

## Chapter 2

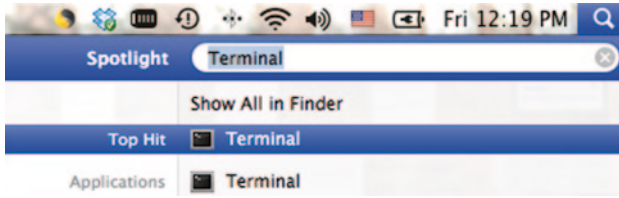
# Text Processing with the Command Line Interface

**Abstract** This chapter aims to help demystify the command line interface that is commonly used in UNIX and UNIX-like systems such as Linux and Mac OS X for language and linguistics researchers with little or no prior experience with it and to illustrate how it can be used for managing the file system and, more importantly, for text processing. Whereas most linguists are used to and comfortable with the graphic user interface, the command line interface does provide us with access to a wide range of computational tools for corpus processing, annotation, and analysis that may not be readily accessible through the graphic user interface. The specific command line interface used for illustration purposes in this chapter is the Terminal in Mac OS X, but the examples work in largely similar ways in the command line interface in a UNIX or Linux system.

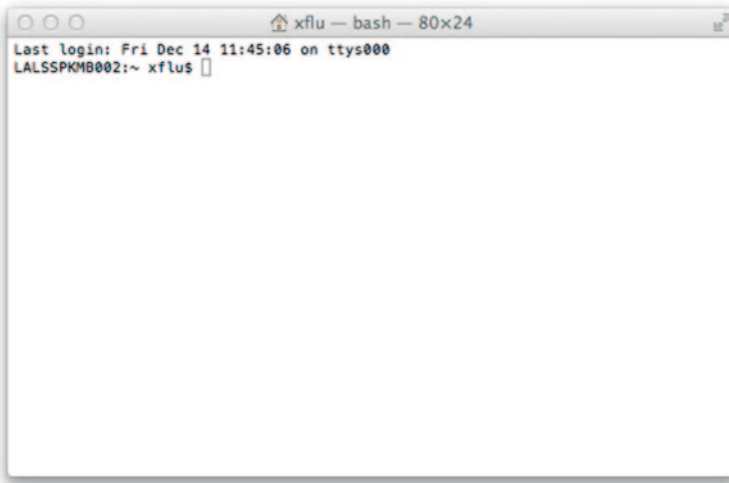
### 2.1 The Command Line Interface

If you have only used the graphic user interface in a Windows-based PC or a Mac OS X to meet your computing needs, but have never or rarely used the Command Prompt in a Windows-based PC, the Terminal in a Mac OS X, or a computer with a UNIX or Linux operating system, you probably think of the command line interface as something that is useful only for geeky scientists and engineers. However, once in a while, you may have encountered one or more text processing tools or corpus annotation and analysis programs that do not have a graphic user interface version but rather can only be invoked from the command line in a UNIX or UNIX-like system (e.g., Linux and Mac OS X), and you may have given up on them with a shake of your head. Although at first look the command line interface may not be as user-friendly and intuitive as the graphic user interface, once you have learned the basics of how it works, you will find it a versatile and powerful way of interacting with the computer. More importantly, the command line interface enables us to access a large set of useful corpus processing, annotation and analysis tools that are not conveniently available via the graphic user interface.

In this chapter, we will illustrate the use of the command line interface, beginning with a set of basic commands that are necessary for navigating the file system and then focusing on several useful tools for text processing. Additional commands will be introduced in the following chapters as necessary. The specific command line interface we will use throughout this book is the Terminal in Mac OS X, but the commands and



**Fig. 2.1** Locating the Terminal in Mac OS X via Spotlight



**Fig. 2.2** The Terminal in Mac OS X

tools covered here and in the rest of the book will work in largely similar ways in the command line interfaces in UNIX and Linux, and you should be able to follow the discussion in the book in a UNIX or Linux operating system without noticing major differences. For a complete introduction to the command line interfaces in Linux and UNIX, see Robbins (2005), Siever et al. (2009), Shotts (2012) or other similar volumes.

If you are not sure how to open the Terminal in Mac OS X, you can do this in one of the following two ways.

1. Navigate to `/Applications/Utilities` (i.e., first navigate to the `Applications` directory, then to the `Utilities` directory within the `Applications` directory), and double click on “Terminal”.
2. Click on the Spotlight icon in the menu bar (shown in the upper right corner in Fig. 2.1), type “Terminal” in the Spotlight box, and click on Terminal (listed after Top Hit and Applications in Fig. 2.1).

