Garrett Bowser
CIS3207 Section 04
Project 2: myshell
Due date: 14 October 2019

Overview

Myshell is a small, semi-featured linux shell. Myshell is capable of creating child processes, input and output redirection, as well background/parallel processing. It uses linux system calls and includes linux-like built-in functions.

Data Structures

Shell

The Shell data structure represents the current state of the shell. It contains a vector of built-in commands called cmdList, the program's parsing and input functions, as well as some information about the state of the shell. Commands are received on standard input with getline() and are tokenized by whitespace into a vector of strings. The first entry in the vector is always considered the command unless the input was read from a file. The shell uses the cmdList to check whether or not the command is internal to the shell. If it is, it simply runs the command's call function with the rest of the user input as an argument. Otherwise, the shell uses the access() linux system call to determine if the command it was given was the name of an executable file in its path that it has permission to run. If so, the shell forks and runs the executable. See redirect tokens below for more information on how commands are parsed.

Command

The shell uses a Command data structure to handle built-in commands. Commands have a minimum argument count, a string vector of aliases, and a call function that is sent a list of arguments when the command is run. The shell checks each built-in Command's aliases before searching for an external executable.

Testing

Myshell was tested using the batch file run.ms that is provided with the shell executable. It contains several commands and is meant to illustrate the shell's basic functionality. This file was changed throughout the development process and now reflects most of the shell's primary functions. Note that background execution tokens are included at the end of the run.ms testing batch file. Executing commands as background processes makes it possible that a command may print to the terminal window after the shell's prompt string prints to the screen, leaving the user with a blank line as a prompt. This may create a confusing user experience.

Redirect tokens

Myshell relies on redirect tokens (<,>,>>,|) and other command line tokens (&) to give the user more control over the way their commands are handled by the shell. Redirect tokens allow the user to redirect command inputs and outputs to files (or other commands in the case of the | (pipe) token). The < (input redirection) token redirects input from the interactive shell prompt to the file specified after the token. Myshell swaps the current stdin file descriptor with the file descriptor of the file specified and runs each subsequent line in the file as an argument to the command provided when the redirect was first invoked. When this finishes, it returns the file

descriptor to where it was prior to the redirect. In the case that input was redirected by a batch file, the input buffer must be cleared prior to redirecting the input to the specified file, and upon reaching the end of the file, the batch file must be reopened.

The > and >> (output redirection) tokens are used to redirect the output of a command to a file. The > token truncates (clears) files before writing to them whereas the >> token appends to them. Both tokens will create a file if the specified file does not exist. The output redirection tokens replace the current stdout file descriptor with the file descriptor of the file specified after the token. Input and output tokens can be used in conjunction with one-another to read input from a file, and send output from processing that input to another file.

The | (pipe) token allows commands to communicate with one-another directly. The command before the pipe sends its output to the pipe, and the command after the pipe receives it as input. To accomplish this, the program is forked twice. The first child, the command before the pipe, sends its output to the pipe and the shell waits for it to exit. The second child receives its input from the pipe and the shell waits for it to exit, after which its return is sent to the shell's standard input. Piping and the standard redirect tokens cannot be used in conjunction with one-another, and only one pipe may be used per line of command input.

The & (ampersand) token tells the shell that it is to run the preceding command in the background, which is to say that the shell is not to wait for the process that is running the command to exit before continuing. This allows the user to specify multiple commands for the shell to run in parallel with one-another. Multiple commands can be invoked as background processes on a single command line but their output can only be redirected once.