OSC Projects

Calin Iaru s1077000

September 2022

1 Implementation

The shell implementation can be separated in two parts. First we have the built-in or internal commands. These consist of exit and cd.
Exit is implemented as it will only be executed if the keyword exit is found in the first postion of the first command, and there aren't any other piped comands afterward, e.g. exit "anyhting" will execute and the shell will close, exit "anyhting" — "anything" won't execute.
Cd is implemented as it will only execute if it's the only command, so no other commands are piped in front or behind it. If only cd is input, the shell will change directory to HOME directory, set in the environment variables. If a vaid path is given to cd, the directory will change to that argument. Otherwise an error will be thrown in screen.

If no exit or cd commands are found according to above rules, the shell starts the execution of internal commands.

This, also means that if there are chained commands, and there is an internal command, it will not be recognized; so in our shell, the internal commands should be the first ones.

For the expressions that consist of more pipped commands, we loop over the commands, and for each command We used the pipe function and then we create a child process using fork().

Furthermore, we declared a variable called prev_pipe, which keeps track of the read end of the previous command. In each child process, first, the input is determined. For the first child process, we also check if the inputFromFile string is empty or not.

If it is not, we open the given file and take it as input. This is achieved via the open() and dup2() system calls. Any subsequent child processes take input from the previous process via prev_pipe, otherwise prev_pied=STIN_FILE.

This is also achieved via the dup2() system call where we take the read part of a pipe as input. Once we are done with the input, we decide where the output should go.

The last process checks if there is an output file. If this is the case, then using the functions open() and dup2() will be used to open, create, or/and overwrite the file.

In the parent we close the write end of the current command, and using the variable prev_pipe we save the next read end of the current command to use in the next iteration.