When the Terminal is opened, you will see a window similar to (but perhaps not exactly the same as) the one shown in Fig. 2.2. At the moment, you do not need to be concerned with the title of the window, which shows the username (in this case `xflu`), the active process name `bash`, and the dimensions of the window

80 × 24, or the first line of the window, which shows the date, time, and terminal ID of the last login. The second line of the window has three parts, first the name of the computer (in this case `LALSSPKMB002`) followed by a colon, then the name of the working directory (in this case `~`, which is short for the home directory) followed by a white space, and finally the command prompt (in this case `xflu$`). Any command you type will appear immediately after the command prompt. In the next two sections, we will first introduce a set of basic commands for navigating the file system and then several tools that are useful for text processing.

## 2.2 Basic Commands

### 2.2.1 Notational Conventions

Throughout the book, we will use the courier font to differentiate commands, file-names, and directory names from regular text. URL addresses mentioned in the main text will be enclosed in angle brackets. The actual commands to be entered in the Terminal will be given in blocks of code, as illustrated in the examples below, where `$` denotes a command prompt (do not type it) and `¶` denotes a line break (an instruction for you to press ENTER). The actual command prompt in your own Terminal will look different (as illustrated by the second line in the Terminal in Fig. 2.2), but that difference is irrelevant here. Lines in the blocks of code that do not begin with `$` and end with `¶` indicate output generated by a command, and they should not be typed or entered into the Terminal. In the case of a long command that runs two or more lines (see Sect. 2.3.6 for examples), use the command-end `¶` to determine where the command ends. You should type a multi-line command continuously (with a white space instead of a line break between lines) and press ENTER only once at the end of the command (i.e., when you reach `¶`).

In the first example below, the `echo` command is used to simply print anything you type after it on the screen. It is crucial that you type all commands exactly as they are provided, as typos as well as missing or extraneous elements (e.g., white space, single or double quotes, etc.) will likely lead to either error messages or unintended results. This is illustrated in the second example below, where a white space is missing between “echo” and “this”.

```
$ echo this is going to be fun¶
this is going to be fun
$ echothis is going to be fun¶
-bash: echothis: command not found
```

### 2.2.2 Printing the Current Working Directory

The file system is hierarchically organized, and it may be easy to lose track of where you are in the hierarchy. The `pwd` command can be used to print the location of the current working directory, as illustrated in the example below. The output shows

that my current working directory is `/Users/xflu`, i.e., in a subdirectory called `xflu` under the `Users` directory, which is also my home directory. When you first open the Terminal, you are by default located in your home directory (i.e., `/Users/yourusername`)<sup>1</sup>, which is the directory that contains your `Desktop`, `Documents`, and `Downloads` folders, among others. If you have difficulty conceptualizing where your home directory is actually located, try finding it using the Finder in your Mac (open the Finder, click on “go” in the menu bar, and then click on “Home”).

```
$ pwd
/Users/xflu
```

### 2.2.3 Listing Files and Subdirectories

The `ls` command can be used to list the contents of a directory, including files and subdirectories. As in the following example, type `ls` after the command prompt to list the contents of your current working directory. If you have not done anything else in the Terminal after opening it and typing the `pwd` command shown above, you should now see a list of subdirectories in your home directory, including `Desktop`, `Documents`, `Downloads`, and possibly a few others.

```
$ ls
Desktop Documents Downloads
```

### 2.2.4 Making New Directories

The `mkdir` command can be used to make a new directory. Let us make a new subdirectory in the home directory called `corpus` using the following example. We will be using the `corpus` directory throughout the rest of this chapter.

```
$ mkdir corpus
```

Now, try listing the contents of the current working directory again. You will see that a `corpus` directory is now shown in addition to the other directories that were shown previously.

```
$ ls
corpus Desktop Documents Downloads
```

In naming directories and files, note the following general rules:

1. Names are case sensitive.
2. Avoid white space in a file name or a directory name. Use the underscore or dash instead to concatenate different parts of a name if necessary.
3. Avoid the following characters, because they have special meanings in commands: `;`, `@`, `#`, `$`, `()<>?\'~{ } [ ] = + & ^ *`

---

<sup>1</sup>If you are using a UNIX or Linux system, the path to the home directory, specifically the part preceding the username, will look different.

