Coding Challenge #1 - Build your own wc!

JOHN CRICKETT

18-MAR-2023

18 10 Share

The Unix command line tools are a great metaphor for good software engineering and they follow the Unix Philosophies of:

- Writing simple parts connected by clean interfaces - each tool does just one thing and provides a simple CLI that handles text input from either files or file streams.
- Design programs to be connected to other programs - each tool can be easily connected to other tools to create incredibly powerful compositions.

Following these philosophies has made the simple unix command line tools some of the most widely used software engineering tools - allowing us to create very complex text data processing pipelines from simple command line tools. There's even a Coursera course on Data Engineering with Bash!

Linux and Bash for Data Engineering

You can read more about the Unix Philosophy in the excellent book The Art of Unix Programming.

The Challenge - Building wc

The functional requirements for wc are concisely described by it's man page - give it a go in your local terminal now:

```
man wc
```

The TL/DR version is: **wc** – word, line, character, and byte count.

Step Zero

Like all good software engineering we're zero indexed! In this step you're going to set your environment up ready to begin developing and testing your solution.

I’ll leave you to setup your IDE / editor of choice and programming language of choice. After that here’s what I’d like you to do to be ready to test your solution, download the text from:

`https://www.gutenberg.org/cache/epub/132/pg132.txt`

and save it as test.txt

Step One

In this step your goal is to write a simple version of wc, let’s call it ccwc (cc for Coding Challenges) that takes the command line option -c and outputs the number of bytes in a file.

If you’ve done it right your output should match this:

```
>ccwc -c test.txt
  341836 test.txt
```

If it doesn’t, check your code, fix any bugs and try again. If it does, congratulations! On to…

Step Two

In this step your goal is to support the command line option -l that outputs the number of lines in a file.

If you’ve done it right your output should match this:

```
>ccwc -l test.txt
    7137 test.txt
```

If it doesn’t, check your code, fix any bugs and try again. If it does, congratulations! On to…

Step Three

In this step your goal is to support the command line option -w that outputs the number of words in a file. If you’ve done it right your output should match this:

```
>ccwc -w test.txt
   58159 test.txt
```

If it doesn’t, check your code, fix any bugs and try again. If it does, congratulations! On to…

Step Four

In this step your goal is to support the command line option -m that outputs the number of characters in a file. If the current locale does not support multibyte characters this will match the -c option.

You can learn more about programming for locales here: https://learn.microsoft.com/en-us/globalization/locale/locale-and-culture

For this one your answer will depend on your locale, so if can, use wc itself and compare the output to your solution:

```
>wc -m test.txt
  339120 test.txt

>ccwc -m test.txt
  339120 test.txt
```

If it doesn’t, check your code, fix any bugs and try again. If it does, congratulations! On to…

Step Five

In this step your goal is to support the default option - i.e. no options are provided, which is the equivalent to the -c, -l and -w options. If you’ve done it right your output should match this:

```
>ccwc test.txt
    7137   58159  341836 test.txt
```

If it doesn’t, check your code, fix any bugs and try again. If it does, congratulations! On to…

The Final Step

In this step your goal is to support being able to read from standard input if no filename is specified. If you've done it right your output should match this:

```
>cat test.txt | ccwc -l
    7137
```

If it doesn't, check your code, fix any bugs and try again. If it does, congratulations! You've done it, pat yourself on the back, job well done!

Help Others by Sharing Your Solutions!

Even though this is only the first coding challenge, I've already been asked to share model solutions in as many programming languages as possible. Which is an awesome idea, but I just don't have the time and don't know even half the programming languages out there so I'd like your help!

If you think your solution is an example other developers can learn from please share it, put it on GitHub, GitLab or elsewhere. Then let me know - ping me a message via Twitter or LinkedIn or just post about it there and tag me.

I'll take a look at every solution and I'll mention the best solution in the following week's newsletter.

Request for Feedback

I'm writing these challenges to help you develop your skills as a software engineer based on how I've approached my own personal learning and development. What works for me, might not be the best way for you - so if you have suggestions for how I can make these challenges more useful to you and others, please get in touch and let me know. All feedback is greatly appreciated.

You can reach me on Twitter, LinkedIn or through SubStack.

Thanks and happy coding!

John

Coding Challenges

A weekly Coding Challenge to help software engineers level up.

By John Crickett

Type your email... Subscribe

18 Likes · 1 Restack

10 Comments

Write a comment...

Isaac Byron Writes Isaac’s Substack Sep 21 Liked by John Crickett

I am a begginner , I tried this in golang , any experienced golang engineers can leave me a review and I will correct my code . It can be found here

https://github.com/UnplugCharger/coding-challenge/blob/master/01_wc/main.go

LIKE (2) REPLY SHARE

3 replies by John Crickett and others

Clov Aug 31 Liked by John Crickett

Here is my solution using good old C 😊 70-80% done ✔️

https://github.com/clovisphere/wc

(I just came across these challenges. Let's see how far I will go)

LIKE (2) REPLY SHARE

8 more comments...