# CS 273 (Operating Systems) Strategy for developing a shell

As with all programming projects, the recommended approach for creating your shell is to *develop the program incrementally in stages.* Here is a suggested strategy:

1. Complete the parsing laboratory assignment, giving you a program that reads a line of input and produces a `struct` containing an array `tok` of the string "tokens" that appear in that input line.
2. Add a call to `execve()` at the end of your parsing program that attempts to execute the first token in an input line as a path to an executable file. You can add a 0 pointer just after the last token in your `tok` array, then use that `tok` array for the "`argv`" argument of `execve`. The resulting program should carry out your parsing steps, then perform the program being "`exec`ed."

   Enter the absolute path of an executable file as the first word of your input line, followed by arguments for that executable. You can find absolute paths of executables using the program `which`. For example, the Linux command `% which ls` prints the absolute path `/bin/ls`.
3. Add a call to `fork()` to your program, and have the child process (only) perform your `execve()` call. Add `printf()` calls to verify that the parent and child process are behaving correctly. See `~cs273/egs/forkeg.c` for an example of `fork()`.
4. Have your parent process call `wait()`, and verify that the parent does wait until the child has finished using `printf()` calls.
5. Now enclose your code for reading a line, parsing it, and `fork()/exec()/wait()` in a loop, so you can repeatedly read program names and arguments then perform those commands. This essentially satisfies the basic assignment.