* Introduction

This project explores process creation and management in C using the fork() system call and execvp() function. The program demonstrates how a parent process spawns multiple child processes, each executing a distinct command. This provides hands-on experience with process control, inter-process communication (IPC), and system command execution.

* Implementation Summary

The program creates ten child processes using a loop that calls fork(). Each child process executes a predefined command using execvp(). The parent process waits for all child processes to finish using wait(), ensuring proper synchronization. The main components include:

The fork() function for process creation.

The execvp() function for executing system commands.

The wait() function for parent-child synchronization.

Error handling to manage potential failures.

* Results and Observations

A. Process Creation and Management:

Each child process is created using fork(), which duplicates the parent process.

The child process replaces its memory with a new program using execvp(), allowing it to execute a specific system command.

B. Parent-Child Interaction:

The parent process does not execute commands directly but instead spawns child processes for each command.

It tracks and waits for each child’s termination using wait(), ensuring orderly execution.

If a child process terminates normally, the parent retrieves and prints the exit status. If terminated by a signal, the parent reports the termination signal.

Conclusion

This project illustrates fundamental process control mechanisms in Linux-based systems. The use of fork(), execvp(), and wait() showcases efficient process management. The experiment demonstrates that each child process runs independently, while the parent ensures their completion. Understanding these concepts is crucial for developing efficient multi-process applications.