### 2.2.5 *Changing Directory Locations*

The `cd` command can be used to change directory locations. For example, you can use the following example to change the current working directory to the `corpus` directory you just created.

```
$ cd corpus
```

The `pwd` command will show that your current working directory is now the `corpus` subdirectory under your home directory (`/Users/xflu/corpus` in my case).

```
$ pwd
/Users/xflu/corpus
```

At this point, let us make two subdirectories within the `corpus` directory, with the names `files` and `programs`. We will be using these two directories to store text files and programs as we work through this book. They will also be useful as we learn more commands for navigating the file system. Assuming your current working directory is still the `corpus` directory, use the first two commands below to create the two subdirectories, and then use the last command to confirm that you have created these two subdirectories successfully.

```
$ mkdir files
$ mkdir programs
$ ls
files programs
```

You can change your current working directory to a different directory by spelling out the absolute or full path to that directory. The absolute path is preceded by a `/` and starts from the root of the file system. For example, the command below can be used to change my current working directory to the `programs` directory I just created under my home directory (replace `xflu` with the name of your own home directory, i.e., your username).

```
$ cd /Users/xflu/corpus/programs
```

You can also change to another directory by specifying a relative path to that directory. A relative path specifies the location of another directory relative to the current directory, and so it starts from the current directory rather than the root of the file system. In order to explain how relative paths work, we need to first introduce two important hidden files called `."` and `."`, respectively. These files are hidden in the sense that you normally do not see them when viewing the contents of a directory. The file represented by a single dot identifies the current working directory, whereas the file represented by double dots identifies the parent directory of the current working directory. A relative path begins with one of these two filenames instead of a `/`. Assuming the `programs` directory is your current working directory, you can change your working directory to the `files` directory using the command below. In this command, the double dots take you to the `corpus` directory (i.e., the parent directory of the current working directory, which is `programs`), and `/files` then takes you to the `files` directory within the `corpus` directory.

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

Now that your current working directory is the `files` directory, try typing the first command below. This command will take you two levels up the directory hierarchy: The first double dots take you to the parent directory of the `files` directory, i.e., the `corpus` directory, and the second double dots then take you to the parent directory of the `corpus` directory, i.e., your home directory. You can verify whether this is the case with the `pwd` command, as shown in the second command below.

```
$ cd ../../
$ pwd
/Users/xflu
```

If you want to go back to the `files` directory using a relative path, you can do so using the command below. Here, the dot identifies the current working directory (which is the home directory at this moment), and `/corpus/files` identifies first the `corpus` directory within the current directory and then the `files` directory within the `corpus` directory.

```
$ cd ../corpus/files
```

In practice, however, if you are trying to get to a child or grandchild directory of the current directory, it is not necessary to type `./` and you can start directly with the name of the child directory instead. Let us return to the home directory with the first command below (where `~` is shorthand for the home directory) and then get to the `files` directory with the second command.

```
$ cd ~
$ cd corpus/files
```

Remember, if at any point you are lost in the directory hierarchy, you can always identify your current working directory with the `pwd` command, check out the contents of the current working directory with the `ls` command, and, as a last resort, return to the home directory from wherever you are with the `cd ~` command.

### 2.2.6 *Creating and Editing Text Files with UTF-8 Encoding*

In general, the text files that we will be working with will be in plain text format (saved with the “.txt” suffix) rather than Word or PDF documents (saved with the “.doc”, “.docx”, or “.pdf” suffix). It is also desirable that the plain text files (regardless of the language they are in) be saved with UTF-8 (short for Unicode Transformation Format 8-bit) encoding to ensure compatibility with the various tools we will be introducing later.

A character encoding system pairs each character in a given character repertoire (e.g., a letter in the English alphabet or a Chinese character) with a unique code (e.g., a sequence of numbers). While humans read characters the way they are written,

computers store and process information as sequences of numbers. Character encoding systems serve as a means to “translate” characters in written form into codes that can be decoded by computer programs. There are many national and international character encoding standards, which differ in terms of the number and types of characters they can encode as well as the types of codes that the characters are translated into. Not all encoding systems have a large enough capacity (or code points) to encode all characters (consider the large number of symbols required in scientific and mathematic texts), and the same character is often represented using different codes in different systems. To ensure that a text can be displayed and processed correctly by a specific computer program, it is necessary to choose an encoding system for the text that covers all the characters in the text and that is compatible with the computer program. This is especially relevant when the text is in a language other than English. To get a sense of what happens when an inappropriate encoding system is used for a text, try the following:

1. Open a web page in Chinese, e.g., `<http://www.nankai.edu.cn>` or French, e.g., `<http://news.google.fr>` in any web browser.
2. Click on “View” in the menu bar of the browser; under “Character Encoding” (in Firefox), “Encoding” (e.g., in Chrome), or “Text Encoding” (in Safari), select an encoding system that is intuitively incompatible with the web page. For example, for the Chinese web page, select an encoding system that starts with “Western”, such as “Western (ISO-8859-15)” or “Western (ISO Latin 1)”; for the French web page, select an encoding system for Chinese, such as “Simplified Chinese (GBK)” or “Simplified Chinese (GB2312)”.
3. You will see that many characters on the web pages will be displayed incorrectly.

