Test Suite: Mini UNIX Shell

Student Name: Jaskaran Singh
Date: December 9, 2025
System Environment: macOS (it is a Unix-based operating system)

Overview

This document demonstrates the functional verification of the mini unix shell. Each test case targets a specific functional requirement..

Test Environment Setup

- **Compilation command used:** make
- **Execution command used:** ./shell

```
(base) ➜  Cornerstone-Project-col7001 git:(main) make
clang  -Wall -Wextra -g shell.c -o shell
(base) ➜  Cornerstone-Project-col7001 git:(main) ✗ ./shell
myshell>
```

1. Basic Command Execution

Objective: To verify the shell can find and then execute external programs using the system PATH.

- **Command:** ls
- **Expected Result:** The shell will list the files in the current directory and return the control back to myshell>.

Screenshot:

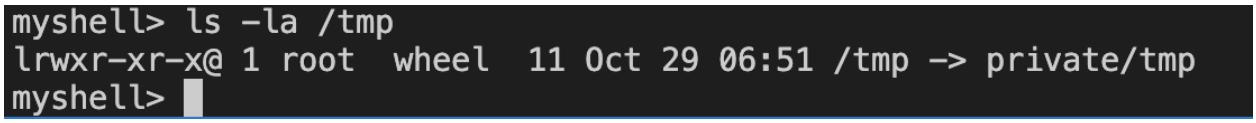
```
(base) ➜  Cornerstone-Project-col7001 git:(main) ✗ ./shell
myshell> ls
I_O_redirection_test.txt        README.md                       shell.dSYM
Lab 1                           shell
Makefile                        shell.c
myshell>
```

2. Argument Parsing

Objective: To verify the shell correctly parses command-line flags and arguments.

- **Command:** ls -la /tmp
- **Expected Result:** The shell will display a detailed list of files in the /tmp directory, including hidden files.

Screenshot:

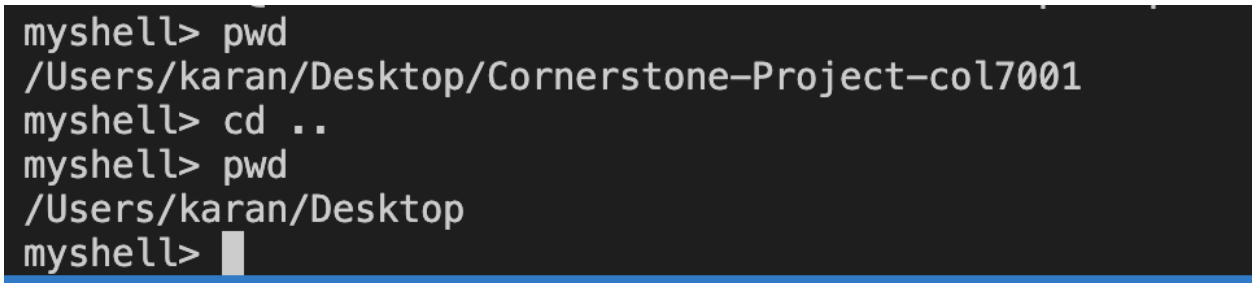


3. Built-in Commands (cd)

Objective: To verify the cd command changes the shell's current working directory (process state) rather than forking a new process.

- **Command Sequence:**
  1. pwd (Check start location)
  2. cd .. (Move up one level)
  3. pwd (Verify new location)
- **Expected Result:** The second pwd output will show the parent directory.

Screenshot:

