Programming Assignment No. 1 Report (CS307 - Operating Systems)

-Hagverdi Ibrahimli (30014)-

Introduction

The program simulates the pipe command in Linux, specifically emulating the command "*man ping | grep -A4 -- -A > output.txt"*. The program utilizes three processes, with the main process acting as the shell and two child processes handling the "man" and "grep" commands, respectively.

Command Explanation & Motivation

The *ping* command in Linux is used to send an Internet Control Message Protocol (ICMP) echo request to a specified network host or IP address to test the reachability and network latency. The flag *-A* of the *ping* command utilizes the concept of adaptive ping, meaning that the time span spent between packets adapts to round-trip-time. I chose this command and option because:

1. *ping* is a great tool to test the availability and latency of the target host.
2. *-A* flag adapts the interpacket interval to the round-trip-time (i.e. latency).

Also, there are some extra options provided in the command such as *-A4* and –. The first one, *-A4* is an extra option for the *grep* command to grab the next four lines from where it first found the searched option *-A*. The reason for us to select 4 lines after the *-A* is found is because the definition of the flag spans 5 lines in total. Consecutively, – (double dash) is used since our search input for the *grep* command *-A* uses the special character - which shell interprets as a special character and by using – shell is informed of the end of the command line options and treats the rest of the input as non-special characters. Lastly > *output.txt* is used to direct the output generated by the *grep* command to the text file *output.txt* instead of the console.

Process Hierarchy

The main process initiates two child processes. The first child process, created by the initial fork, executes the "man" command. The second child process, created by another fork, executes the "grep" command. Both the "man" and "grep" processes run concurrently and

independently, allowing the simulation of the pipe command in Linux. However, the main process waits for both child processes to complete before printing the completion message.

My program is 2a as for the report template. The parent-child relationship (i.e. main is the parent process of both man and grep processes) between the main-man and main-grep are evident as the main process creates both the "man" and "grep" child processes. My man and grep processes have a sibling relationship as they are both created by the same parent process, the main shell process. My man and grep processes can run concurrently. This is because the program creates two child processes using the *fork()* system call. The first child process executes the man command and writes its output to the write end of the pipe. The second child process executes the grep command and reads its input from the read end of the pipe. Since the two child processes are independent, they can run concurrently.

The program's execution flow demonstrates the parent-child and sibling relationships, ensuring the concurrency of the "man" and "grep" processes, which aligns with the specified guidelines.