The Unicode Standard solves the problems introduced by the existence of multiple encoding systems by assigning unique codes to characters in all modern languages and all commonly used symbols. There are seven character encoding schemes in Unicode, among which UTF-8 is the *de facto* standard for encoding Unicode on UNIX-based operating systems; it is also the preferred encoding for multilingual web pages, programming languages, and software applications. As such, it is desirable to save texts, particularly non-English texts, with the UTF-8 encoding. For further information about UTF-8 encoding or the Unicode Standard (including its other six encoding schemes) in general, consult the Unicode Consortium webpage.<sup>2</sup>

We will not look at how to create or edit plain text files through the command line interface, as in Mac OS X this can be done easily in a text editor that you are already familiar with, such as Microsoft Word or TextEdit. Let us now create a simple text file with the name `myfile.txt` and save it to the `files` folder with UTF-8 encoding. Make sure the file contains the following two lines only (press ENTER once at the end of each line), with no extra empty lines before or after them. Note that, any formatting of the text (e.g., highlighting, italicizing, bolding, underlining, etc.) will not be saved in the plain text file. If this sounds trivial to you, you can do this directly on your own and skip the next two paragraphs.

---

<sup>2</sup>`<http://www.unicode.org>`

```
This is a sample file.  
This is all very simple.
```

To generate this file using Microsoft Word, open a new file in Microsoft Word, type the two English sentences mentioned above, and then save the file in the following steps.

1. Click on “File” in the menu bar and then click on “Save As...”.
2. Enter `myfile` as the filename in the “Save As:” box and choose “Plain Text (.txt)” for the “Format:” box.
3. Locate the `files` folder (under the `corpus` subdirectory in your home directory) and click on “Save”.
4. At this point, a “File Conversion” dialog box will pop up (see Fig. 2.3). Click on “Other encoding” and then choose “Unicode 6.0 UTF-8”. Choose “CR/LF” for “End line with:”. Click on “OK”.

TextEdit can be used for the same purpose in a similar fashion. To open TextEdit, type “TextEdit” in the Spotlight box and then click on “TextEdit”, similar to how you opened the Terminal (see Fig. 2.1). Now type the two English sentences mentioned above in the editor. To save the file in plain text format with the name `myfile.txt` in the `files` folder, follow the following steps:

1. Click on “Format” in the menu bar and then click on “Make Plain Text”.
2. Click on “File” in the menu bar and then click on “Save”.
3. Enter `myfile` in the “Save As:” box and choose “Unicode (UTF-8)” for the “Plain Text Encoding:” box.
4. Locate the `files` folder (under the `corpus` subdirectory in your home directory), and click on “Save”.

## 2.2.7 Viewing, Renaming, Moving, Copying, and Removing Files

In this section, we will learn a set of commands that can be used to view, rename, copy, delete, and move files. Whereas you can perform these tasks easily with the graphic user interface, you will find it more efficient to get them done via the command line interface sometimes, especially when you are dealing with a large number of files or if you are already working on some files via the command line.

Before we start, first make sure that you have created the file `myfile.txt` and saved it to the `files` folder following the instructions in Sect. 2.2.6. Next, go to <http://tinyurl.com/corpusmethods> (hosted on Google Drive) and download the following three files: `mylist.txt`, `mypoem.txt`, and `speech.txt` to the `files` folder. We will be using these files for illustration purposes throughout the rest of this chapter. The file `mylist.txt` contains part-of-speech and frequency information for the 3,000 most frequent unlemmatized words in the British National Corpus (BNC). Each row in the file contains three tab-delimited columns or fields: a word (in lowercase), a tag indicating its part-of-speech category, and its frequency



