Introduction

Process management is an important concept in operating systems which involves the creation, execution and termination of a process. This C program demonstrates process creation and management using the fork() system call. The program spawns multiple child processes to execute a variety of Linux commands.

Implementation Summary

The program is structured around creating and managing multiple child processes. The key components of the implementation are process creation, task execution, error handling and parent process management. The fork() function creates ten child processes, each child process executes a different command using execlp, which replaces the child's execution context with the specified command. If the exec call fails an error message is displayed and the parent process waits for all child processes to complete using wait().

Results and Observations

A. Process Creation and Management

Each child process is created using fork() with a unique process id (PID) assigned. The execlp() function is used to execute system commands within child processes. If the function fails due to an invalid command, then an error message is printed and the program exits with a failure status. Meanwhile the parent process monitors the child process' execution.

B. Parent and Child Process Interaction

The parent process initiates child processes in a loop maintaining an array of PIDs. Each child process prints its PID before executing its command. The parent process collects exit statuses of child processes using wait() allowing it to detect normal termination or errors. The output displays a sequence of executed commands showing the process order.

Conclusion

In conclusion, the project successfully demonstrates process creation, execution, and termination in a Linux environment using fork() and execlp(). The relationship between parent and child processes demonstrates process management.