Coding Challenge #1

John Crickett
Helping you become a better software engineer though coding challenges that

3 articles ✓ Following

March 18, 2023

This week's challenge is to build your own version of the Unix command line tool wc!

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 following text: https://www.gutenberg.org/cache/epub/132/pg132.txt and save it as text.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

Report this

Published by

John Crickett
Helping you become a better software engineer though coding challenges that build real applications.
Published • 1w

3 articles

✓ Following

I've just published the first coding challenge for my Coding Challenge newsletter. In this challenge you walk through building your own version of the unix command line tool wc.

As an aside I introduce some concepts from the Art of Unix Programming and introduce the idea of data engineering with unix command line tools and bash!

#coding #programming #unix #dataengineering

Like Comment Share

Thayub J and 33 others 12 comments

Reactions

+22

12 Comments

Most relevant

Add a comment...

Mikhail Golubitsky • 3rd+ 1w •••
Software Engineer

Thank you, this was really fun in Ruby! I might try it in a language I don't know well.

That Final Step required some interesting refactoring!

...see more

Like · 2 | Reply · 3 Replies

Load previous replies

John Crickett • Following 6d •••
Helping you become a better software engineer though coding challenges that build real applications.

I love that you added some automated tests Mikhail Golubitsky!

Like · 2 | Reply

Mikhail Golubitsky • 3rd+ 6d •••
Software Engineer

John Crickett having those tests in place made it very quick to catch mistakes I was making during refactoring for the Final Step!

Like · 1 | Reply

Chaw Kok Fei (He/Him) • 3rd+ 6d •••
Technology Intern @ Deloitte | I Bring Codes to Life

Not sure my Python solution is qualified as a Unix command line tool or not but here is my first ever attempt to build one. Nevertheless, I really enjoyed the process of building the solution for the challenge. Thank you, please keep it coming!

Like · 1 | Reply · 2 Replies

John Crickett • Following 6d •••
Helping you become a better software engineer though coding challenges that build real applications.

Thanks Chaw Kok Fei, glad you enjoyed it. That's a good start, you might be interested in checking out Click or Typer as nice libraries to handle command line arguments in Python.

If you want to package a Python script for better

Like | Reply

Chaw Kok Fei (He/Him) • 3rd+ 6d •••
Technology Intern @ Deloitte | I Bring Codes to Life

Thanks for the tips, will definitely check them out.

Like · 1 | Reply

Load more comments

Coding Challenges

A weekly coding challenges for software engineers looking to level up or learn a new programming language

1,316 subscribers

Subscribed

More from this newsletter

Coding Challenge #2

John Crickett on LinkedIn

Published on LinkedIn

Welcome to the Coding Challenges weekly newsletter!

John Crickett on LinkedIn