OS_Lab3_Assignment

Basant Kumar Meena

14CS01007

1 Question:

Design a shell that supports the following features:

- The prompt you print should indicate your current working directory. It may also indicate other things like machine name or username or any other information you would like. Try getcwd(char * buf, size_t size).
- You should allow the user to specify commands either by relative or absolute pathnames. To read in the command line, you may want to consider the readline() function from the GNU **readline** library as it supports user editing of the command line.
- You should be able to redirect STDIN and STDOUT for the new processes by using < and >. For example, **xyz < infile > outfile** would create a new process to run **xyz** and assign STDIN for the new process to **infile** and STDOUT for the new process to **outfile**. In many real shells, it gets much more complicated than this (e.g. >> to append, > to overwrite, >& redirect STDERR and STDOUT, etc.)!
- You should be able to place commands in the background with & at the end of the command line. You do not need to support moving processes between the foreground and the background (ex. bg and fg). You also do not need to support putting built-in commands in the background.
- You should maintain a history of commands previously issued. The number of previous commands recorded can be a compile time constant, say 10. This is a FIFO list, you should start numbering them with 1 and then when you exhaust your buffer you should discard the lowest number, but keep incrementing the number of the next item. For example, if storing 10 commands, when the 11th is issued, you would be recording commands 2 through 11.
- A user should be able to repeat a previously issued command by typing **! number**, where number indicates which command to repeat. !-1 would mean to repeat the last command. !1 would mean repeat the command numbered 1 in the list of command returned by history.
- A built-in command is one for which no new process is created, but instead the functionality is built directly into the shell itself. You should support the following builtin commands: **jobs, cd, history, exit** and **kill**. Jobs provide a numbered list of processes currently executing in the background. **cd** should change the working directory. **history** should print the list of previously executed commands. The list of commands should include be numbered such that the numbers can be used with ! to indicate a command to repeat. **exit** should terminate your shell process, **kill %num** should terminate the process numbered, num in the list of background processes returned by jobs (by sending it a **SIGKILL** signal).
- If the user chooses to **exit** while there are background processes, notify the user that these background processes exist, do not exit and return to the command prompt. The user must **kill** the background processes before exiting.
- You may assume that each item in the command string is separated on either side by at least on space (e.g. prog > outfile rather than prog>outfile).

2 Answer

- To run program use **gcc bshell.c -o sh -lreadline** command. where **-lreadline** to link **GNU readline library**.
- Basically bshell.c program contains **three functions**.
  1. **Readline:**To read commands from user (using GNU readline library).
  2. **ParseCmd:**To parse commands given from readline
  3. **ExecuteCmd:** To Execute commands and show result
- I am using **GNU readline library** for reading commands from user.
- FDs manipulation (using **dup(),dup2(),close() etc.**) used for Redirection of **STDIN** and **STDOUT** to **inputfile** and **outfile**.
- To implement **Background Process Command**, i am not waiting for **child process** to end after fork() and after that i am ripping all **zombie** processes.
- **History** was implemented manually to support **!1 , !-1** commands and Max 20 Numbers of history can be store.
- Please run and test the bshell.c program to check all functionality of shell.