CS307 *Programming Assignment 1* Report

Name: Muhammed Orhun Gale *Student ID:* 27654

Selected Command: ping

Selected Flag for Kill: -A

Command.txt: man ping | grep "\-A" -A 2 > output.txt

"ping" command is used for sending ICMP (Internet Control Message Protocol) ECHO_REQUEST to network hosts to get ECHO_RESPONSE from a host or gateway. "-A" flag stands for the adaptive ping which adapts the Interpacket interval to the RTT (round-trip time) so that a new ping request goes out as soon as the last reply is received. I picked this command and option because I am really curios about how computers communicate with the networks and other devices at low levels. I want to learn how to manage my devices' communication with networks.

Program Flow and Process Hierarchy

My program consists of 3 process' namely "Parent" (process 0), "Child" (process 1) and "Grandchild" (process 2). In order to provide inter-process communication, I used a Linux *pipe* via pipe() system call. I designed the process hierarchy of my program as follows:

1) ***Shell Process*** (Process-0)
   a) Main process is named as process-0, and it imitates the "Shell".

   b) Main process forks a child process (process 1).

   c) Waits until the Child process terminates.

2) ***"man" Command Process*** (Process-1)
   a) The process that is forked from the Parent is named as process-1 and it executes the "man ping" command.

   b) Child process forks a process namely Grandchild process (process-2)

   c) After it executes the given command, it gives the output of the command to the pipe's writing end (pipe_fd[1]) in order to transmit it to the process-2. Since it uses the pipe to write, pipe's reading end (pipe_fd[0]) is closed. Also, to use the pipe's writing end as the output, the standard output (STDOUT) of this process is closed too.

3) "grep" Command Process (Process-2)

a) The process that is forked from the process-1 is named as process-2 and it executes the "grep "\-A" -A 2 > output.txt" command.

b) In this process, pipe's writing (pipe_fd[1]) is closed, therefore it waits for an input to read from the pipe. In order to execute the "grep" command with the "man" command that comes from the process-1, standard input (STDIN) of this process is closed and pipe's reading end (pipe_fd[0]) is assigned instead.

c) In order to writing the executed command's output to the "output.txt", standard output (STDOUT) of this process' is closed and the file number of the output.txt is assigned instead.

Observation: Even though my program always gives the same output with the command execution in the terminal, size of the terminal, which the command executed, and the code compiled, affects the output. Output adapts to the size of the terminal and therefore it takes the lines according to the size. However, as I stated before, my output is always same as the terminal's output.