Projet Architecture-Système 2

Martin Cuingnet

December 15, 2023

1 How to compile

Use `make` to create the executable `shell` and then launch it with `./shell`.
Some tests can be found at `Test/testsuite.txt`.

2 Answers

2.1

A plain command is a command that does not use any shell operator like | or && and doesn't use parenthesis.
I use the `execvp` command.
Here are the reasons why I did not choose the others functions `exec(L|V)(P|E)`

- **L** : because the number of arguments is not given in the `cmd` struct.
- **E** : because the `PATH` can't be accessed (which is needed to fully interact with files)

2.2

The sequence operator is ; in bash.
The command `cat non_existing_file && echo hello` will produce the output
`cat: non_existing_file: No such file or directory`
whereas the command `cat non_existing_file ; echo hello` will produce the output
```
cat: non_existing_file: No such file or directory
hello
```

2.3

I implement `C_AND` and `C_OR` by first executing the left command and if its result is true (resp. false), I execute the right command.

2.4

Yes, this command is possible.
The purpose of the parentheses is to change the order of execution of the commands, here enabling the path at the end to be taken as an argument by the whole command.
For instance, `(ls ; ls) | wc` will print the number of bytes in the concatenation of the two outputs produced by `ls` and `ls ; (ls | wc)` will print the output of `ls` and then its number of bytes;

2.5

When `Ctrl + C` is pressed, the current process stops and the shell closes. To prevent the shell from closing, and instead close the current process that the shell is executing, I need to catch the `SIGINT` signal produced by `Ctrl + C` using `signal` and then reset the display (to avoid display bugs).

2.6

The command displays the output of `ls` in the standard output instead of printing the result in the dump file.

2.7

Conversely, to `dup2`, `pipe` creates new file descriptors for reading and writing. The behaviour of `pipe` can't be replicated using `dup2` because it will use the stdin and stdout as the pipe, meaning that a command like `ls | wc` will print the output of `ls` to the terminal before printing the actual output of `ls | wc` : which is not the expected behaviour of `pipe`.

3 Implementation

3.1 Plain commands

When needing to execute a command :

- The shell forks
- The son apply its possible redirection and executes the command
- The shell waits for its son and returns its status

3.2 Operators different than pipe

When handling an operator other than `pipe` :

- I call recursively `execute` on `cmd->left`

- Depending of the return status of `cmd->left`, I call `execute` on `cmd->right`

3.3 Pipe

When handling `pipe` :

- The shell creates a pipe
- The shell forks
- The son dupes its `STDOUT_FILENO` to the output of the pipe and executes `cmd->left`
- The shell waits for its son and forks again
- The new son dupes its `STDIN_FILENO` to the output of the pipe and executes `cmd->right`
- The shell waits for its son and return its status

I had soem troubles implementing `Ctrl+C` : I realized after some time and research the functions needed in the handler to reset the shell's interface.

4 Bonus

4.1 Wildcards

I implemented the wildcard extensions :

- First, I check in every argument of `cmd` is there is a `*.` wildcard pattern (using `strstr`)
- If I find one, I use a pipe to first call `ls path | grep '.* .extension$'` (which collects every file at `path` ending with `.extension`)
- I get the result of the command in `stdin` with successive `fgets` in a string array
- I create a new `args` array by replacing the widlcard argument with the string array produced by the `ls | grep` command
- I recursively call `execute` on `cmd` with the record `args` being replaced with the newly created arguments.