This report summarizes the use of a Process Management Simulator in the C programming language, created on a Linux operating system. A process in programming is a program in execution, the ability for a program to create multiple processes allows the system to perform several tasks simultaneously. The program uses fork() and execvp(), and creates ten child processes.

The C program starts by using a for-loop to create 10 child processes, each represented with a pid. Each child process is being created by using the fork() system call. This splits the child processes from the parent process. The child process performs a specific task, then terminates. Tasks are executed using execvp() and an args[] array, which allows the child to perform the commands such as ‘echo’ and ‘ls’.

The for-loop executes from int i == 1 until i == 11, executing a total of 10 times for the 10 child processes. If fork() is successful, it continues to the next step. The parent process calls waitpid() to wait for each child process to complete. After a child process finishes, the parent checks the exit status to know whether the child exited normally or by signal. Afterwards, the parent reports how the child process was terminated. The program waits for each child process to finish before continuing, so that child processes are completed in order.

This project demonstrates fundamental concepts in process management, creating the ten child processes, executing commands, and managing the process termination. Fork() and execvp() are used to create the child processes and execute tasks, and waitpid() is used to ensure the parent process waits for each child to complete.