Kaitlyn Ellis

Dr Burge

CSCI 401

25 October 2023

The project's main objective is to develop a command shell that emulates the behavior of well-known shells like sh, bash, csh, and tcsh. The project is divided into stages, allowing for a structured approach to building the shell. Each step addresses specific functionalities and features with the ultimate goal of creating a fully functional and user-friendly shell. This paper will explore the design choices made during development, focusing on the stages outlined in the project instructions.

The design of the command shell encompasses various components and functionalities, which were carefully considered to create a coherent and user-friendly interface. First, the user interface prominently features a prompt displaying the current working directory. This design choice enhances the user experience by helping users keep track of their location within the file system.

To process user input effectively, the code includes tokenization. This process splits the user's command into individual tokens, simplifying parsing and execution. Moreover, the shell incorporates built-in commands to streamline essential operations such as changing directories, displaying the current working directory, and setting environment variables.

Process forking is implemented to execute non-built-in commands. When users provide a command that is not built-in, a child process is created to execute the command, ensuring the stability of the shell. The shell also supports background processes, recognizing the "&" symbol at the end of a command line, enabling users to run commands in the background while continuing to interact with the shell.

Signal handling, particularly SIGINT (ctrl-C) management, is critical for user convenience. When users press ctrl-C, the shell returns to the prompt instead of quitting, ensuring a responsive user experience. Additionally, error handling is vital to the design, enhancing the shell's robustness. For instance, the chdir function includes error checking to manage directory change failures.

Effective documentation plays a significant role in understanding and maintaining the code. The code is extensively commented on to describe the purpose and functionality of each section. These comments clarify the program's flow and the implementation of specific functionalities. Furthermore, functions are documented to explain their parameters, return values, and specific tasks. This documentation assists developers in using and modifying functions as needed.

To facilitate user adoption and help test the code's functionality, the code is complemented with usage examples. These examples illustrate how to use the shell and its built-in commands. They serve as practical guides for users exploring the shell's capabilities and features.

In conclusion, this project aims to create a functional command shell that mirrors the design of widely used shells while providing a structured and user-friendly experience. The design choices, project stages, and comprehensive documentation contribute to a well-structured and understandable shell program. The stages of development address specific functionalities, each carefully designed to make the shell powerful and user-friendly.