Operating System (CS 341)

ASSIGNMENT 1

Group:

- 140101004 (Abhishek Tyagi)
  - Implemented History
  - !! (to execute latest command)History
  - Debugging of code
  - Documentation and commenting
- 140101005 (Abhishek Yadav)
  - Implemented Multiple piping
  - Implemented I/O redirection
  - Parsing of input command
  - Debugging of code
- 140101057 (Rajan Garg)
  - Implemented basic commands and history
  - Parsing of input
  - Usage of fork() and execvp()
  - Documentation.
- 140101071 (Siddharth Khabia)
  - Implemented Multiple piping
  - Implemented I/O redirection
  - Parsing of input command
  - Usage of fork() and execvp()

Implementation of Basic Shell

Features:

- History feature : View last 10 commands.
- !! : Execute last command.
- ! n : Execute last nth command.
- Implemented of I/O redirection to run commands like cat > abc.txt.
- Implemented Multiple pipes to run commands like ls –l | sort.
- No space in Hard disk is used , all space and computation in RAM.
- All ~500 unix commands are supported.
- Editor commands like touch, vim and atom editors supported.
- Root privileges like sudo apt-get XXX are supported.
- Environment variables editing are supported.

Problems:

- Memory is used for storing all user commands in a linked list.
- Parser is a very basic one. It only separates command by '|' and ' \t\r\n'. Tokenizing of commands by complex delimiters not supported.

How to Run:

-Extract the assignment zip file.
-Run Terminal and go to the assignment folder.
-Type "bash bash.sh" to run the program.
-Type "exit" to quit.

Code Description:

Header files used:

stdio.h , stdlib.h, string.h, limits.h, unistd.h, sys/wait.h

C functions used:

fork(), execvp(), pipe(), dup2(), fileno(), wait_pid(), perror()

Code Structure:

1) Input Command.
2) Parsing.
   - Commands are separated by pipes(|) , if piping is present.
   - Each command is then separated by Delimiter ' \t\n\r' to obtain smaller commands.
3) Execution.
   - Commands separated by | are run iteratively in loop.
   - pipe() function is called in each iteration to create a piping system.
   - Create a child process by using fork().
   - Duplicate file descriptor of input/ output standard by using dup2().
   - Execute using execvp().
   - Error handling.
4) Repeat step 1.