Word Count Code Challenge

This task involves implementing a word count application. This application is very similar to "wc", the venerable Unix command line utility. So from now on we'll call this application "WordCount". WordCount is a command line application. It reads a file as a command argument and process the words contained in that file. There is no need for a GUI or a database to persist state between invocations.

Write the code in your favourite programming language, be inspired by other implentation out there, but of course copying existing solution will automatically terminate the hiring process.

You may use open source libraries but they must be retrievable via public repositories and your application must build easily and be deployable using some quick command to pull in any relevant dependencies, compile and run.

This exercise is an opportunity for you to demonstrate your technical abilities. You should seek opportunities to demonstrate your knowledge of good engeneering practises as well as wider design principles. Show off your craftsmanship. Even though the exercise is simple you should imagine you are building it for performance and scalability.

Feature requirements

WordCount accepts zero or one command line options. Consider them mutually exclusive (it is not necessary to implement a solution that accepts combinations of command line option, only the first one will be considered). WordCount should:

1. Output the total number of words in a file:

```
$> echo "Very nice ad-hoc counting." > temp.txt
$> wordcount temp.txt
$> 5
```

Consider "word" any group of 1+ consecutive letters or numbers (`[a-zA-Z0-9]`) separated by something outside this set, like spaces or punctuation. For instance, "ad-hoc" are two words.

2. Output additional metrics, such as number of lines and characters:

```
$> echo "Very nice counting. Really very nice!" > temp.txt
$> wordcount --all temp.txt
$> line:1, words:6, chars:30
```

Please only consider letters and numbers for character count (`[a-zA-Z0-9]`). Punctuation, spaces or other special characters should not be counted. Lines are separated by new-lines chars (`"\n"`).

3. Show frequencies, like for example:

```
$> echo "Very nice counting. Really very nice" > temp.txt
$> wordcount --frequencies temp.txt
$> ["nice" 2] ["very" 1] ["Really" 1] ["counting" 1] ["Very" 1]
```

In this case WordCount will be case sensitive. Use the same definition of word as point 1.

4. Have WordCount to run on concurrently on multi-core hardware to speed up processing

```
$> curl http://www.gutenberg.org/cache/epub/2600/pg2600.txt > war-and-peace-full-book.txt
$> wordcount --parallel war-and-peace-full-book.txt
$> 566321
```

For the last point it is expected that >1 core of a multi-core machine is busy computing the word count. Most

modern languages contain some facility or library to achieve this. In case the selected language does not contain such a facility, it is okay to show that the word count is happening on multiple (green) threads.

Please also provide

- Quick description of any assumptions you are making, general organisation of the code, design.
- Quick "how to" run the application.
- Source code, related artifacts.