History of the Shell

Table of Contents

Overview of the Shell

- the shell is a interfaced layer of abstraction of the operating system that allows for people to communicate commands to the OS without getting into too much detail
  - within the operating system, there is the program, data and the other stuff that links together many programs
  - glorified configuration language process
- the shell is an instance of a command line interface which are powerful since they are scriptable

Layers of Abstraction

```
 +------------------------+
| apps | # The app layer is what you are familiar with
+------------+-----------+
| | libs | # Libraries that the operating system will inherit
| kernel +-----------+
| | C stdlib | # The kernel is the operating system that manages states and such
+------------+-----------+
| hardware | # The hardware that connects with the machine code to do work
+------------------------+
```

- sh and Emacs and any of their independent instances are themselves applications, one of many that sit atop the operating system.

Introspection: When a program looks at itself ("when we use tools to find out more about our tools"). Knowing how to perform introspection is a portable, universal skill that lets you explore or relearn something about an unfamiliar program.

Priviledges of the Shell

- Superusers ("root") are the only ones with permission to kill PID 1, system.
- The sudo command lets you run a command as root: `sudo sh`
- Killing instances of processes can be done with `kill pid_num`
- We want to have these priviledges so that the user can have limited control over what they need to solve their problem. For instance, if we give a shell script to a grader, we want to give them read and execute, rather than write to ensure there are no errors

Bourne Again Shell (Bash)

- A shell is itself a program, and programs themselves are just files that can be executed by the operating system.
- sh is the predecessor to bash (Bourne Again SH). sh was designed to work on 16-bit machines so it's a very little language. Bash adds some features in addition ot the original sh.
  - There is also a lot of other shell languages ending in sh. Having so many distinct shell languages becomes a

problem, so the POSIX standard was created as a spec for shells.

- Creating another instance of the shell from within the shell itself to execute a one-off command: `sh -c <command>`

This is what you call a subprocess, or a child process. Within your running instance of `sh` (the CLI you're typing into), another instance of `sh` is spawned as a child of that `sh`. In fact, every time you run a command, an instance of their little program is attached to your shell as a child process. You can see this for yourself when using commands like `ps` to show processes and their parentage.