## P3. Cluster Shell

### Concepts Used

client-server communication via TCP, processes and fork, threads using pthread

### Design

The program is divided into two files based on general functionality. One file codes the behavior of the server and the other codes for the behavior of the clients.
The programs implement a cluster shell, that is, it creates an environment that allows multiple shells to run commands on each other and obtain the results.
In the environment created, there is one server and multiple clients.

Server: The server is responsible for delegating the commands it receives to the corresponding clients, getting the output produced as a result of running the tasks assigned to the client(s) and sending back the combined output to the client that requested the command to be run.

Client: Every client has two distinct behaviors. These are implemented using processes. The parent process runs the front-end to the shell, displays the prompt, gets commands as input from the user and displays the output. The child process runs silently in the background and waits for the server to send it a command. Once it receives a command, it executes it and returns the output produced to the server.

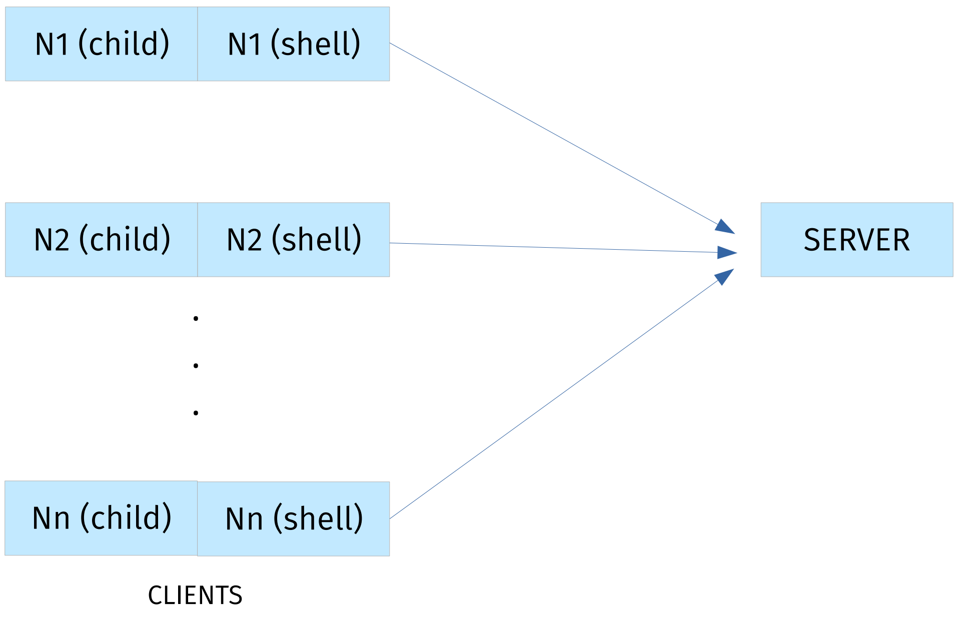

Fig 1

*The clients' shells are all connected to the server. They send the commands received as input on these TCP sockets to the server and receive the command outputs on these sockets as well.*

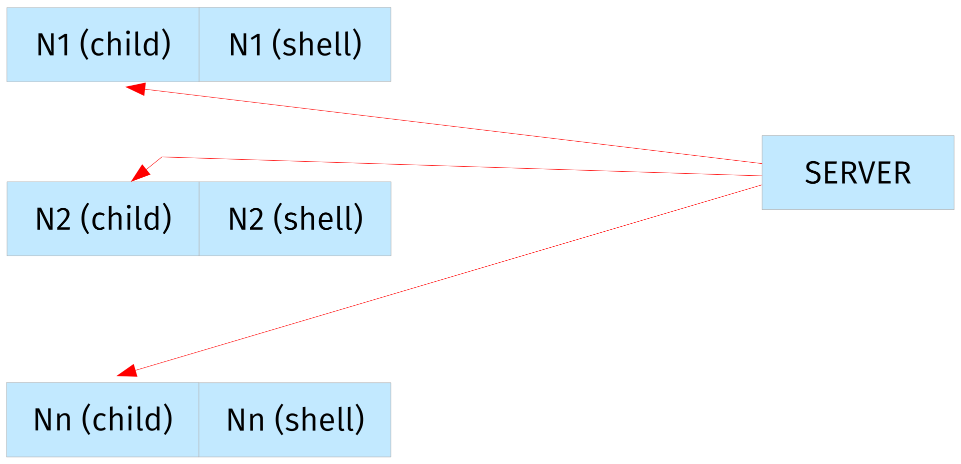

Fig 2

*The connections that the server makes for each sub-command with the clients that are mentioned in the command obtained.*

Example: if n1 runs `n2.ls|n3.wc` then, first n1's shell connects to the server and sends the whole command to it. The server send each sub-command to its corresponding client (ls to n2 and wc to n3). After performing all operations, it then sends the output of this command back to n1 which has been waiting for the output.

## Files Included

- clustershell_client.c
- clustershell_server.c
- config
- Makefile

## How to Run

- To compile, use `make`, or `make all`
- to run server, `./srv`
- to run clients `./cli <node name> (eg. ./cli n1)`
- *`run the server before the clients`*
- *`when using pipes, do not give spaces between the sub-commands,`* `(valid : n2.ls|n3.wc) (invalid : n2.ls | n3.wc)`

## Limitations

- the shell process and the child process cannot communicate directly. Upon running `n1.cd <path>`, commands `n1.ls` and `ls` give different outputs.
- The `exit` command only quits from the shell, it does not kill the child process. (to quit from both, use `ctrl-c` instead)

Ashwin Kiran Godbole, 2018B5A70423P
Samarth Krishna Murthy, 2018B2A70362P