Shell in C: Part 2

Technical Goals

Learn about…

- Shells
- file descriptors/redirection
- Using Pipes
- Environment variables
- Builtins

Prerequisite knowledge

Piping

Shells have a feature called piping. This means you can connect commands together such that the output (standard output) of a command becomes the input (standard input). The final command should print to the regular terminal output.

Example:
echo “Hello” # this will write “Hello” to standard output (stdout)
wc # Reads standard input (stdin) and counts the number of lines, words, and bytes
| is the symbol for piping

echo “Hello” | wc
> 1 1 6
* The 6 is because echo appends a \n to the end by default (\n is one character)

Builtins

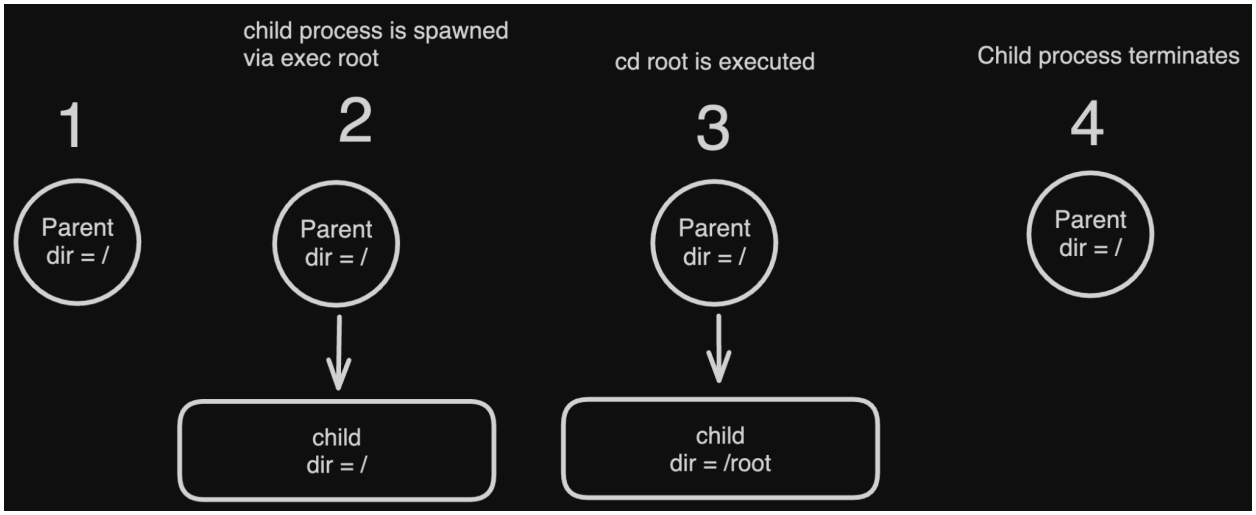
There are some features that are not provided by commands, but by the shell itself. For info on your default shell builtins see `man builtins`. Let's look at an example:

You may or may not have noticed `cd` (change directory) does not work in your personal shell. That is because it is not a program on your computer. There exists a system call called chdir(2), so why is it not a program, and how does it work in other shells like bash or zsh?

Let’s imagine there is an executable program called `cd` that changes the directory of its running process to whatever was specified.

Nothing would happen in the parent process. This is because the effects of cd would happen in its own process - the child, and since processes are independent of each other, it would not affect the parent process.

This means we should hard code `cd` into the shell itself.



Deliverable

1. Implement piping

You should use IPC we learned about on Monday to complete this using `pipe`, `dup`, and/or `dup2`

2. Implement the cd builtin using chdir(2)

This is due next Monday before the Education Meeting.

Feel free to come to my office hours Monday/Wednesday in Volen from 4-5pm to ask any questions, message me on slack, or set up a time to meet!