Network Programming Assignment 1

Design Document

This is a cluster shell which can run commands on all its connected servers.

Date Submitted: October 10, 2019

Vidit Jain (2016A7PS0064P)
Jitvan Shukla (2016A7PS0083P)

Deliverables included: clustershell_client.c, clustershell_server.c, run.sh

How to run client:

```
$ bash run.sh clustershell_client
$ prompt>
```

Now you can start using the cluster shell

How to run server:

```
$ bash run.sh clustershell_server
```

This will get connected to the client to run commands when asked by client

Config File Format-

```
Number of nodes
Node name ip.address
Node name ip.address
…….
```

Constraints:

For commands that take input, the input has to be entered in the server itself if it is not a pipe.

The client and servers all must have different ip addresses.

Working of the code:

Once the prompt starts, it will work like a normal shell. Type ‘exit’ to return to your normal
terminal. While writing commands, you can use as many whitespaces as you want.

Implementation:

- The client has parent process trying to connect to all inactive servers forever whereas the server, when started, accepts the connection since a client should initiate a connection.
- When the user enters a command, the child process sends the command to the parent process, signals the parent and terminates.
- A new child process is spawned after the execution of the previous command which asks for user input command. This was done to ensure socket connections remain the same for parent and child process.
- Once the command is entered, it is parsed to check the existence of pipe in command.
- If pipe not present,
    - nodeidx = -2 (local command), nodeidx=-1(all nodes), nodeidx=+ve number, index of node in the list.
    - For local, the command is simply run on the client by executing after a fork.
    - If the command is meant for 1 server, the command is sent to the server using tcp connection where the server runs the command and returns the output to the socket.
    - If the command is meant for all nodes, runcommand() is run for every active node once.
- If pipe present, a pipe() is created for each | and stdout is also sent to the stdin of client along with command (in the case of local command) or it is sent to the server through tcp socket along with command(in case of server command). The extra input through tcp socket is handled by sending 0 or 1 by client to server to indicate if the command is piped or not.

- If nodes is entered, sleep 0.00001 is sent to every active node so that disconnected nodes are detected.