Mini shell

Authored by Natan Protector

==Description==

This program implements a simple command prompt shell for linux. on activation the shell will prompt the user to input a command to execute a program by typing in the name of the program to be activated, followed by the arguments for the program all separated by spaces. This basic shell has the following features:

* Chaining commands separated by a semicolon ‘;’.
* Chaining output/input between commands using ‘|’.
* Output final result to text file using ‘>’ and the formal <command>><file name>
* Pause last command using ctrl z, unpause and move to background using bg.
* run commands in background from the beggining using ‘&’ and the format <command>&
* Declaring local variables in the following format: $<variable name>=<something> To be then used again be declaring <something maybe>$<variable name><space or / or “>.
* All arguments between two quotes in the input will be used as one argument.
* When inputting an empty command three times in a row will exit from the shell.
* The shell keeps track of the number of commands and number of separate arguments passed so far and together with the address of the current director, displays them in the terminal when prompted for an input.

No support for changing directory “cd” command.

Program DATABASE:

Two dynamic arrays of dynamic strings used to contain the local variables. One for the variable names and one for the corresponding values, matched by the index in the array.

Functions:

* One main function that contains the main loop that’s prompting the user for input, and inner loop that’s parsing through strings separated by semicolons.
* Prompt - displays current directory, and the number legal arguments and commands given to the shell from the start of the program.
* Parse - returns a string with the next argument starting from an index.
* variableDeclaration - determines whether a command is a legal variable declaration and returns the index of the declaration name.
* getValue - based on a name input, return the corresponding value from the local variable s array.
* two pause handlers: handlerForChildTermination() and pauseHandler(), one for terminating children processes finishing in background and one for handling SIGTSTP signal and pause child.
* function for handeling pipes: createPipe(), closePipe()
* function for a handeling dup2 process for changing file descriptors: dupTwo()
* myStrTok() is a custom tokenizer used to ignore delimiters that are inside quotes

==Program Files==  
ex1.c - contains the main, declarations, functions and database.

==How to compile?==  
compile: gcc ex1.c -o ex1  
run: ./ex1

==Input:==  
User input between every execution, input in the following format:

<name of command> <arg1> <arg2> <arg3>...

Or for variable declaration with following format:

<something maybe>$<variable name><space or / or “>.

==Output:==  
Other than the prompt and error messages there are no outputs associated with the shell itself.

Outputs are generally expected from the executed commands.