Homework 2
Shells and Your Environment

Assigned: Friday, January 15, 11:00AM

Due: Friday, January 22, 11:00AM (Hard Deadline)

Submission Instructions

Submit this assignment on Gradescope. You must submit every page of this PDF. We recommend using the free online tool PDFescape to edit and fill out this PDF. You may also print, handwrite, and scan this assignment.

1 Understanding your environment

When you open a new terminal window, a lot has actually happened behind the scenes before your prompt shows up. In lecture we introduced the files `~/.bashrc` and `~/.bash_aliases` as some files that are read when a new session starts, however, there more than just those.

List all of the files read by your shell during startup that contribute to your environment. List files in the order that they are read. *Hint: Don't forget about global files, such as those in* `/etc`

Run the command `set`.[1] This command prints your current environment, that is, all of the variables currently set to any value. Also try the command `env`, which only includes variables that have been "exported", that is variables that will be set for child processes. Why do you think so many variables are set, but not exported?

Pick a variable that is set but not exported (something printed by `set` but not by `env`). Explain why you think that variable is useful for the shell process, but not for a child process spawned (created) by the shell:

While some variables should be familiar from the first part of this question, there are many more variables in your environment than those set by `.bashrc` and friends.

Pick a variable not set during shell startup. Explain where that variable comes from (what sets it) and what its value means *(do not use the same variable as the previous question)*:

[1] That dumps a lot of text to the screen. Try running `set | less` to get a scrollable view or `set > output.txt` to save the output to a file for easier viewing.

2 Special Variables

Bash has quite a few special variables that can be very useful when writing scripts or while working at the terminal.

What does the variable `$?` do? Give an example where this value is useful

What does the variable `$_` do? Give an example where this value is useful

3 Understanding your PATH

In a terminal, type `PATH=` (just hit enter after the equal sign, no space characters anywhere). Try to use the terminal like normal (try running `ls`). What happened?

Give an example of a command that used to work but now doesn't:

Can you still run this command with an empty `PATH`? How?

Give an example of a command that works the same even with an empty `PATH`. Why does this command still work?

4 Basic Scripting

Recall from lecture that scripting is really just programming, only in a very high-level language. Interestingly, `sh` is probably one of the oldest languages in regular use today.

`make` is a good tool for build systems, but we can actually use some basic scripting to accomplish a lot of the same things. First, write a simple C program that prints "Hello World!". Write a shell script named `build.sh` that performs the following actions:

1. Compile your program
2. Runs your program
3. Verifies that your program outputs exactly the string "Hello World!"
4. Prints the string "All tests passed." if the output is correct, or prints "Test failed. Expected output >>Hello World<<, got output >>{the program output}<<".

Copy the output of `cat build.sh` here: