-letter command.)

A single dot on its own, ., always means "the current directory", so ls on its own and ls . do the same thing, while cd . has no effect (because it moves you into the directory you're currently in).

One final special path is ~ (the tilde character), which means "your home directory", such as /home/repl. No matter where you are, ls ~ will always list the contents of your home directory, and cd ~ will always take you home.

**Question:** If you are in /home/repl/seasonal, where does cd ~/../. take you?

**Answer:** /home

**Note:** The path means 'home directory', 'up a level', 'here'.

**Wrong answer:**

* /home/repl
* /home/repl/seasonal
* / (the root directory)

### How can I copy files?

You will often want to copy files, move them into other directories to organize them, or rename them. One command to do this is cp, which is short for "copy". If original.txt is an existing file, then:

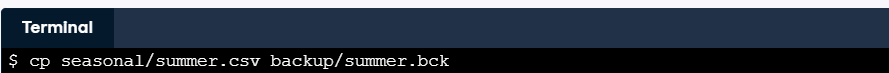
cp original.txt duplicate.txt

creates a copy of original.txt called duplicate.txt. If there already was a file called duplicate.txt, it is overwritten. If the last parameter to cp is an existing directory, then a command like:

cp seasonal/autumn.csv seasonal/winter.csv backup

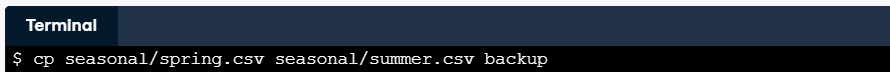
copies all of the files into that directory.

**Question:** Make a copy of seasonal/summer.csv in the backup directory (which is also in /home/repl), calling the new file summer.bck.

**Answer:**

**Question:** Copy spring.csv and summer.csv from the seasonal directory into the backup directory without changing your current working directory (/home/repl).

**Answer:**



**Note:** **By default**, there's no cp in Windows Command Prompt (cmd.exe). The equivalent cmd.exe command is copy. cp is a Unix command.

**Example:** copy D:\test1\test1.txt test2

How can I move a file?

While cp copies a file, mv moves it from one directory to another, just as if you had dragged it in a graphical file browser. It handles its parameters the same way as cp, so the command:

mv autumn.csv winter.csv ..

moves the files autumn.csv and winter.csv from the current working directory up one level to its parent directory (because .. always refers to the directory above your current location).

**Question:** You are in /home/repl, which has sub-directories seasonal and backup. Using a single command, move spring.csv and summer.csv from seasonal to backup.

**Answer:**



**Note:** **By default**, there's no mv in Windows Command Prompt (cmd.exe). The equivalent cmd.exe command is move. mv is a Unix command.

**Example:** move D:\test1\test1.txt test2

### How can I rename files?

mv can also be used to rename files. If you run:

mv course.txt old-course.txt

then the file course.txt in the current working directory is "moved" to the file old-course.txt. This is different from the way file browsers work, but is often handy.

One warning: just like cp, mv will overwrite existing files. If, for example, you already have a file called old-course.txt, then the command shown above will replace it with whatever is in course.txt.

**Question:** Go into the seasonal directory.

**Answer:**



**Question:** Rename the file winter.csv to be winter.csv.bck.

**Answer:**



**Question:** Run ls to check that everything has worked.

**Answer:**



**Example** **In Windows OS (operating system)**: move test1.txt test1.txt.bck

### How can I delete files?

We can copy files and move them around; to delete them, we use rm, which stands for "remove". As with cp and mv, you can give rm the names of as many files as you'd like, so:

rm thesis.txt backup/thesis-2017-08.txt

removes both thesis.txt and backup/thesis-2017-08.txt

rm does exactly what its name says, and it does it right away: unlike graphical file browsers, the shell doesn't have a trash can, so when you type the command above, your thesis is gone for good.

**Question:** You are in /home/repl. Go into the seasonal directory.

**Answer:** ****

**Question:** Remove autumn.csv.

**Answer:** 

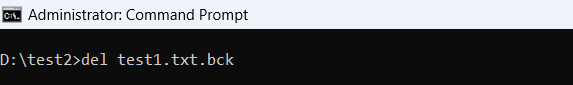
**Question:** Go back to your home directory.

