Lab 2 Report

Ruby Rosales

CS 470

February 8th, 2025

Table of Contents

[Introduction 3](#_Toc190078949)

[Implementation 4](#_Toc190078950)

[Results and Observations 5](#_Toc190078951)

[Conclusion 5](#_Toc190078952)

# Introduction

Lab two explored process creation in C. Processes are programs in execution. The lab required a parent process to be created which in turn creates ten child processes. The child processes are then required to perform unique tasks and report completion status. This report will discuss how the lab was implemented and the results/observations made after implementation.

# Implementation

The C program follows a simple structure that consists of the required header files and a main function. The required header files include the stdio, unistd, sys/wait, and stdlib header files. Within the main function, variables are declared to hold the process ID, process status, number of children processes, and commands. The commands variable stored the commands in an array structure and they include: echo(used to display lines of text), ls(used to list contents of directories), ps(used to display information about the processes currently being run), date(used to display current date and time), mkdir(used to create directories), pwd(used to display the path of the current working directory), find(used to search for files and directories), touch(used to create empty files or modify file timestamps), cat(used to display file contents), and whoami(used to display the username of the current user). The main function contains several print statements used to display process ID’s, errors that have occurred, and execution status. To create the ten child processes, the fork() function was used within a for loop that iterated ten times. Within the for loop, if and else if blocks are used to indicate if the fork has failed(fork() returns negative value) or have the child process execute a command from the array at index i using execvp(fork() returns 0). After creating child processes, the parent process waits for the child processes to complete by using wait(). The wait() function is used in the conditional of a while loop that prints the status of a child process and iterates until no more child processes exist(wait() returns -1).

# Results and Observations

Processes are created by the parent process when using fork(), this function creates a duplicate of the parent process. The fork() function returns a negative value to indicate an error has occurred with creating the new process, returns 0 to the newly created child process, or it returns a positive value to the parent process.

# Conclusion

This lab explored process creation and command execution through a C program. The lab saw the creation of multiple child processes through the use of the fork() function and error handling. These child processes were then able to execute basic linux commands by using the execvp() function.