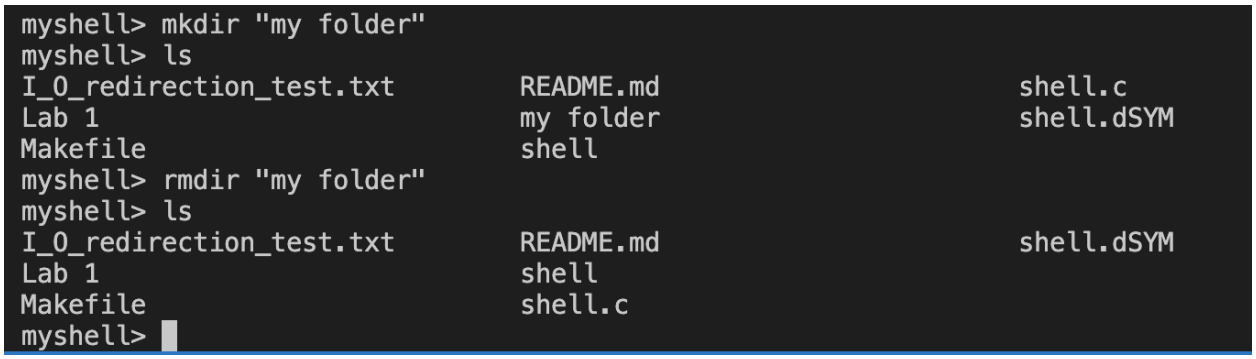


4. Quoted String Support

Objective: To verify that the custom parser handles spaces inside double quotes as a single argument.

- **Command:** mkdir “my folder”
- **Expected Result:** Only a single folder named “my folder” is created and not 2 different folders named ‘ “my ‘ and ‘ folder” ‘

Screenshot:

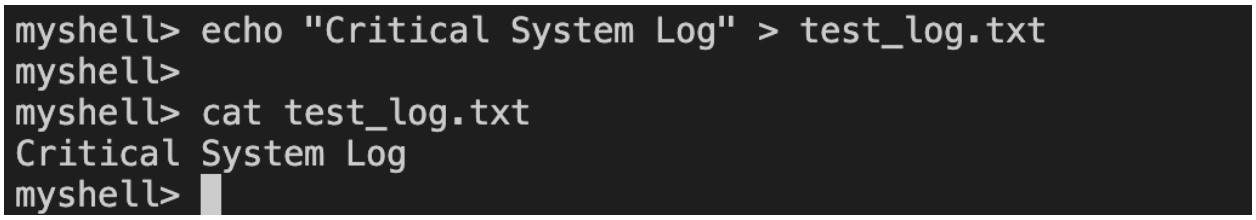


5. Output Redirection (>)

Objective: To verify that the shell can redirect standard output to a file.

- **Command Sequence:**
  1. echo "Critical System Log" > test_log.txt
  2. cat test_log.txt
- **Expected Result:** The echo command will print nothing to the screen. The cat command shows the content "Critical System Log" inside the file.

Screenshot:

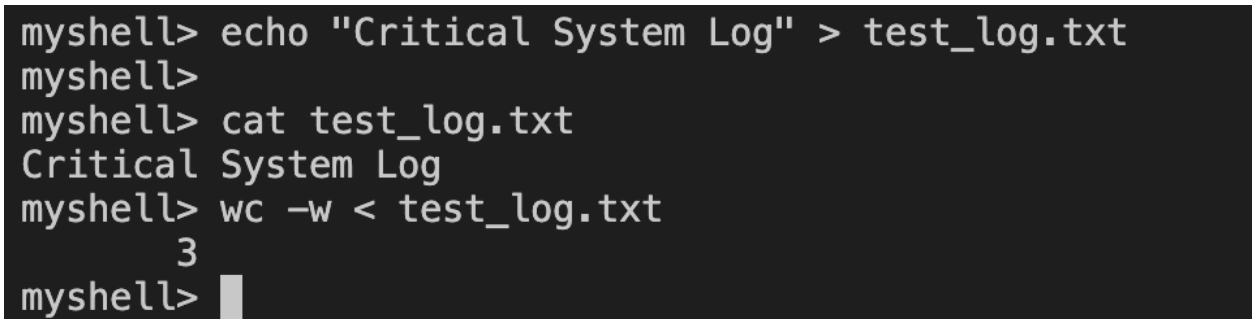


6. Input Redirection (<)

Objective: To verify that the shell can read standard input from a file.

- **Command:** wc -w < test_log.txt
- **Expected Result:** The wc (word count) command will display the number of words in the file (should be 3 if using the file from Test 5).