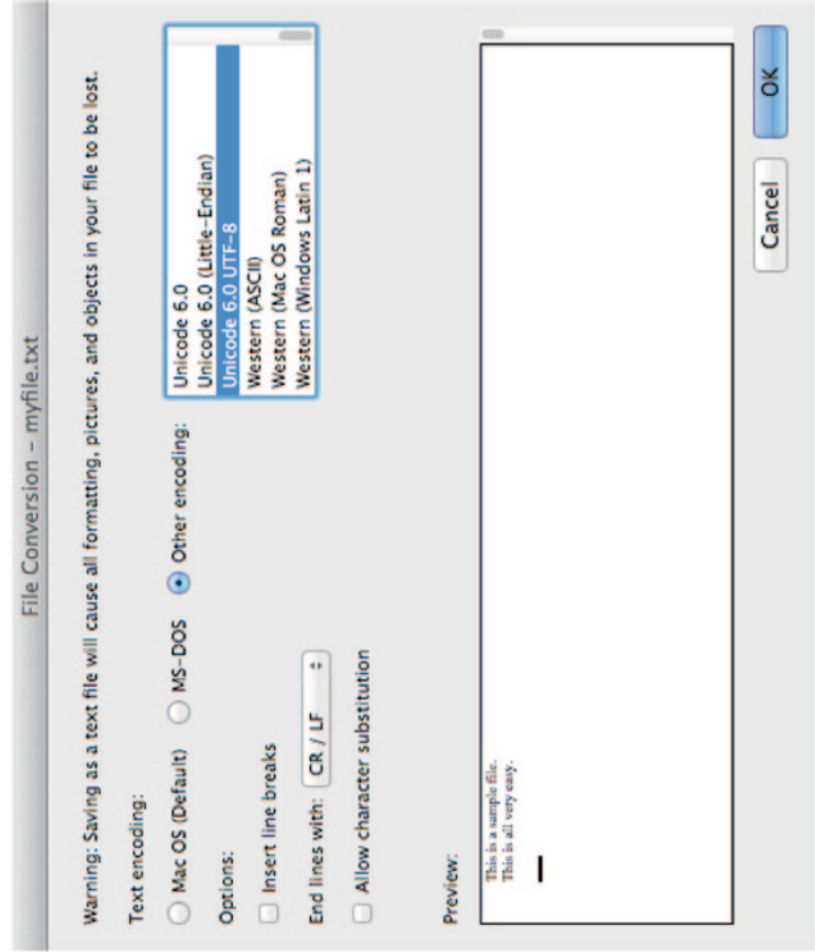


Fig. 2.3 Selecting UTF-8 encoding when saving a file in plain text format in MS Word

in the BNC.<sup>3</sup> The part-of-speech tags will be discussed in detail in Chap. 3. The file `mypoem.txt` contains a short poem “Men Improve with the Years” by the Irish poet William Butler Yeats. Finally, the file `speech.txt` contains the transcript of the speech “I Have a Dream” delivered by Martin Luther King, Jr. on August 28, 1963.

If for any reason your current working directory is no longer `files`, change it back to `files` using the first command below, and then use the second command to verify that it contains the following four files: `myfile.txt`, `mylist.txt`, `mypoem.txt`, and `speech.txt`.

```
$ cd ~/corpus/files
$ ls
myfile.txt mylist.txt mypoem.txt speech.txt
```

The `more` command can be used to display the content of a text file on the screen. Use the first example below to view the content of `myfile.txt`. Since the text is short, the command prompt will be displayed in the next line immediately following the end of the text. Use the second example below to view the content of `mylist.txt` (the output of the command is omitted here). As the text has 3,000 lines and is longer than the remaining space in the Terminal, only the first screen is shown. You can press the SPACE bar on the keyboard to continue to the next screen or press Q on the keyboard to exit the file and return to the command prompt.

```
$ more myfile.txt
This is a sample file.
This is all very simple.

$ more mylist.txt
```

If you want to know the size of a file, you can use the `wc` command to display the number of lines, words, and characters in it. The first example below shows that `myfile.txt` has 2 lines, 10 words (as delimited by white space), and 46 characters (including white spaces and line breaks).

```
$ wc myfile.txt
2 10 48 myfile.txt
```

In the case of a long file, sometimes you may wish to view only the first or last few lines, instead of the whole file. The `head` and `tail` commands can be used for these purposes. The first example below shows the first 10 lines (by default, including empty lines) of `mypoem.txt`. You can also specify the exact number of lines you wish to view from the top with a command line option, in this case a dash followed by a number. This is illustrated in the second example below, which shows the first 5 lines of `mypoem.txt`.

```
$ head mypoem.txt
Men improve with the Years
```

---

<sup>3</sup>This file was adapted from the file `all.num.o5` made publicly available by Adam Kilgarriff at <<http://www.kilgarriff.co.uk/BNClists/all.num.o5>>.

Computational Methods for Corpus Annotation and  
Analysis

Lu, X.

2014, XI, 186 p. 22 illus., Hardcover

ISBN: 978-94-017-8644-7