Lecture 8, Jan 24, 2024

Subprocesses

- `int execlp(const char *path, const char *arg, ...);` is a more convenient alternative to `execve()`
  - Instead of having to build an argument array we can use varargs to specify program arguments
  - Will also search `PATH` for the executable, so we don't have to specify the full path
- `int dup(int oldfd);` and `int dup2(int oldfd, int newfd);` duplicates a file descriptor and returns an independent file descriptor that refers to the same file
  - The `oldfd` and `newfd` file descriptors will refer to the same thing after the call
  - `dup()` will return the lowest file descriptor
  - `dup2()` will close the `newfd` file descriptor and then make that file descriptor refer to the same thing as `oldfd`
    - * This can guarantee that we get the exact file descriptor number that we want
  - This is an atomic system call (can't be interrupted)
  - Note: Closing the original file descriptor does not close the new returned file descriptor! (i.e. we need to close the file descriptor returned by `dup()` separately)
- Our goal is to create a new process with a specified executable name, and be able to send to the process' `stdin` and receive any data it writes to `stdout`
  - We `fork()` and call `execlp()` in the child to start the process
  - To get the child's output, `pipe()` before forking, and then call `dup2()` to replace the child's `stdout` file descriptor with the write end of the pipe; now in the parent process we can read from the pipe to get the child's output
  - Similarly to send data to the child, replace the child's `stdin` with the read end of a pipe