Devin Griffin
Dr. Burge
Operating Systems
10/16/23

This report delves into the architecture decisions and accompanying documentation of a basic shell program constructed in the C language. The shell, an integral component of Unix-based operating systems, provides a user interface to interact with the system and execute a variety of commands. The code under review offers a rudimentary shell implementation with the ability to execute commands, manage foreground and background processes, and includes some embedded commands. The code is organized into several unique sections, each with a designated function. It begins with the inclusion of relevant libraries, constant definitions, and global variables declaration. Following this, it defines two signal handlers for Ctrl-C and timer expiration signals. The core of the program lies in the main function, which encompasses the primary execution flow. This section is subdivided into four significant parts: initialization, input handling, built-in commands, and command execution. The shell commences by initializing specific variables and registering the Ctrl-C signal handler, ensuring it can appropriately respond to user input and handle signals. The program's essence is user interaction; the shell presents a command prompt inclusive of the current working directory, reads user input, and tokenizes the command line input. It also checks for EOF (Ctrl+D), facilitating a graceful shell exit for the user.

The shell's distinct feature is its ability to execute built-in commands like cd, pwd, echo, exit, env, and setenv. These commands are identified by the input line's first token and are executed directly within the shell, bypassing new process creation. The shell's primary function is to execute user-supplied commands. When a command is neither built-in nor explicitly set to

run in the background (using &), the shell generates a child process and runs the command within it. If a command is set to run in the background, the shell manages it by redirecting standard input, output, and error to /dev/null, preventing terminal interference. The program handles signals, notably SIGINT (Ctrl-C) and SIGALRM (for background processes). The SIGINT signal handler, sigint_handler, allows the user to interrupt a running foreground process gracefully, while the SIGALRM handler, sigalrm_handler, manages background processes exceeding a specified time limit. This architecture decision enhances user control over processes via signals, improving user experience and safety. An important design decision is the capability to run commands in the background by appending & to a command. The shell isolates these background processes, ensuring they don't obstruct shell interactivity. The use of /dev/null for redirection enables background processes to run without terminal interference. Robust error handling mechanisms are integrated into the code to address potential errors, whether they stem from user input, system calls, or command execution. By providing meaningful error messages and taking appropriate actions, the shell's reliability is enhanced.

The inclusion of built-in commands directly in the shell's code is a design decision improving usability and efficiency. These commands conduct common operations without launching external processes, resulting in faster execution and improved resource management. The code includes comments and documentation to clarify each section's purpose, key design decisions, and the role of various functions and variables. Though the code offers a functional shell, it has room for improvement, such as better user input handling, more comprehensive error messages, and additional built-in commands. In conclusion, the simple shell code's design decisions and documentation demonstrate a foundational understanding of shell operation and user interaction. This code provides a base for a more feature-rich and user-friendly shell,

enhancing its capabilities and user experience. By refining existing design choices and expanding functionality, the code has the potential to morph into a robust and versatile shell.