SimpleShell: A Unix Shell in C from Scratch

Contributors: Aditya Gupta & Abhishek Bansal

Individual Contribution

- Aditya: base code for the shell (fork commands in the do while loop and reading and splitiing the lines) , including the specifications of pid , execution time , etc in the history . code for the pipe commands . starter code for using '&' for the background processes .
- Abhishek: starter code for history (write_history and clean_history) . handling history with pipe commands . improving the '&' for background processes. running commands from a file (.sh file) by making the run_script function . Error handling .

Implementation

the steps for the implementation of the shell in c are as follows:

1. The main method calls 2 functions which are signal and shell_loop. If the user sends ctrl+c as the input then , the signal calls the handler function and prints the lines "Ctrl + C was pressed! Exiting..." and "Showing history..." followed by the history along with the details of each command . If the user does not press ctrl + c , then the shell loop is invoked and in the shell_loop function, a do-while loop is present which runs endlessly.(till crtl+c is pressed) .
2. now the command is taken using the readline function which uses the getline function call to take the user input . Now this is a line with spaces between the arguments . Now this command is passed onto the splitting function which breaks the command in an array of char* (char **args) which are broken by the delimeters " " . This will help us in inputting these as the arguments in the execvp system call ahead .
3. we are also handling flags background ,bg_ended for the background process . The background process have "&" as there last argument . Thus the flag is made 1 for them otherwise it is 0 .Now we have broken the args further based on the piping . Like if we have cat fib.c — grep printf — wc -l , it will become "cat", "fib.c",NULL ,"grep" ,"printf", NULL,"wc", "-l", NULL .This will help us iterate in the piping now. This is stored in the char *commands[10][MAX_ARGC+1] .
4. Now as any normal shell, we are making a child process and making the parent process wait for the child process if the child is not be run at the background. If the background flag of the child is 1 then, the parent will not wait for the child process to stop and the child process will run in the background as a zombie (eventually ending after adoption from init) .
5. Now in the child process ,we have used pipe function call in a for loop for inputting the output of one command into the input of the next command . Thus we will do this till the last command is left and handling the last command separately. This allows us to run both piped and non-piped commands . Also we are handling the history command seperately and the run command seperately . "run file_name" is used for running commands inside a .sh file.
6. for the commands.sh file we are using the run_script function which takes filename as the argument and reads the contents line by line to give output for each and every command present in that . Except for the for loop which takes inpout line by line by fgets the whole code is similiar to the one in the shell_loop.
7. the get_history, write_history and clean_history functions are helping us to manipilate the history.txt file and cleaning it afterwards when the program stops working.
8. We have also implemented the cd commands using the chdir() system call and we are changing the path as we go inside a directory using cd dir_name or coming out of the directory using cd .. , This is our implementation of the SimpleShell.

Github Repository

WHICH COMMANDS WILL NOT BE RUN IN OUR SHELL

1. The commands which are not present in the /usr/bin directory except for the history and the cd commmands will not be able to run on our shell as the execvp command will not be able to access them like alias, etc . Also the commands which have two commands seperated by && or ; are not supported in our shell . For example: cat fib.c && history | grep clear and cat fib.c ; history | grep clear. Also globbing is not supported in our shell for example: ls [[:alpha:]]* . Thus these are some set of instructions not supported in our shell .

Steps to run:

To run non-piped and piped commands

1. run the c file simpleShell.c and then the terminal will open. Write the commands in the command prompt to see them working .
2. you can also run history command with pipes and the history command will show all the information about a process like process pid, time at which the command was executed, total duration the command took for execution . These are for the non-piped and piped commands.
3. for the & commands we have to write "command_name &" and it will work for them .

To run commands inside .sh file

1. to run the commands inside of a .sh file we have to type "run file_name" to run all the commands in the file (if they are correct and executable) .

THESE WERE SOME OF THE INSTRUCTIONS TO RUN THE COMMANDS .