**Answer:** ****

**Question:** Remove seasonal/summer.csv without changing directories again.

**Answer:** ****

**Note:** InWindows use del **Example:** del test1.txt.bck



### How can I create and delete directories?

mv treats directories the same way it treats files: if you are in your home directory and run mv seasonal by-season, for example, mv changes the name of the seasonal directory to by-season. However, rm works differently.

If you try to rm a directory, the shell prints an error message telling you it can't do that, primarily to stop you from accidentally deleting an entire directory full of work. Instead, you can use a separate command called rmdir. For added safety, it only works when the directory is empty, so you must delete the files in a directory before you delete the directory. (Experienced users can use the -r option to rm to get the same effect;)

**Question:** Without changing directories, delete the file agarwal.txt in the people directory.

**Answer:** ****

**Question:** Now that the people directory is empty, use a single command to delete it.

**Answer:** ****

**Note In Windows is the same:** rmdir test2



**Question:** Since a directory is not a file, you must use the command mkdir directory\_name to create a new (empty) directory. Use this command to create a new directory called yearly below your home directory.

**Answer:**



**Note In Windows is the same:** mkdir test2

**Question:** Now that yearly exists, create another directory called 2017 inside it without leaving your home directory.

**Answer:** ****

### Wrapping up

You will often create intermediate files when analyzing data. Rather than storing them in your home directory, you can put them in /tmp, which is where people and programs often keep files they only need briefly. (Note that /tmp is immediately below the root directory /, not below your home directory.) This wrap-up exercise will show you how to do that.

**Question:** Use cd to go into /tmp.

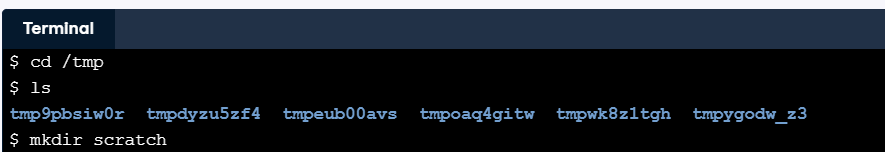
**Answer:** ****

**Question:** List the contents of /tmp without typing a directory name.

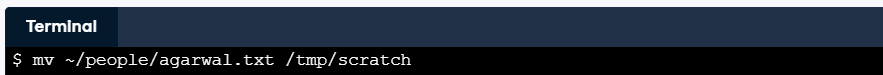
**Answer:** ****

**Question:** Make a new directory inside /tmp called scratch.

**Answer:**

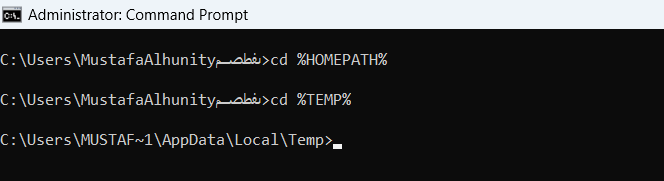


**Question:** Move /home/repl/people/agarwal.txt into /tmp/scratch. We suggest you use the ~ shortcut for your home directory and a relative path for the second rather than the absolute path.

**Answer:** ****

**Note In Windows OS is:** cd %TEMP%

**Note home directory in Windows OS is:** cd %HOMEPATH%



## Manipulating data

The commands you saw in the Manipulating files and directories allowed you to move things around in the filesystem. Here will show you how to work with the data in those files. The tools we’ll use are fairly simple, but are solid building blocks.

### How can I view a file's contents?

Before you rename or delete files, you may want to have a look at their contents. The simplest way to do this is with cat, which just prints the contents of files onto the screen. (Its name is short for "concatenate", meaning "to link things together", since it will print all the files whose names you give it, one after the other.)

cat agarwal.txt

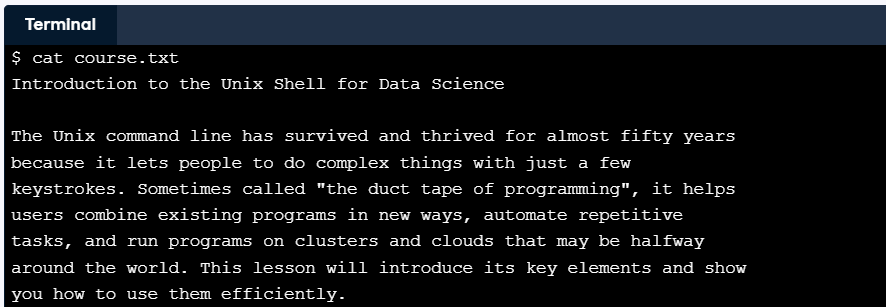
name: Agarwal, Jasmine

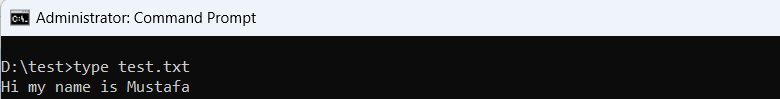
position: RCT2

start: 2017-04-01

benefits: full

**Question:** Print the contents of course.txt to the screen.

**Answer:** ****

**the Windows equivalent of the Unix command cat is type:**

### How can I view a file's contents piece by piece?

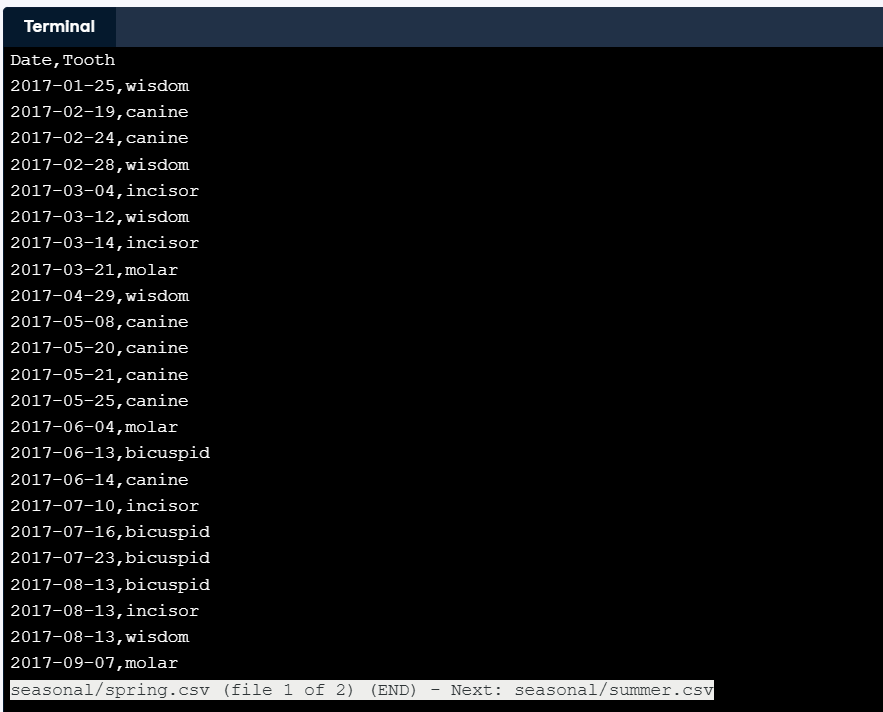
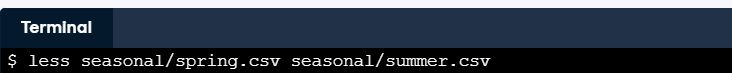
You can use cat to print large files and then scroll through the output, but it is usually more convenient to **page** the output. The original command for doing this was called more, but it has been superseded by a more powerful command called less. (This kind of naming is what passes for humor in the Unix world.) When you less a file, one page is displayed at a time; you can press spacebar to page down or type q to quit.

