Learn a Command-line Interface

A modern introduction to age-old sorcery.

Kris Jordan

Why a command-line interface?

Computers are machines. Their original intent was to help us do work at previously unthinkable scales analyzing large data sets, generating reports, finding patterns, and so on. As the availability and distribution of computers spread beyond universities and businesses and into our backpacks and pockets, their widespread use is as much about *accessing* and *sharing* information as it is about *doing work*. In this transition, most people interacting with computers are typically software *consumers* as opposed to system *operators* and software *creators*.

If you want to succeed as a *data-driven manager*, an *analyst*, an *engineer*, or a *scientist*, the ability to effectively *operate* a computer and *develop* programs will give you super powers. The command-line interface popularized by Bell Labs' Unix system in the 1970s continues to serve as the foundation of modern operations and development. Despite the advent of operating systems with graphical user interfaces such as Microsoft's Windows or Apples macOS, the textual interface of a command-line offers much more power, flexibility, and simpler automation if you're willing to climb your way up its steeper learning curve.

The purpose of this text is to help you gain comfort working at a command-line. It assumes no prior experience and intends to help you establish a mental model of the shell, file system, process model, and important tools that is technically sound without being pedantically overwhelming. After you understand a handful of key concepts, a new world of possibilities and capabilities opens up to you.

0.1 Special Thanks

Many colleagues and students provided feedback and corrections to this text. Special thanks to Madison Huber, Helen Qin, Jeffrey Young, Solomon Duncan, Christina Barta, Baron Northrup, and Eric Schneider for their suggestions and corrections.

Getting Started

The examples in these lessons center around a Unix-based operating system. Unix's ideas date back to the early 1970s. These ideas have withstood the test of time.

A historical challenge of learning these ideas and skills was gaining access to a system configured for learning. No more!

Thanks to innovative, free, open source software, you can gain access to the same command-line tool bench used by world class engineers and data scientists, on your personal computer. The learning exercises in these lessons give you a *sandbox*: the programs you run are isolated from your primary operating system and files so as to avoid any accidental deletion of important files. You are safe to, and *encouraged to* experiment!

The sandboxing you'll use in these tutorials are provided through containers. A container allows you to run programs in a controlled, reproducible environment. Containers were invented to simplify industrial systems operations and are used as infrastructural building blocks by Google, Amazon, Facebook, and more. In the past 10 years, container technology became more user friendly and readily accessible.

Containers offer educational benefits, too. Once containers are up and running on your personal computer you can quickly and easily replicate the systems, tools, and configurations used throughout these tutorials.

0.2 Windows Installation

0.2.1 Update to Windows 10, Version 2004+

Docker for Windows runs best on Windows Subsystem for Linux 2 (WSL2), a feature that is included in the lastest versions of Windows but requires some configuration to enable. Check your Windows version number before continuing:

Press `Windows Button + R`, type `winver`, and press Ok or Enter. You should see at least:

- Version 2004 or later
- Build 19041 or later

If it is not, visit the following URL and look for the Windows 10 May 2020 Update area. Follow the steps needed to upgrade your operating system:

https://www.microsoft.com/software-download/windows10

0.2.2 Installing WSL2

Microsoft's instructions for enabling WSL2 have steps that involve "Open PowerShell as Administrator". PowerShell is a Windows-specific command-line interface. Running it as "Administrator" gives it privileges to modify system settings it couldn't otherwise. To do this, you will need to open your start menu, search for PowerShell, rightclick on it and *Run as administrator*.

Follow Microsoft's instructions for enabling WSL2 by completing the only the first two sections of the following guide: https://docs.docker.com/docker-for-windows/wsl/ - you do not need to install a Linux distribution, or the steps after it.

Update the WSL2 kernel by following these instructions: https://docs.microsoft.com/en-us/windows/wsl/wsl2-kernel

Finally, run Powershell as an Administrator once more, and then run the following commands (press enter after each line):

```
wsl --set-default-version 2
Set-ExecutionPolicy RemoteSigned
```

0.2.3 Installing Docker

Follow Docker's instructions to install Docker Desktop Stable:

https://docs.docker.com/docker-for-windows/wsl/#download

You can stop at Step 4 after confirming Docker is using the WSL2 based engine.

0.2.4 Installing Windows Terminal

The recommended Terminal software for this course is the Windows Terminal app made by Microsoft. You can install it from the Microsoft Store app by searching for "Windows Terminal".

0.2.5 Installing `git` and Cloning the `learncli` Repository

After installing Windows Terminal, start it normally, *not* as an administrator.

Confirm whether you have `git` installed by running the command:

```
git --version
```

If you get an error, please follow the instructions to download and install git here:

https://git-scm.com/downloads

After `git` is installed, restart Windows Terminal normally and run the following `git` command:

```
git clone https://github.com/comp211/learncli211.git
```

You'll learn exactly what this command is doing, and much more, soon!

0.2.6 Beginning a Shell Session

After completing the install steps above, you should be able to navigate into the `learncli211` directory with the *change directory* (`cd`) command and then run a *PowerShell* script (the `.ps1` file) to start up the `learncli` container. These are the steps you'll follow in the future to get back into your `learncli` container:

```
cd learncli211
./learncli.ps1
```

If all is well, then you will see some UNC CS ASCII art and a `learncli$` prompt string awaiting your command. Woohoo! That's all you need to accomplish for now.

End your container session by running the command `exit` command, don't just close your Terminal Window else the container will run in the background.

0.3 macOS Installation

0.3.1 Update to Catalina 10.15.6+

Confirm your mac operating system is Catalina, version 10.15.6 or greater, by opening the Spotlight Search and searching for *About this Mac*. If your operating system version is not 10.15.6 or greater, click the *Software Update...* button and install recommended updates.

0.3.2 Installing Docker

To install Docker on macOS, please follow the official instructions:

https://docs.docker.com/docker-for-mac/install/

0.3.3 Installing `git` and Cloning the `learncli` Repository

Open a Terminal via *Applications* > *Utilities* > *Terminal*, or by searching for it in Spotlight Search.

Run the command:

```
git --version
```

If prompted that you need to install git, follow the instructions to install it. If you see an error, try running `xcode-select --install`

Once `git` is installed, close and open a new Terminal window and run the following command. You'll learn more about what exactly this is doing soon!

```
git clone https://github.com/comp211/learncli211.git
```

0.3.4 Beginning a Shell Session

If you do not have a Terminal window open, go ahead and do open one via *Applications* > *Utilities* > *Terminal*, or by searching for it in Spotlight Search.

Then, run the following:

```
cd learncli211
./learncli.sh
```

If all is well, then you will see some UNC CS ASCII art and a `learncli$` prompt string awaiting your command. Woohoo! That's all you need to accomplish for now.

End your container session by running the command `exit` command, don't just close your Terminal Window else the container will run in the background.