Mentor Notes

Problems

- Question 1 should be fairly straight-forwards, just getting them to use `fork()`, `exec()` (one of the variants) and `wait()`. Part 2 is just learning how to use the `pipe2()` syscall
- Question 2 part 2, they won't see the `exec()` because they run on the child processes. Part 3 should serve to illuminate
- You'll notice that bash doesn't search over the PATH when you run `ls`. That's because the binary will already be in its cache
- For q2 p4, `echo` is a builtin, so they won't see a `fork()` (or rather, a clone). Bash just… Prints it
- For q2 p5, you should be able to identify a bunch of `stat()` calls. These are bash searching over the path. Notice this only happens because the bash cache doesn't contain an entry for `potato`.
- For p2 p6, `zsh` does something I frankly find hillarious. It just `fork()`'s and proceeds to `exec()` every single possible binary in PATH, trying the next one if it gets an error. I haven't tested it, but I suspect this is honestly very comparable in time to the `stat()` solution (as internally the kernel would have to check if the file exists to perform the exec). It might even be faster, as it doesn't have to return the full stat struct, but any differences are probably negligible.
- For problem 3, you might want to read the implementation of nohup either here or here (the former is easier to read but is old and I suspect is what FreeBSD uses. The latter is written by our own Jim Meyering and is what most Linux systems have installed). This is basically what they're implementing, but without most of the complexity (they should just do the `signal()` and the `exec()`).