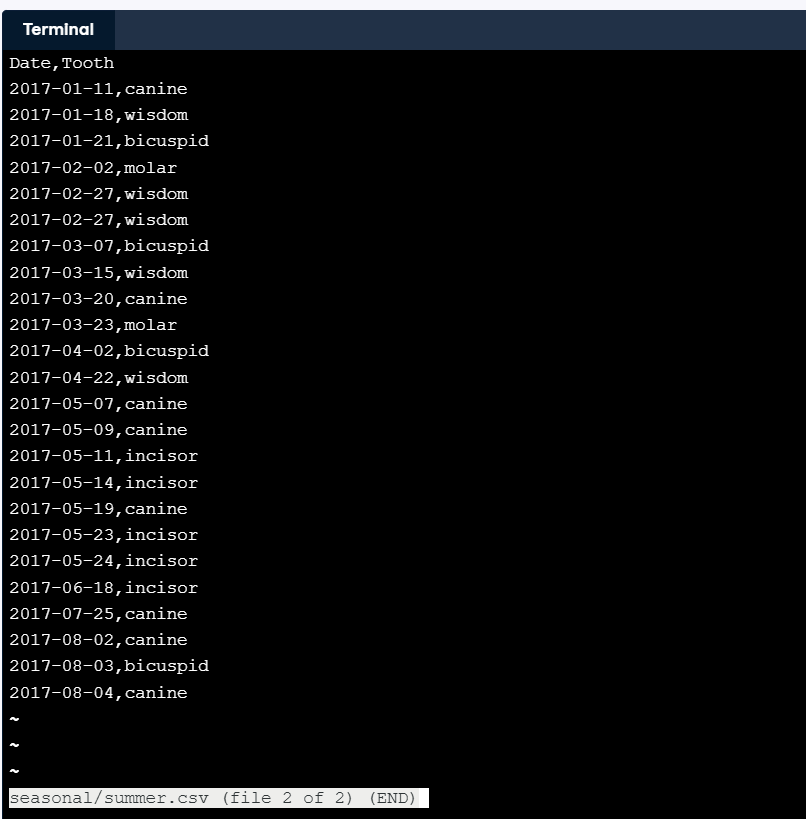
If you give less the names of several files, you can type :n (colon and a lower-case 'n') to move to the next file, :p to go back to the previous one, or :q to quit.

Note: If you view solutions to exercises that use less, you will see an extra command at the end that turns paging off so that we can test your solutions efficiently.

**Question:** Use less seasonal/spring.csv seasonal/summer.csv to view those two files in that order. Press spacebar to page down, :n to go to the second file, and :q to quit.

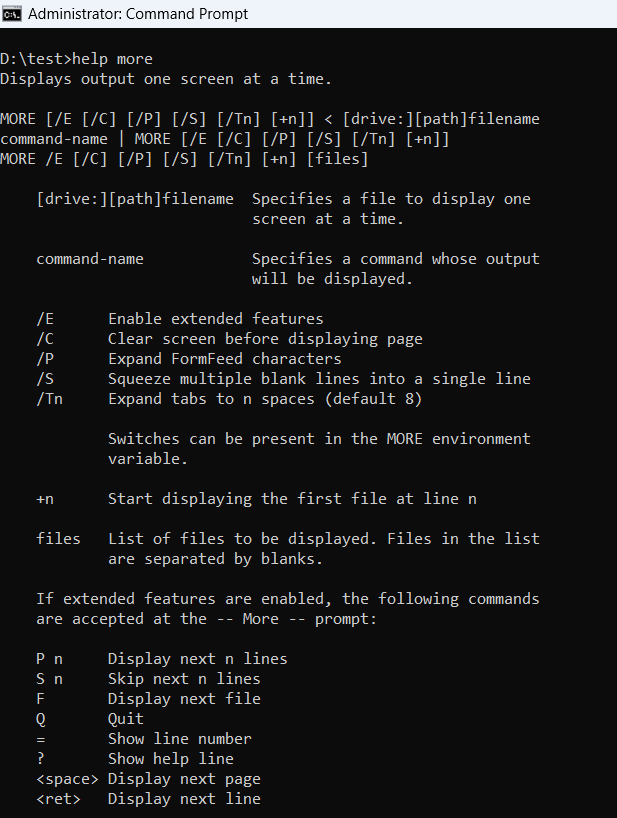
**Answer:**





**the Windows equivalent:**

**more** command



### How can I look at the start of a file?

The first thing most data scientists do when given a new dataset to analyze is figure out what fields it contains and what values those fields have. If the dataset has been exported from a database or spreadsheet, it will often be stored as **comma-separated values** (CSV). A quick way to figure out what it contains is to look at the first few rows.

We can do this in the shell using a command called head. As its name suggests, it prints the first few lines of a file (where "a few" means 10), so the command:

head seasonal/summer.csv

displays:

Date,Tooth

2017-01-11,canine

2017-01-18,wisdom

2017-01-21,bicuspid

2017-02-02,molar

2017-02-27,wisdom

2017-02-27,wisdom

2017-03-07,bicuspid

2017-03-15,wisdom

2017-03-20,canine

What does head do if there aren't 10 lines in the file? (To find out, use it to look at the top of people/agarwal.txt.)

## Combining tools

## Batch processing

## Creating new tools