Appendix C

Working from the Command Line

Operating systems (Windows, Linux, macOS, etc) include graphical File Managers that allow you to easily manage a range of operations associated with your file system, for example: **move**, **copy**, **delete**, **rename**, **compress** files and folders; **assign** or **remove permissions**; **install** and **execute programs**; conduct **searches**. Your operating system also provides a **text-based command line** interface that can be used to conduct not only these operations (often with additional functionality), but **any** operating system commands. An extensive overview of these commands is beyond the scope of this book but this appendix will demonstrate how to work from the command line to issue some common file management commands, which will give you some familiarity with the interface and encourage further research. Appendix G introduces two tutorials to introduce version control systems, and these tutorials make extensive of the command line.

The Terminal Window

In order to work from the command line, you must first open a **Terminal window** which provides a **command prompt** from which to issue instructions. This is straightforward but since the method varies between operating systems and versions, check how to do this on your computer.

Go ahead and open a Terminal window on your computer. You will see a **command prompt**, waiting for your next command. The prompt itself includes the file path of your current folder location within the file system, initially this is most likely your home directory (if the location includes the ~ **tilde** character this always refers to your home directory). Any commands that you issue from the prompt will be executed with reference to your current location (you can easily change your location as you will see). Each time you type a command and press the Enter key, the operating system will execute the command and may display some kind of result, or error message, before returning to the command prompt, ready for your next command.

The following list of examples will demonstrate some ways in which you can navigate your file system and manage files and folders from the command line. Note that Windows commands derive from the **Microsoft Disk Operating System** (MS DOS), while macOS and Linux commands derive from Unix so the examples will show both versions when these differ. Note also that some commands are relatively harmless and do not make changes, for example displaying the contents of a directory. Other commands do make changes, for example, renaming or deleting files or folders. Be careful that you understand its purpose before issuing any command.

(The terms folder and directory are used interchangably.)

To View the Operating System Version

Windows: type **ver** at the command prompt and press **Enter**

macOS: type **sw_vers** at the command prompt and press Enter

Linux: type **cat /etc/os-release** at the command prompt and press **Enter** (this runs the **cat** (short for **concatenate**) utility which, among other useful things, can be used to display the content of a file, in this case a file named **os-release**, which is located in the **/etc** folder).

To Clear the Screen

Sometimes it is helpful to clear the screen, for example to view a long directory all at once. Type **cls** (Windows) or **Ctrl L** (Linux or macOS).

To View the Content of the Current Folder

Windows: type **dir** and press **Enter**. A list of the files and folders in this directory will be displayed, including some additional information such as the creation date and time, **DIR** if the item is a folder, and the size of the file. If the list is too long to be seen on a single screen you can use **dir /p** which will pause the listing so that you can view it one screen at a time (just press **Enter** to continue the listing). To view only the file and folder names **across** the screen, type **dir /w** (wide).

macOS, **Linux**: type **ls** and press **Enter**. You will see a list of files and folders across the screen, with different colors to indicate files and folders. For a more detailed listing type ls -l (lower-case L) which will list each item on a separate line. Each listing will include a d at the start if the item is a directory, followed by: the permissions assigned to different user categories; the number of hard links; the owner and the group associated with the item; size; last modification date; and name. Type **ls -a** if you want the listing to include **hidden** files.

The use of **/p** or **/w** with the **dir** command, or **-l** or **-a** with the **ls** command are examples of the options that can be used with different commands. The large number of these options permit far more specificity and functionality than you can achieve using your operating system's graphical file manager.

To View the Contents of a Different Folder

You can view the content of another folder without changing your current folder location: simply use a **relative address** or **absolute address** (complete address from the root folder) to identify the location of the other folder. If you read Appendix B you will be familiar with the use of relative and absolute addresses to locate files on a Web server and you can use the same syntax when working with your local file system (refer to Appendix B for some help with relative addresses). Here are some examples using the **dir** or **ls** command (note that Windows uses the **back slash** \ character to separate items in a path, where macOS and Linux use the **forward slash** / character).

Windows:

dir images will display all files and folders in the **images** folder, located inside the current folder.

dir media\images will display all files and folders in the **images** folder which is located in a folder named **media** which is located inside the current folder.

dir .. will display all files and folders in the parent folder (the folder that **contains** the current folder).

dir ..\images to display all files and folders in the folder named **images** which is in the same parent folder that **contains** the current folder.

dir ..\..\media\images to display all files and folders in the folder named **images**, which is in a folder named **media**, which is in the folder that **contains** the folder that **contains** the current folder.

dir \Users\sarah\documents\images to display all files and folders in the **images** folder, using an absolute address starting from the root folder (the root folder is indicated by the initial back slash \).

macOS/Linux:

ls images will display all files and folders in the **images** folder, located inside the current folder.

ls media/images will display all files and folders in the **images** folder which is located in a folder named **media** which is located inside the current folder.

ls ../ will display all files and folders in the parent folder (the folder that **contains** the current folder).

ls ../images to display all files and folders in the folder named **images** which is in the same parent folder that **contains** the current folder.

ls ../../media\images to display all files and folders in the folder named images, which is in a folder named media, which is in the folder that contains the folder that contains the current folder.

ls /Users/sarah/documents/images (macOS) or **ls /home/sarah/documents/images** (Linux) to display all files and folders in the **images** folder, using an absolute address starting from the root folder (the root folder is indicated by the initial forward slash /).

