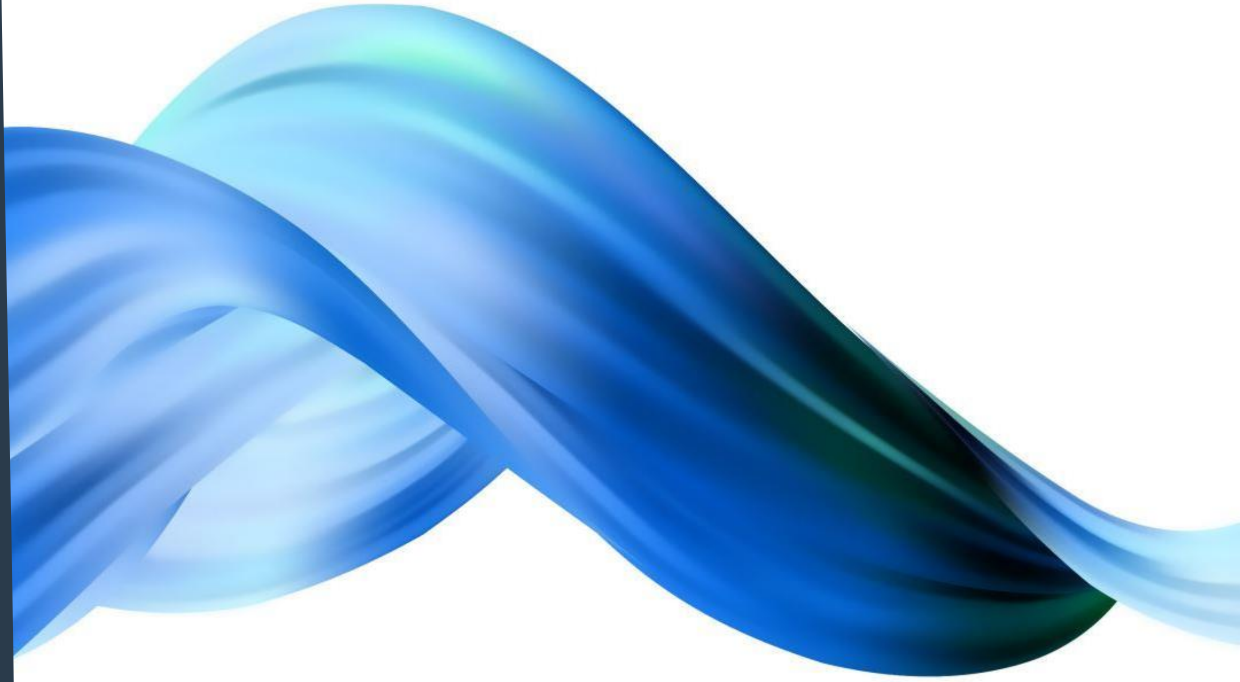
Linux Shell CSE x61 Operating Systems Lab 1

es-amin.mohamedamin2026@alexu.edu.eg

Amin Mohamed Amin El-sayed

21010310



Main Function

- The program sets up a signal handler using the signal function. Specifically, it associates the SIGCHLD signal with a function called on_child_exit.
- This signal is typically sent by a child process to its parent to indicate that it has terminated, allowing the parent process to clean up resources associated with the child process.
- After setting up the signal handler, the program calls the setup_environment function to navigate to the home directory, ensuring that the shell starts in a predictable location.
- Finally, the shell function is invoked, which initiates a loop to read and execute commands entered by the user in the terminal.

Write to log file function

- This function is responsible for logging information about child processes to a file. It opens the log file in append mode and writes information about the child process, including its process ID (pid), whether it exited normally or was terminated by a signal, and its exit status. It then closes the file.

Reap child zombie function

- This function is used to reap zombie child processes. It continuously calls waitpid with the p-id parameter set to -1, indicating that it should wait for any child process. It uses the WNOHANG option to check for child processes without blocking the parent process. When a child process is found, its information is logged using the write_to_log_file function.

On child exit function

- This function serves as a signal handler for the parent process. It is invoked when a SIGCHLD signal is received, indicating that a child process has terminated. It calls the reap_child_zombie function to handle the terminated child process.

Set-up environment function

- This function sets up the environment for the shell by changing the current working directory to the home directory of the user. It achieves this by using the chdir function with the path obtained from the HOME environment variable

Shell Function

- This function, shell(), represents the **core functionality of a shell program**. Here's a breakdown of what it does:
- **Initialization:** It initializes variables including command_exit, which serves as a flag to indicate whether the shell should exit.
- **Command Execution Loop:** It enters a loop to continuously read and execute commands until the command_exit flag is set.
- **Reading and Parsing Commands:** It reads a command from the user input, removes any unnecessary spaces, and checks if the command is empty. If the command is intended to run in the background (denoted by '&'), it sets the background flag and removes the '&' character from the command.
- **Parsing and Identifying Command Type:** It parses the command into individual parameters and determines the type of command based on the first parameter. If the command is "exit", it sets command_exit flag and identifies it as an exit command. If it's a built-in shell command (like "cd", "echo", or "export"), it identifies it as a built-in command. Otherwise, it's identified as a normal command.

Shell Function

- **Execution:** Based on the identified command type, it executes the command accordingly. If it's a built-in shell command, it calls execute_shell_built_in function. If it's a normal command, it calls execute_command function with appropriate parameters. If it's an exit command, it breaks out of the loop.
- **Loop Continuation:** The loop continues until the command_exit flag is set, indicating that the user has entered the exit command.

Remove starting trailing spaces

- This function removes leading and trailing spaces from a string. It iterates through the string, advancing the pointer until it encounters a non-space character for the start, and then truncates the string at the last non-space character from the end.

Read Command Function

- This function prompts the user with the current working directory (cwd), followed by a command line indicator ($). It then reads the user input using f-gets and removes the newline character at the end.

Parse command Function

- This function parses the command line into individual parameters. It first checks for and replaces any variables using the replace_var function.
- Then, it tokenizes the command line using space (" ") as the delimiter, but it also distinguishes between built-in commands (cd, echo, export) and other commands.
- For built-in commands, it keeps the entire command line as a single parameter, while for others, it splits the command line into separate parameters.

CD Function

- cd: This function changes the current working directory. If no directory is provided, it changes to the user's home directory. If the provided path starts with '~', it expands it to the home directory. It then uses chdir to change the directory and handles errors if the operation fails.

Export Function

- This function sets environment variables. It extracts the name and value of the variable from the command string and uses setenv to set the environment variable. It also handles cases where the value is enclosed in double quotes.

ECHO Function

- This function prints the provided string to the console. It removes double quotes if they are present at the beginning and end of the string and then prints the string followed by a newline.

Execute Shell Built in Function

- This function determines which built-in command is being invoked and calls the corresponding function (cd, echo, export) to execute it.

Execute Command Function

- This function forks a new child process to execute a command. In the child process, it uses execvp to execute the command with the provided parameters.
- If the command is not found, it prints an error message.
- In the parent process, it waits for the child process to finish unless the command is intended to run in the background.

Execute Command Function

- This function forks a new child process to execute a command. In the child process, it uses execvp to execute the command with the provided parameters.
- If the command is not found, it prints an error message.
- In the parent process, it waits for the child process to finish unless the command is intended to run in the background.

TEST CASES (See GitHub Demo Videos)

https://github.com/WorldisAmen/Linux-Shell-OS