Myshell Report

How to Run

To compile, write `make` and to run write `./myshell`

Algorithm

Firstly, the myshell tokenizes a given command, forming a linked list of tokens. The token types are defined in the “token.h” file. In cases of missing quotes or more than three consecutive ‘>’ symbols, the token type becomes TOKEN_ERROR. Inside of quotes completely generate a single TOKEN_ARG type token. There is also a depth parameter in tokenization to detect commands. The depth of a command starts at zero and is incremented by one in each tokenization step. Additionally, after encountering the '&' symbol, the depth is reset to zero. Each command is checked for an alias. If an alias is found, a recursive call of the tokenization function tokenizes the alias and replaces the command with alias tokens. After alias tokenization, the next tokens are connected to the tail of the linked list. Aliases of aliases are prevented by the is_alias parameter.

After tokenization, parsing of the command line follows. Next token after ‘>’ is the redirect file, after ‘>>’ is the append file and after ‘>>>’ is the reverse append file. ‘&’ makes the command background task. The other TOKEN_ARG type tokens are arguments for command. Command struct also forms a linked list split by ‘&’ symbols. If a command is a background command, it runs in background then the next linked command is executed.

The execution of a command proceeds as follows: If the command is 'exit,' the program terminates. 'Bello' and 'alias' execute the corresponding executables in the './bin/' folder. Other commands are searched in the PATH. If an executable is found, the first one is executed. Otherwise, 'Command not found' is printed. If there is a TOKEN_ERROR, multiple different redirections, or no redirection file after the redirection symbol, 'Invalid argument' is printed.

If everything is valid then the forking process starts. To handle reverse output redirection in a background process, the fork() is used twice. The first fork() for background management. If the command has ‘&’, the parent process does not wait for the child process. The child process then uses another fork() to manage output redirection. The grandchild process changes standard output in case of redirection, then calls exec() to run the given command. The child process waits for the termination of the grandchild process. After termination, output gets reversed and printed .reverse temporary file if necessary. When the child process is finished, the parent process continues to the next command if the child process is not a background process that has not been already waited.

The .history and .alias files are kept in the source directory. Commands are appended to the .history file after the execution. Besides, alias command history is kept in .alias file.

Difficulties

The splicing of the alias linked list and next argument’s linked list made handling aliases easier. Besides, the trickiest part for this project is handling reverse append in the background process, which is managed by two forking in the project.

Author

- Atakan Yaşar