Using wildcards

You can construct **search patterns** by including any number of asterisks * and question marks ? in your file names in order to restrict listings to elements that match these patterns. An asterisk in a pattern indicates any number of unidentified characters at that location within a name (including **no** characters), whereas a question mark indicates exactly one single unidentified character at that location within a name. Here some examples:

Windows:

dir *.jpg will display **all** files in the current folder with any name that ends with the **.jpg** extension.

dir m* will display all items that begin with **m**, followed by any sequence of characters, including any extensions, that are located in the current folder.

dir m*.jpg will display all items that begin with **m** followed by any sequence of characters, followed by a **.jpg** extension, that are located in the current folder.

dir m*t?.jpg will display all items that begin with **m** followed by any sequence of characters, followed by t followed by exactly one character, followed by a **.jpg** extension, that are located in the current folder.

dir ..\..\media\images*.jpg will display all .jpg files in the folder named **images** which is in a folder named **media**, which is in the folder that contains the folder that contains the current folder.

macOS/Linux:

ls *.jpg will display **all** files in the current folder with any names that ends with the **.jpg** extension.

ls m* will display all items that begin with **m**, followed by any sequence of characters, including any extensions, that are located in the current folder.

ls m*.jpg will display all items that begin with **m** followed by any sequence of characters, followed by a **.jpg** extension, that are located in the current folder.

ls m*t?.jpg will display all items that begin with **m** followed by any sequence of characters, followed by t followed by exactly one character, followed by a **.jpg** extension, that are located in the current folder.

ls ../../media\images/*.jpg will display all .jpg files in the folder named **images** which is in a folder named **media**, which is in the folder that contains the folder that contains the current folder.

To Change to Another Folder Location

Use the **cd** (**change directory**) command to change from your current location to a different folder. The cd command is the same on Windows, macOS and Linux. In the following examples **xxx**, **yyy** and **zzz** represent the names of folders on your disk (note that the forward slash / is used in these examples, for Windows be sure to substitute the back slash character \).

cd / will change your location to the **root** folder.

cd /xxx will change your location to a folder named **xxx** that is located in the root folder.

cd /xxx/yyy will change your location to a folder named **yyy** that is located inside a folder named **xxx** which is located in the root folder.

cd .. will change your location to the parent folder of the current folder (the folder that contains the current folder).

cd ../xxx will change your location to a folder named **xxx** which is located in the same parent folder as the current folder.

To Copy Files

Use the copy (Windows) or cp (macOS/Linux) to copy files. The copy command can include relative addresses and wildcards.

Windows:

copy ex1.txt ex2.txt will copy a file named **ex1.txt** to a file named **ex2.txt** that will be created in the current folder

copy ex1.txt xxx\ex2.txt will copy a file named **ex1.txt** from the current folder to a file named **ex2.txt** in a folder named **xxx** that is located inside the current folder.

copy ex1.txt ..\yyy\ex1.txt will copy a file named **ex1.txt** from the current folder to a file also named **ex1.txt** that will be created in a folder named **yyy** that is located inside

the same parent folders as the current folder. Note that, since the new file is to have the same name, copy **ex1.txt ..\yyy** would achieve the same result.

copy *.txt ..\yyy will copy all files with a .txt extension from the current folder to a folder named **yyy** that is located inside the same parent folder as the current folder.

copy xxx\ex1.txt ..\yyy\ex2.txt will copy a file named **ex1.txt** from the **xxx** folder in the current folder to a file named **ex2.txt** that will be created in a folder named **yyy** that is located inside the same parent folder as the current folder.

macOS/Linux:

cp ex1.txt ex2.txt will copy a file named ex1.txt to a file named ex2.txt that will be created in the current folder

cp ex1.txt xxx/ex2.txt will copy a file named **ex1.txt** from the current folder to a file named **ex2.txt** in a folder named **xxx** that is located inside the current folder.

cp ex1.txt ../yyy/ex1.txt will copy a file named **ex1.txt** from the current folder to a file also named **ex1.txt** that will be created in a folder named **yyy** that is located inside the same parent folders as the current folder. Note that, since the new file is to have the same name, **copy ex1.txt ../yyy** would achieve the same result.

cp *.txt ../yyy will copy all files with a **.txt** extension from the current folder to a folder named **yyy** that is located inside the same parent folders as the current folder.

copy xxx/ex1.txt ../yyy/ex2.txt will copy a file named **ex1.txt** from the **xxx** folder in the current folder to a file named **ex2.txt** that will be created in a folder named **yyy** that is located inside the same parent folder as the current folder.

