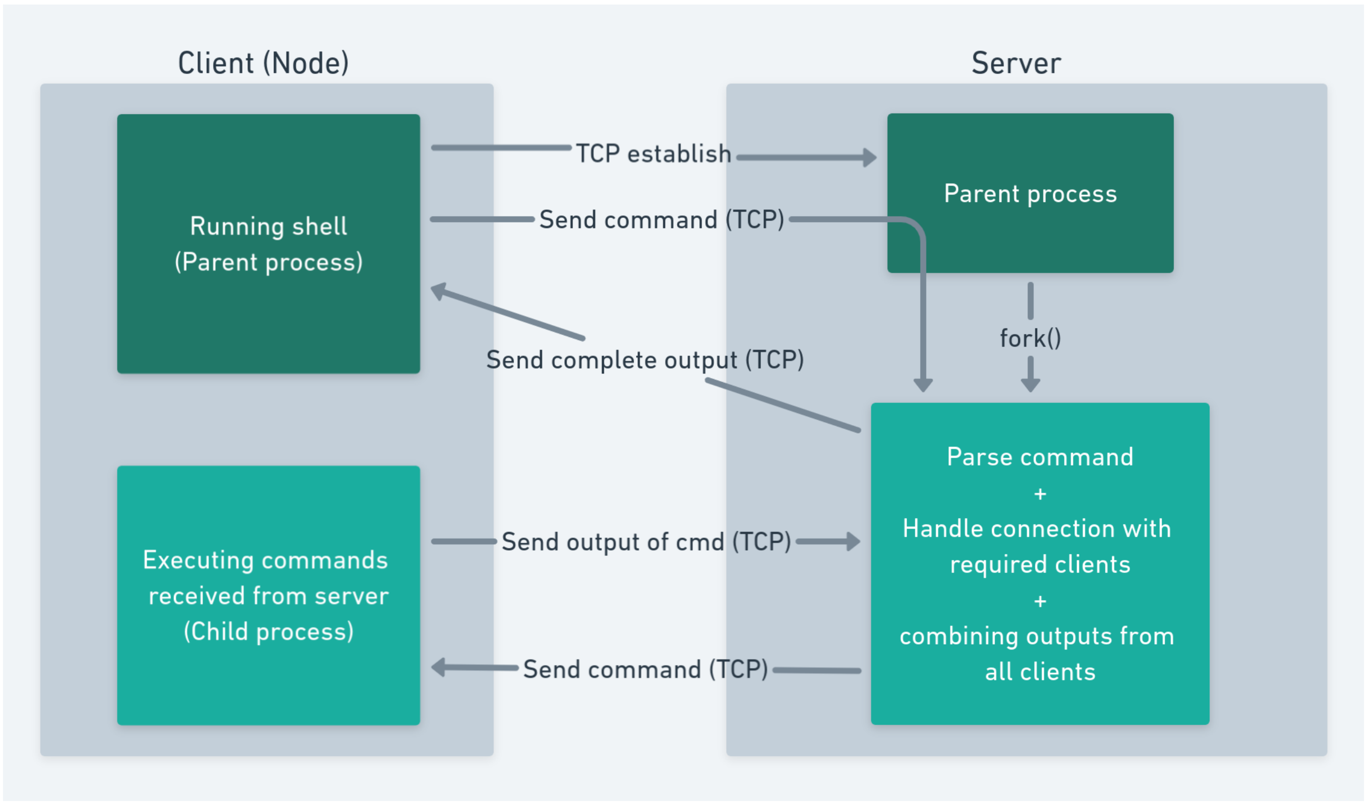
Cluster Shell

Submitted to: Dr. Hari Babu
Course: Network Programming IS F462
Assignment 1

Design

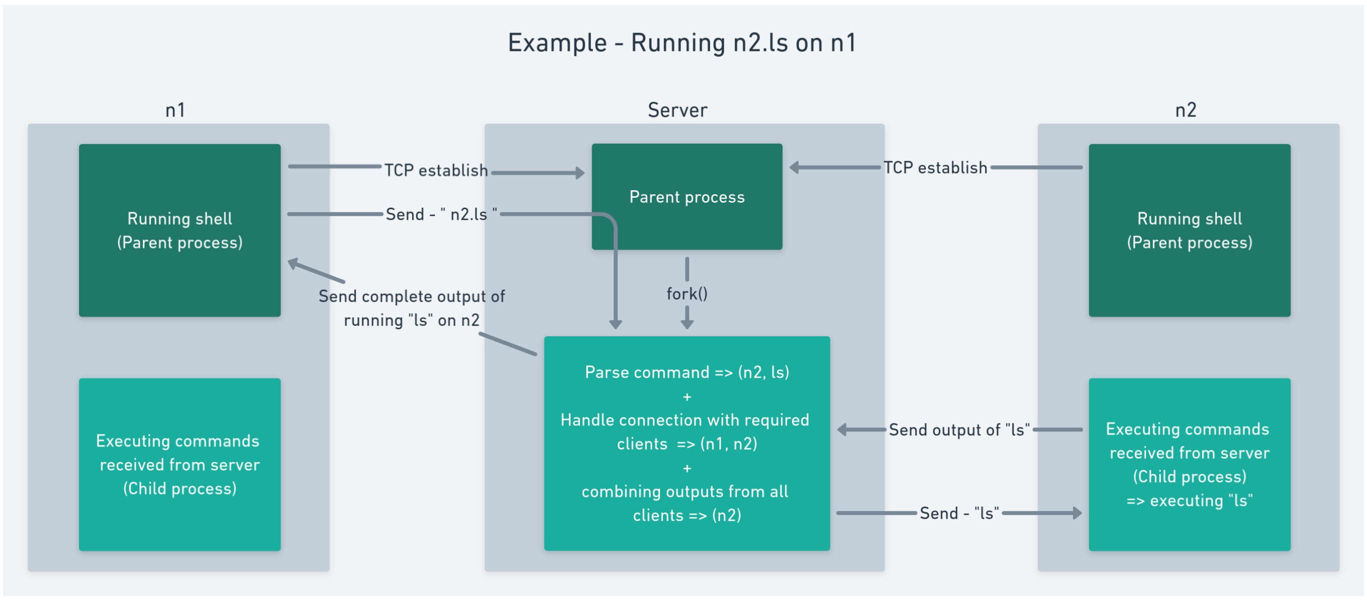
This project aims to extend the shell features to a cluster of machines. There is a central machine (server) that coordinates all the interactions between different nodes (clients).

The following figure gives an illustration of the design we incorporated.


Client (Node)
Server
Running shell (Parent process)
Executing commands received from server (Child process)
TCP establish
Send command (TCP)
Send complete output (TCP)
Send output of cmd (TCP)
Send command (TCP)
Parent process
fork()
Parse command + Handle connection with required clients + combining outputs from all clients

As soon as a node (client) starts the shell on their machine, a TCP connection is requested and established with the server (which creates a new child process to handle the connection), and the shell is then ready to accept commands from the user. There is also a child process created on the client, which waits for an other connection to be established with the server. This child process is responsible for executing the commands that are received from the server and are required to be executed on the given machine.
When the user enters the command on the shell, it is sent to the server. On the server, the command is first parsed to break it into sequence of (command, node number) pairs, so that the server knows which command is to be executed on which node. The output from all the nodes is collected by the server and the result is sent to the command executing node.

The following figure shows how an example command would run on this architecture.


Example - Running n2.ls on n1
n1
Server
n2
Running shell (Parent process)
Executing commands received from server (Child process)
TCP establish
Send - " n2.ls "
Send complete output of running "ls" on n2
Parent process
fork()
Parse command => (n2, ls) + Handle connection with required clients => (n1, n2) + combining outputs from all clients => (n2)
TCP establish
Send output of "ls"
Send - "ls"
