Linux Terminal Simulator

Limiting your abilities, one signal at a time.

TYLER TREPANIER
A00850517

Linux Terminal – The Terminal Simulator

The Linux Terminal is a program designed to operate similarly to the standard Linux Terminal prevalently in all Unix-based machines. Designed to limit your original abilities given to you by the original terminal, all signals (really, it's most of the signals. Refer to the technical report on which signal still penetrate this application) will be ignored and redirected to a signal handler that performs no actual tasks.

The Linux Terminal performs serveral functions (aside from being really limiting) these tasks:

1. *Disable all (most) signals.*
2. *Disable standard terminal commands.*
3. *Replace "redirects" terminal functionality to another key aside from the standard keyset. Note: All "special" characters are not displayed in the formatted message.*
    a. *'E' behaves as the Enter key*
    b. *'T' Terminates the program gently.*
    c. *'X' Acts like the backspace and erases the previous character from the output.*
    d. *'ctrl-k' IMMEDIATELY kills the program and all child processes.*
4. *Uses multiple processes that communicate with each other (to handle your anger and frustration while providing very little remedy).*
5. *Displays your original message (except for 'ctrl-k') and formats that message. This will eliminate all of the special characters that have defined functions.*

Why do we need another terminal application if the original terminal works perfectly fine and has fully matured?

Because we can.

Process Implementation

INPUT → TRANSLATE → OUTPUT

Make a note that the input does not have access to the output directly, having that functionality defeats the whole purpose of this program.

Main

This is the main entry point into the application.

1. Redirect all signal calls to the Sig_Handler function (Note: this simply ignores signal calls)
2. Eliminate most of the standard terminal functionality
    - *system ("/bin/stty raw igncr -echo")*
        - *Do not allow echoing and ignores some basic functions*
2. Open translation communication pipe for reading and writing
    - Failure: Fatal error, close program.
3. Open output communication pipe for reading and writing
    - Failure: Fatal error, close program.
4. Fork two processes and assign each process a task

NOTE: The assigned tasks are part of a loop, only normal termination will allow each process to proceed to next step!!!

    - PARENT: Assign **ProcessInput** task (Handles keyboard input)
    - CHILD: Assign **ProcessTranslate** task (Translates input)
    - ERROR: Fatal error, close the program.
5. Restore all signals and terminal functionality.
6. Close the Program.

ProcessInput

Assigned to the main program process, this function carries out received input from the keyboard inside of an endless while loop. There are several special characters that perform special actions that are defined in this pseudocode.

1. Close the ability for reading with the translate pipe.
2. BEGIN THE ENDLESS LOOP
    a. Acquire character from keyboard.
    b. Compare character to list of special characters
        i. ‘E’:
            - Send the raw message to the input pipe to be translated.
            - RESTART LOOP
        ii. ‘T’:
            - Empty the raw message buffer into the translate pipe
            - Wait for the ProcessTranslate to finish.
            - Restore terminal functionality
            - LEAVE THE ENDLESS LOOP
        iii. ‘ctrl-k’:
            - Stop everything,
            - restore terminal functions
            - KILL this process and its children (you monster).
            - KILL PROGRAM.
        iv. Nothing special:
            - Add the character into the raw message. RESTART LOOP
    c. END OF LOOP
3. Quit program.

ProcessTranslate

This is the task assigned to the child process of the main program process (aka ProcessInput). This performs the task of receiving raw input, translating the raw input and sending both the raw and formatted messages to the ProcessOutput to be displayed onto the screen. Pressing 'ctrl-k' from the Process Input will IMMEDIATELY suspend all actions and terminate this process.

1. Disable writing to the input pipe (only reading required).
2. Create output pipe for communication with the ProcessOutput task.
3. Fork two processes and assign each process a task

   NOTE: The child begins its loop inside of its own function, while only the ProcessTranslate continues on to the next step!

   a. PARENT: Begin the next step of the endless loop.
   b. CHILD: Assign **ProcessOutput** task (Outputs received messages)
   c. ERROR: Fatal error, close the program.
4. BEGIN ENDLESS LOOP
   a. Read raw message from input pipe
      i. Failure: RESTART THE ENDLESS LOOP
   b. Send the raw message to the output to be displayed.
   c. Translate the raw input:
      i. 'a': convert into 'z', place into formatted message
      ii. 'X': erase the last character from the formatted message
      iii. 'K': erase all previous input
      iv. 'E': Don't place into the formatted message
      v. Default: Add character to the formatted message
   d. Send the formatted message to the output pipe.
      i. If the 'T' character was detected while translating, LEAVE THE ENDLESS LOOP
      i. Else RESTART THE ENDLESS LOOP
5. Loop ended, quit program.

ProcessOutput

Grand-child of the ProcessInput and child of the ProcessTranslate, this process polls for new messages and displays the message to the screen.

1. Close the output pipe for writing (don't need it).
2. Create a counter which allows two output messages: .
3. BEGIN ENDLESS LOOP
    a. Read message from output pipe
        i. Failure: RESTART THE ENDLESS LOOP
    b. Print raw message to the terminal.
    c. Check if the 'T' character was inside of the message
        i. Increase the counter
    d. Print formatted message to the terminal
        i. If 'T' was previously found, LEAVE THE ENDLESS LOOP
    e. RESTART THE ENDLESS LOOP
4. Loop ended, quit program.

Sig_Handler(Signal)

Performs no major functions except for ignoring all (most) signal messages.

1. Do nothing.