C Sells C Shells on the C Shore

Yiğit Kağan Poyrazoğlu

December 10, 2023

1 Introduction

In this project, I was asked to implement a simple shell, named myshell, that provides a subset of functionalities of a real shell (namely bash or zsh). My implementation consists of three files, each for a different step of the REPL loop for the shell.

1.1 eval.c

This file handles tokenization and evaluation of the input string. The syntax as described in the description is simple, and it is mentioned multiple times that we bore ourselves not with the extreme cases; hence, the tokenizer and the evaluater uses simple, brute-force methods akin to more complex ones encountered in real life.

1.2 execution.c

This file handles path searching, fork-exec of child processes and stdout/stderr piping. dup2() is used to redirect to a given file, on which a new file descriptor is opened.

1.3 myshell.c

This is the main entry of the program. The 2 built-in functions (bello, alias) are implemented here. Here are also some other functionalities, such as signal handlers and backspace parsers.

2 Difficulties

One of the main difficulties I encountered were to mimic bash/zsh behaviour on SIGINT. While it is easy to ignore the signal, to provide the exact behavior i needed to realize that bash/zsh print their prompts to stderr, not stdout. I also needed to create 2 signal handlers based on where the signal is caught to make printing prettier and to capture exact behavior.
Another difficulty for me was the parsing of the input string, which is made easy only through the sentence "do not bore yourselves with extreme cases". I've tried using Regex, but things got too complicated; so I resorted to simple usages of strtok().
One final difficulty was overall memory management. Things got too complicated with dynamically allocated memory, so I've resorted to allocating memory statically at function caller and to pass the pointer to it into the function.