To Rename or Move a File

Use the **ren** command (Windows) or **mv** (macOS/Linux) to rename a file. For example in Windows, **ren ex1.php ex2.php** will rename a file from **ex1.php** to **ex2.php**, whereas under macOS or Linux, the same operation is achieved using **mv ex1.php ex2.php**.

The **mov** command is used in Windows to move a file to another location, while macOS and Linux use the same **mv** command that is used to rename a file (since a move operation applied to a file without indicating a new folder location is equivalent to renaming the file). For example, in Windows **mov * ..** will move all the files in the current folder to the parent folder of the current folder, while **mv * ../** will perform the same operation under MacSOX or Linux.

Deleting Files

Files can be deleted using the **del** command in Windows or the **rm** command under macOS or Linux. Just as with previous commands these can include the use of relative

addresses to apply to files in other folders, and use of wildcards to operate on multiple files that match a search pattern.

Creating and Removing Folders

Windows, macOS, and Linux all use the **mkdir** command to create a new folder and the **rmdir** command to remove a folder. As a precaution, a folder must first be empty before **rmdir** can be successfully applied.

Running Programs from the Command Line

Many programs are designed to run from the command line, some are included with your operating system, while others can be installed, and some will even perform installations. For a harmless example a version of the **more** program is available on Windows, macOS and Linux; this program can be used to display the contents of a file in the Terminal window one screen page at a time (just press the space bar for the next page). For example to use more to view the contents of a file named **test.php** in the current folder you would simply type **more test.php**.

Running git Commands at the Command Line

The tutorials referenced in Appendix G provide instructions to install and run a version control program named **git**, and demonstrate how to work with this program from the command line.

Recalling Previous Commands

It can be tedious to retype the same commands time after time, especially when these commands include lengthy file paths, or when you forget the exact syntax of the command. Your operating system maintains a lengthy **history** of your previous commands, extending beyond the current session to previous sessions. You can use the **Up** and **Down** cursor keys in the Terminal window to scroll through a history of previous commands, with each command sequentially listed at the current prompt as you scroll. This makes it easy to recall and re-execute a previous command, and you can even modify a previous command to suit your current purpose.

Creating Batch or Bash Files

A **batch** (Windows) or **bash** (macOS and Linux) file is a text file that contains a list of one or more operating system commands. Once created you can "run" the file, so that

the commands listed in the file will execute in sequence. This is useful if you have to issue the same set of commands regularly (for example to process some code you are working on, start or exit a number of programs, copy a set of files to another local or remote folder location, or to perform deletions and other cleanup operations at the end of a work session).

In Windows batch files usually end with a **.bat** extension and simply contain a list of one or more commands in the correct sequence. Under macOS and Linux executable bash files usually end with an **.sh** extension (although no extension is required) and must include a specific first line that tells the location of the program (**bash**) that will execute the remaining lines. Here are examples of simple batch and bash files and how to run them:

Windows: Use any text editor to create a new text file that contains the three lines listed below, then save the file to your home (or any other) directory with the name **test.bat** (or any name with a .bat extension). Here are the three lines to add to the file:

```
@echo off
dir
echo Hello world
```

Each line is a separate command (the **@echo off** command ensures that the output will be displayed on a clean line in the Terminal window, without a prompt). To run this batch file simply go to your Terminal window and change directory to the folder where **test.bat** is located. Now type **test.bat** and press **Enter**. The commands in the file will be executed and you should see a directory listing, followed by the message "Hello World".

macOS and Linux: Use any text editor to create a new text file that contains the three lines listed below, then save the file to your home (or any other) directory with the name **test.sh** (or any name). Here are the three lines to add to the file:

```
#!/bin/bash
  ls
echo Hello world
```

Note that the line **#!/bin/bash** (often referred to as the **shebang**) must **always** appear on the very **first** line in the file. To run the commands in this file simply go to your Terminal window and change directory to the folder where **test.sh** is located. In order to treat this as an executable file you must first allow the file to be executable (files are not executable by default). You can use the **chmod** command to set the appropriate file permissions: just type **chmod 755 test.sh** (this allows you, as the owner, to **write** and **modify** the file, while anyone can **execute** it). Now type **bash test.sh** at the command line and press **Enter**. The commands will be executed and you should see a directory listing, followed by the message "Hello World".

These are, of course, simple and harmless examples. As a programmer you will find many, often very powerful, uses for batch or bash files to serve your purposes. Go carefully at first and build and test your executable files one line at a time.

Conclusion

As you can see, you can conduct any file management operations at the command line, including operations that you more usually conduct using your GUI File Manager. Once you become accustomed to the command line, and to batch processing, you are likely to find that some operations can be performed much more efficiently at the commend line, while other operations can **only** be performed at the command line.