Princeton University
COS 217: Introduction to Programming Systems
Ish: Development Stages

Stage 0: Preliminaries

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

Study the assignment specification and the assignment supplement. Study the lecture notes on system calls, processes, and signals. Study literature on Unix system calls, processes, and signals. Chapter 8 of the book *Computer Systems: A Programmer's Perspective* (Bryant & O'Hallaron) is appropriate.

Decide, at least tentatively, on the key modules in your program.

Stage 1: Lexical Analysis

Create the lexical analysis phase of *ish*. That is, create a lexical analyzer whose input is a sequence of characters from a **character array** and whose output is a **token array**.

Write the high-level code that calls your lexical analyzer. The code should first read lines from the .ishrc that resides in your HOME directory until it reaches EOF. (It should print each line that it reads from .ishrc immediately after reading it.) Then the code should read lines from stdin until it reaches EOF (simulated by Ctrl-d).

Testing: Create temporary code that prints the token array that your lexical analyzer produces for each input line.

Stage 2: Syntactic Analysis (alias Parsing)

Create the syntactic analysis phase of *ish*. That is, create a parser whose input is a **token array** and whose output is a **command**.

Write the high-level code that calls your parser. The code should pass the token array (created by your lexical analyzer) to your parser.

Testing: Create temporary code that prints the command that your parser produces for each input line.

Stage 3: Executable Binary Commands

Create an initial version of the execution phase of *ish* so it can execute executable binary commands. For now, assume that neither stdin nor stdout is redirected. Use the **fork()** and **execvp()** system calls.

Write the high-level code that calls your executable binary command execution code.

Testing: Use *ish* to execute numerous executable binary commands (**cat**, **more**, etc.) with and without arguments.
Stage 4: Shell Built-In Commands

Enhance the execution phase of ish. Specifically, create code that executes the built-in commands exit, cd, setenv, unsetenv.

Testing: Test the cd built-in command by executing it and the pwd and ls executable binary commands. Test the setenv and unsetenv built-in commands by executing them and the printenv executable binary command. Execute the exit command.

Stage 5: I/O Redirection

Enhance the execution phase of ish so it can execute executable binary commands that redirect stdin and/or stdout. Use the creat(), open(), close(), and dup() or dup2() system calls.

Testing: Repeat the tests for previous stages, adding I/O redirection.

Stage 6: Signal Handling

On your own!

Copyright © 2009 by Robert M. Dondero, Jr.