Princeton University
COS 217: Introduction to Programming Systems
The Shell Assignment: Development Stages

Stage 0: Preliminaries

Learn the overall structure of ish and the pertinent background information.

Study the assignment specification and the assignment supplement. Review the lecture slides and precept material from the first half of the course on testing, building, debugging, style, and especially modularity. Study the lecture slides and precept material from the second half of the course on exceptions and processes, process management, I/O management, signals, and alarms. Complete the pertinent required reading, especially Chapter 8 of Computer Systems: A Programmer’s Perspective (Bryant & O’Hallaron).

Stage 1: Lexical Analysis

Compose a lexical analyzer module whose input is a sequence of characters from a character array and whose output is a token array.

Compose a top-level client named ishlex.c. Use ishlex.c, your lexical analyzer module, and any additional modules that you have composed to build a program named ishlex. ishlex must read a line from stdin, write the line to stdout, pass the line to your lexical analyzer module, accept the token array that your lexical analyzer module generates, write the tokens to stdout, and repeat until EOF (simulated by Ctrl-d).

Test your ishlex program (and thus your lexical analyzer module) thoroughly by comparing its behavior with that of the given sampleishlex program.

Stage 2: Syntactic Analysis (alias Parsing)

Compose a syntactic analyzer module whose input is a token array and whose output is a command.

Compose a top-level client named ishsyn.c. Use ishsyn.c, your lexical analyzer module, your syntactic analyzer module, and any additional modules that you have composed to build a program named ishsyn. ishsyn must read a line from stdin, write the line to stdout, pass the line to your lexical analyzer module, accept the token array that your lexical analyzer module generates, pass the token array to your syntactic analyzer module, accept the command that your syntactic analyzer module generates, write the command to stdout, and repeat until EOF (simulated by Ctrl-d).

Test your ishsyn program (and thus your syntactic analyzer module) thoroughly by comparing its behavior with that of the given sampleishsyn program.

Stage 3: External Commands

Compose a top-level client named ish.c. Use ish.c, your lexical analyzer module, your syntactic analyzer module, and any additional modules that you have composed to build a program named ish. Your ish should execute simple external commands. That is, your ish should assume that neither stdin nor stdout is redirected. Use the fork(), execvp(), and wait() system-level functions.
Your **ish** must read a line from **stdin**, write the line to **stdout**, pass the line to your lexical analyzer module, accept the token array that your lexical analyzer module generates, pass the token array to your syntactic analyzer module, accept the command that your syntactic analyzer module generates, execute the command, and repeat until **EOF** (simulated by Ctrl-d).

Test your **ish** program thoroughly by comparing its behavior when executing simple external commands with that of the given **sampleish** program.

**Stage 4: Shell Built-In Commands**

Enhance your **ish** program so it can execute the shell built-in commands **exit**, **cd**, **setenv**, **unsetenv**.

Test your **ish** program thoroughly by comparing its behavior when handling shell built-in commands with that of the given **sampleish** program. Specifically, test the program's handling of the **cd** shell built-in command by executing it and the **pwd** and **ls** external commands. Test the program's handling of the **setenv** and **unsetenv** shell built-in commands by executing them and the **printenv** external command. Test the program's handling of the **exit** shell built-in command by executing it.

**Stage 5: I/O Redirection**

Enhance your **ish** program so it can execute external commands that redirect **stdin** and/or **stdout**. Use the **creat()**, **open()**, **close()**, and **dup()** or **dup2()** system-level functions.

Test your **ish** program thoroughly by comparing its behavior when handling external commands that contain redirection with that of the given **sampleish** program.

**Stage 6: Signal Handling**

The challenge part.