Advanced Programming

Lab Sheet 6 – Regular Expressions

Introduction

This lab task encourages you to gain some experience in a useful technology supported by most programming languages, and by many non-programming tools also. The exercises focus on regular expressions, a powerful and performant means of text pattern recognition. You can do some background reading on regular expressions here.

Regular expressions are a standardised method for finding text inside larger blocks of text, much as the standard find/replace dialogue boxes you're used to accessing in various applications. Regular expressions work across programming languages and/or tools, and as such are a good example of highly transferable knowledge: learn to use them in one context, and its easy to do so elsewhere.

Task 1: Setting Up

You are going to use a command-line utility for using regular expressions to match text in files named *grep*. The grep utility ships as part of the standard package of utilities on UNIX/Linux operating systems, but is not included with windows. Fortunately, the standard package of UNIX utilities is available as a single program that runs on windows, named busybox. You can download the windows build here, about halfway down the page: look for the underlined busybox.exe link.

Busybox is designed to be a very compact copy of the standard UNIX utilities, and is commonly installed on home routers, which often run a cut-down distribution of the Linux operating system, and have very limited storage space. Once you've downloaded busybox's .exe file, you can run it in one of two ways:

1. Open a command prompt (cmd), navigate (cd) to the folder in which you saved the busybox .exe file, and run "busybox bash" (without quotes) to start a linux shell

2. Create a shortcut to the busybox.exe file, right click the shortcut and select properties, and add " bash" to the end of the shortcut's target (no quotes. Note the space!)

Regardless of which method you use, you'll end up with a standard-looking Linux shell, which will accept most standard terminal commands (cd, ls, rm, mkdir, rmdir, mv, unzip, wget, grep). You

might want to spend a few minutes familiarising yourself with these commands if you aren't used to them.

Task 2: Using Grep

I have uploaded a zip file (holmes.zip) to moodle, which contains the Project Gutenberg version of the Sherlock Holmes novels and short story collections. Quite a lot of text! You'll need to download the zip file, unzip the contents to a convenient directory (you can use the unzip command in busybox!) for working. Your task is to use grep to find:

1. How many times does the word "elementary" appear in the literature? Look at the context in which the word appears, what isn't as you might have expected?

2. The books aren't consistent in their spelling of the word Colour. Sometimes the correct spelling is used, and sometimes the Americanised "color" spelling is used instead. How many of each spelling is there in the collection?

3. How would you use grep to find both spellings of the word colour, but without finding partial word matches (e.g. "discoloured" or "colourless")

4. What percentage of the words in the novels contain just 1 character? 2? 3?... 25? How does that compare with average word lengths calculated by others? (e.g. this)

You might find the official grep documentation, available here, useful in completing this task.