---

title: The Unix Shell
subtitle: Introducing the Shell
minutes: 10

The Unix Shell: Introducing the Shell

Computer-Human Interfaces

At a high level, computers do four things:

- run programs
- store data
- communicate with each other
- interact with us

They can do the last of these in many different ways, including direct brain-computer links and speech interfaces. Since these are still in their infancy, most of us use windows, icons, mice, and pointers. These technologies didn't become widespread until the 1980s, but their roots go back to Doug Engelbart's work in the 1960s, which you can see in what has been called "The Mother of All Demos (http://www.youtube.com/watch?v=a11JDLBXtPQ)".

Going back even further, the only way to interact with early computers was to rewire them. But in between, from the 1950s to the 1980s, most people used line printers. These devices only allowed input and output of the letters, numbers, and punctuation found on a standard keyboard, so programming languages and interfaces had to be designed around that constraint.

The Command Line

This kind of interface is called a **command-line interface**, or CLI,
to distinguish it from the **graphical user interface**, or GUI, that most people now use.

The heart of a CLI is a **read-evaluate-print loop**, or REPL: when the user types a command and then presses the enter (or return) key, the computer reads it, executes it, and prints its output.
The user then types another command,
and so on until the user logs off.

The Shell

This description makes it sound as though the user sends commands directly to the computer, and the computer sends output directly to the user. In fact,
there is usually a program in between called a **command shell**.

What the user types goes into the shell; it figures out what commands to run and orders the computer to execute them.

Note, the reason why the shell is called *the shell*: it encloses the operating system in order to hide some of its complexity and make it simpler to interact with.

A shell is a program like any other. What's special about it is that its job is to run other programs rather than to do calculations itself. The commands are themselves programs: when they terminate, the shell gives the user another prompt ($ on our systems).

Bash

The most popular Unix shell is **Bash**, the Bourne Again Shell (so-called because it's derived from a shell written by Stephen Bourne --- this is what passes for wit among programmers). Bash is the default shell on most modern implementations of **Unix**, and in most packages that provide Unix-like tools for Windows.

Why Use a Shell

Using Bash or any other shell sometimes feels more like programming than like using a mouse. Commands are terse (often only a couple of characters long),
their names are frequently cryptic, and their output is lines of text rather than something visual like a graph.

On the other hand, the shell allows us to combine existing tools in powerful ways with only a few keystrokes and to set up pipelines to handle large volumes of data automatically.

In addition, the command line is often the easiest way to interact with remote machines. As clusters and cloud computing become more popular for scientific data crunching, being able to drive them is becoming a necessary skill.

Today we'll learn the bare necessities of bash, so that you can go on to learn other programming languages like R and Python, and be comfortable creating, executing, and deleting these scripts in the Unix filesystem.

Adapted from: [Software Carpentry (http://swcarpentry.github.io/shell-novice/)](http://swcarpentry.github.io/shell-novice/)