Screenshot:

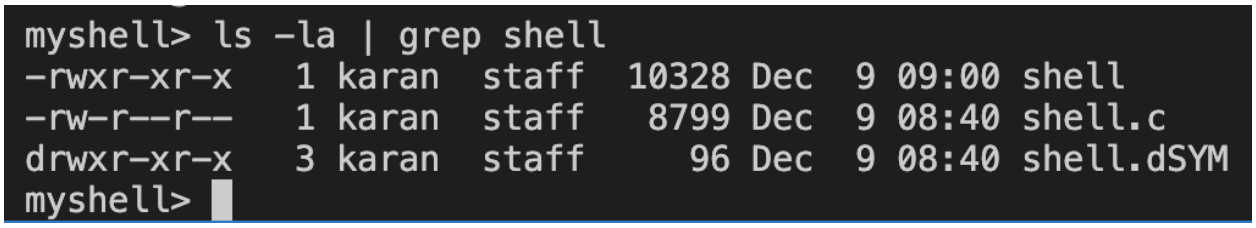


7. Pipeline Support (|)

Objective: To verify that standard output of one command flows into standard input of the next.

- **Command:** ls -la | grep shell
- **Expected Result:** The output should only show the files containing the word "shell" (e.g., shell.c, shell).

Screenshot:



8. Background Execution (&)

Objective: To verify the shell runs commands asynchronously and returns the prompt immediately.

- **Command:** sleep 10000 &
- **Expected Result:** The shell prints [Started process <ID of the process>] and immediately shows the myshell> prompt. The shell is usable while sleep runs silently. We tried using ps command to list all processes while sleep process was still running.

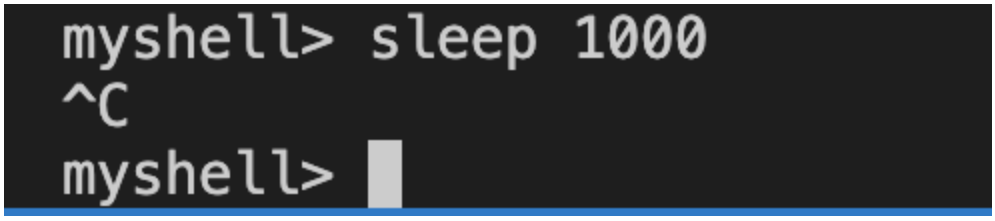
Screenshot:

9. Signal Handling (Ctrl-C)

Objective: To verify the SIGINT interrupts the foreground process but **not** the shell.

- **Command Sequence:**
  1. Run sleep 1000
  2. Press Ctrl-C
- **Expected Result:** The sleep command terminates immediately. The shell prints a new line and a fresh myshell> prompt.

Screenshot:

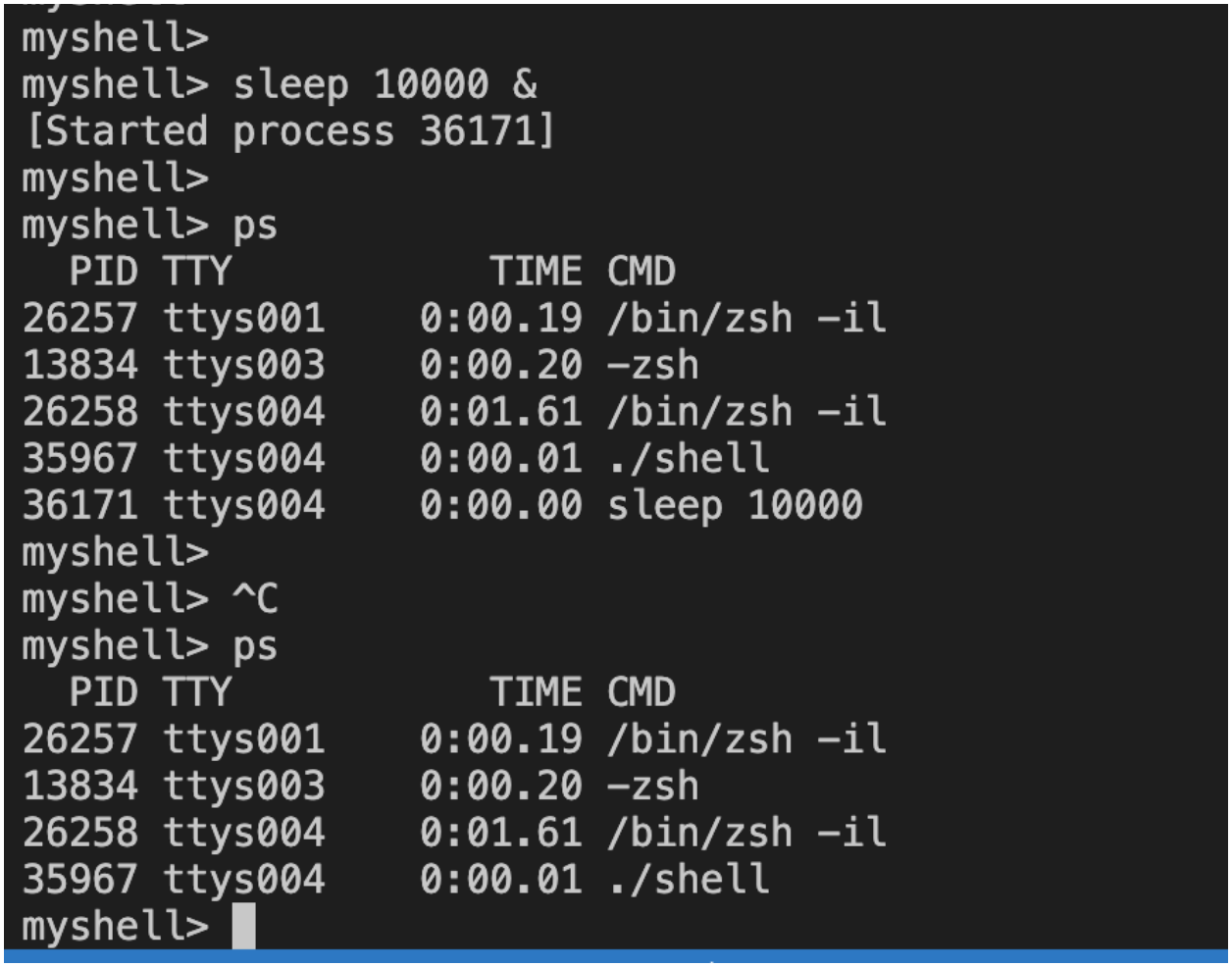


10. Signal Handling (Ctrl-C) with background process

Objective: To verify the SIGINT interrupts the background process but **not** the shell.

- **Command Sequence:**
  3. Run sleep 10000 &
  4. Press Ctrl-C
- **Expected Result:** The sleep command terminates the background sleep process. Running the "ps" command before and after the Ctrl+c shows the sleep process with id 36171 existing and then not existing. Hence we can conclude ctrl+C command killed the process.

Screenshot:

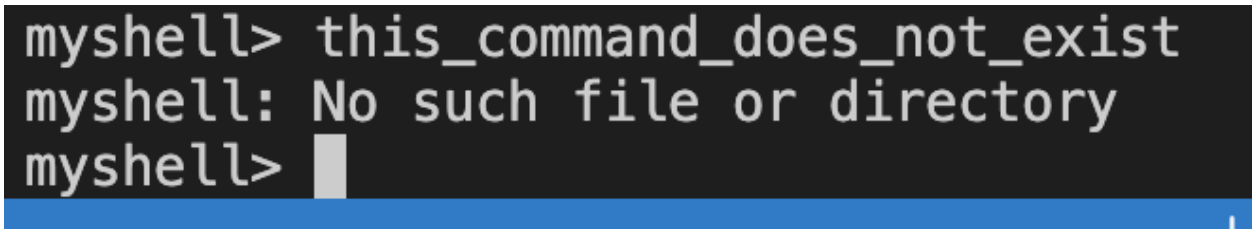


11. Robustness & Error Handling

Objective: To verify that the shell handles invalid commands gracefully without crashing.

- **Command:** this_command_does_not_exist
- **Expected Result:** The shell prints an error message (e.g., myshell: No such file or directory) and returns to the prompt cleanly.

Screenshot:



<---------- END OF REPORT ---------->