Operating Systems Lab
Project Based Learning

SIMPLE SHELL DEVELOPMENT

A PROJECT BY

AKSHAT KAUSHIK (1/23/SET/BCS/578)

YASH KATARIA (1/23/SET/BCS/566)

RITIKA SHARMA (1/23/SET/BCS/337)

Introduction

A shell is an interface between the user and the operating system, allowing users to execute commands and manage processes.

This project, "Simple Shell Development," aims to design and implement a basic command-line shell with fundamental functionalities like executing commands, handling input/output redirection, and managing processes.

Objectives

- To understand and implement process creation and execution in Linux.
- To develop a command-line interface that can interpret and execute user commands.
- To integrate features like input/output redirection and piping.
- To implement background process execution and signal handling.

Technologies used

Programming Language: C

Operating System: Linux (Ubuntu)

System Calls:

- fork() - Creates a new child process, allowing the shell to execute commands in a separate process.
- execvp() - Replaces the current process image with a new program, executing user commands.
- wait() - Makes the parent process wait until its child process completes execution
- dup2() – Redirects the file descriptors, essential for handling input and output redirection.
- pipe() – Enables inter – process communication

Features of the Simple Shell

- Basic Command Execution
- Built-in Commands
- Background Process Execution (&)
- Input & Output Redirection (< , >)
- Piping (|) for Command Chaining.
- Process Management

Desktop | WebIDE | Terminal | Web 8080 | +

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/wait.h>
#include <fcntl.h>

#define MAX_ARGS 100
#define MAX_INPUT 1024

// Function to execute single commands
void execute_command(char *args[], int background, char *input_file, char *output_f
    pid_t pid = fork();

    if (pid < 0) {
        perror("Fork failed");
    }
    else if (pid == 0) {  // Child process
        // Handle input redirection
        if (input_file) {
            int in_fd = open(input_file, O_RDONLY);
            if (in_fd < 0) {
                perror("Input file error");
                exit(1);
            }
            dup2(in_fd, STDIN_FILENO);
            close(in_fd);
        }

        // Handle output redirection
        if (output_file) {
            int out_fd = open(output_file, O_WRONLY | O_CREAT | O_TRUNC, 0644);
            if (out_fd < 0) {
```

Code Snippets

```
labex:project/ $ chmod 711 ./my_shell.c
labex:project/ $ gcc my_shell.c -o my_shell
labex:project/ $ ./my_shell
myshell> ls
my_shell  my_shell.c  my_shell.txt
myshell> mkdir ARY
myshell> ls
ARY  my_shell  my_shell.c  my_shell.txt
myshell> pwd
/home/labex/project
myshell> whoami
labex
myshell>
```

Live Demonstration of Project

Future Enhancements

Support for More Built-in Commands:

- Currently, the shell may support only basic commands.
- Future improvements can add built-in commands like history, alias, jobs, and fg/bg for better functionality.

Auto-completion & Command Suggestion:

- Implement **tab-based auto-completion** similar to Bash.
- Introduce command suggestions when users enter incorrect commands.

Improved Error Handling & Logging:

- Display user-friendly error messages for invalid commands.
- Maintain logs of executed commands for debugging.

Outcomes

A functional shell that can execute system commands and built-in commands.

Support for background execution and I/O redirection.

Enhanced understanding of operating system concepts like process management and system calls.

Thank You…

Visit our GitHub repository for source codes and all necessary files.

https://github.com/Ak-270704/Simple-Shell-Development