MoSh: Report

Roll No: 150100012

The code is partitioned into 4 files: mosh.c, funclist.c, parse.c and header.h .The directory *code/* contains a makefile for compiling the above code.

Testing:

Open the terminal in the code directory and enter the command “make”. The Makefile is executed to create executable “**mosh**” in the same directory.

Note: This warning pops up:

“gcc mosh.c funclist.c parse.c -o mosh

/tmp/cc5M9985.o: In function `main':

mosh.c:(.text+0x172): warning: the `gets' function is dangerous and should not be used.”

Code:

- header.h : Contains the function declarations which are declared in mosh.c, funclist.c and parse.c
- funclist.c : Contains the function “**funcList**” which takes in a command and executes it ; and also some subordinate functions which calculates the length of arrays. “**funcList**” has an argument which determines whether the command be executed in the background or not.
- parse.c : Contains the parsing functions:
  - **tokenise** : Takes in a line from the terminal and tokenises it.
  - **checkSemiColon** : Takes in the list of tokens and parses them to find “;;” token**.** Returns list of commands splitted by “;;”
  - **checkAmpersand** : Takes in the list of tokens and parses them to find “&&” token**.** Returns list of commands splitted by “&&”
  - **parseRedirect** : Parses the token ‘>’ on providing a list of tokens. Currently I have only assumed that > appears only once in tokens.
- mosh.c : Contains the main code and functions which handle all the signal interrupts. The flow of the code is as follows:

- Every time we enter the loop, the current file directory is printed out and the prompt expects an input.
- After we enter the command, it is parsed by calling tokenise to obtain a list of tokens (If we simply hit enter, the shell doesn't break down, and simply moves on to expect another input from the user)
- The list of tokens is then sent to a parser which checks for "&&" and splits the commands accordingly to fire them up in the background. If the parser returns a command which has a length of 1, then we move on to check for ";;" in the token. In this case, the function "**funcList**" is called iteratively to execute the processes in foreground one by one.
- This file also has the function for handling signals: "**sigHandler**", which handles SIGINT and SIGKILL. The variable **fg_kill** determines whether a foreground process was killed or not during the sequential execution (set when SIGINT is called). The variable **bg_processes** stores the list of all background processes and the parent process (./mosh) kills the processes in this list before exiting.

Beautification:

I have used some inbuilt strings to beautify the shell too.