ECE 695 Shell

Yizhou Shan
Shan13@purdue.edu

Overall Design

The mechanism of a shell is very simple and straightforward: 1) standby waiting for input, 2) parsing input into commands, 3) run commands properly.

In the parsing stage, we parse token by token. The tokens are predefined, and have very specific rules about their type and usage. Tokens together will form a command, which consists of executive file name, and several arguments. After all, several commands together will form a list of commands.

The commands within a list has specific relationship with each other. For example, the next command will be executed if and only if the previous command returns success. The semantics in my implementation is similar to bash.

To run commands, the parent process need to handle multiple cases. Normally, it will just fork a new process and load the executive file. But for some built-in commands, e.g., cd, exit and jobs. The parent shell process needs to execute these commands in its own process, because these have to happen within the same process.

The subshell is handled by forking a child process to run the new subshell. The CMD_AND, CMD_OR and CMD_PIPE are handled properly between subshell and parent shell.

Noteworthy

Personally speaking, I think the most useful features of a shell are its redirection and pipe. During the implementation, I learned a lot about the pipe, and file number etc. In order to make these two features work, we need to handle the file numbers very carefully, for example, we need to invoke dup2 properly.

Aside

During these days, I relearned some bash script, it is very helpful. And I also take a quick look at the GNU bash source code, which, to me, is written in an unfriendly way.

Known Bugs

At the end of the subshell process, I will call exit(status) to end subshell. And the parent shell process will wait this event and capture the exit status. However, the waitpid function reports that the chils process terminated because of some signal (which is 14 or 55), but not exit().