Assignment 1, P1: Custom Shell in C
Design Document

Dhruv Rawat & Anirudh Singh

October 29, 2022

1 Problem description

For this problem, we had to implement a command line interpreter or shell. The shell was supposed to take in a command from user at its prompt, execute it, and then prompt for more input from the user. In addition to this, shell was to support piping as well as input/output redirection. Support for two new pipeline operators,|| and ||| was to be provided as well.

2 How to run

```
$ gcc shell.c
$ ./a.out
```

3 Features

Note: Separate different commands, arguments with a whitespace. Avoid whitespace before double or triple pipes.

3.1 IO Redirection

The shell supports input and output redirection. For output redirection, the symbol > needs to be used followed by the filename. If the output needs to be appended, use >>. Similarly for input redirection, < needs to be used. If the file to which the output needs to be redirected or appended does not exist, a new file will be created with the given name. In the file from which the input needs to be redirection does not exist, the shell throws an error.

3.2 Pipelining

The shell supports three types of pipelining operators - |, || and |||. The shell provides support for multiple single pipelines (|), which can be nested inside || or ||| commands. A sample command could be -

```
$ ls -l || grep printf | wc > count.txt, wc
```

Only one double or a triple pipe can be used in the input command.

3.3 Built-in commands

The shell provides support for a number of builtin commands.

1. **cd** $< path >$: Allows the user to change directory specified in $< path >$.
2. **help**: Lists the authors and all builtin commands available
3. **exit, logout, quit**: Exit the shell

4. **clear**: Clear the screen
5. **type** $< cmd >$: Tells whether a command $< cmd >$ is a builtin or has a PATH
6. **history, !** $< num >$: Shows a table of all commands entered. Also allows to execute those commands via corresponding index number $< num >$.

3.4 Background processes

The shell supports execution of processes in the background. The user just needs to end the input command with & symbol.

3.5 Multiple commands together

The shell also supports execution of multiple commands entered in same input. The user needs to separate the commands entered by semi-colon, ";".

4 Syntax

⟨MULTI_COMMANDS⟩ ⊨ ⟨FULL_COMMAND⟩ | ⟨FULL_COMMAND⟩ ; ⟨MULTI_COMMANDS⟩
⟨FULL_COMMAND⟩ ⊨ ⟨FULL_COMMAND⟩ `&`
⟨FULL_COMMAND⟩ ⊨ ⟨CMD_LIST⟩ `TPL_PIPE` ⟨CMD_LIST⟩,⟨CMD_LIST⟩,⟨CMD_LIST⟩
⟨FULL_COMMAND⟩ ⊨ ⟨CMD_LIST⟩ `DBL_PIPE` ⟨CMD_LIST⟩,⟨CMD_LIST⟩
⟨FULL_COMMAND⟩ ⊨ ⟨CMD_LIST⟩ `SGL_PIPE` ⟨CMD_LIST⟩
⟨CMD_LIST⟩ ⊨ ⟨COMMAND⟩ | ⟨COMMAND⟩ `SGL_PIPE` ⟨CMD_LIST⟩
⟨COMMAND⟩ ⊨ ⟨ATOM_CMD⟩ `>` *output* | ⟨ATOM_CMD⟩ `>>` *output*
⟨COMMAND⟩ ⊨ ⟨ATOM_CMD⟩ | ⟨ATOM_CMD⟩ `<` *input*
⟨ATOM_CMD⟩ ⊨ *command* | *command* `ARGS`

Note: SGL_PIPE refers to |, DBL_PIPE refers to || and TPL_PIPE refers to |||.

5 References

1. The Linux Programming Interface by Michael Kerrisk
2. Advanced Programming in the UNIX Environment by W. Richard Stevens and Stephen A. Rago
3. Tutorial - Write a Shell in C: Stephen Brennan
4. Bash Colors
5. Lecture slides and class notes of IS F462: Network Programming