Running shell (Parent process)
Executing commands received from server (Child process) => executing "ls"

Lifecycle of a command execution

- **When server starts** - It creates a socket and starts listening for connections from clients. It works like a concurrent server, where a new process is spawned to handle each new connection. When a connection is established with a client, server waits to read a message from the client.
- **When client starts** - It tries to establish connection with the server (i.e. server should be running before trying to run the client). After the connection is established, it starts running a shell to accept commands from the user. The client also spawns a child process in the beginning, which creates a socket and starts listening for connection from the server. This process would be responsible for executing commands whose output is required by other nodes (clients).
- **When user enters a command on the client shell** - The command is sent to the server via the already established connection. The shell then waits to read the output from server on this connection.
- **When server receives command from the client** - First it parses the command to break it at pipes (`|`) creating smaller sub-commands and to determine which sub-command is to be sent to which client for execution (via name-IP pairs specified in CONFIG file). After parsing, the sub-commands are sent to the respective IP addresses, via a connection where the client acts as the passive end.
- **When client receives command from the server** - Client runs the received sub-command using `popen()`. The output is first written into a pipe, which is then read and sent back to the server.
- **After server receives outputs from all the clients** - It combines them as required and sends them to the original client that ran the main command.
- **After client receives output from the server** - The output is displayed for the user to see and the shell is ready to accept another command.

Usage

Update the CONFIG file with the name and IP values for each client. Open 1 terminal for the server and run

```
```

```
make server
./server.out
```

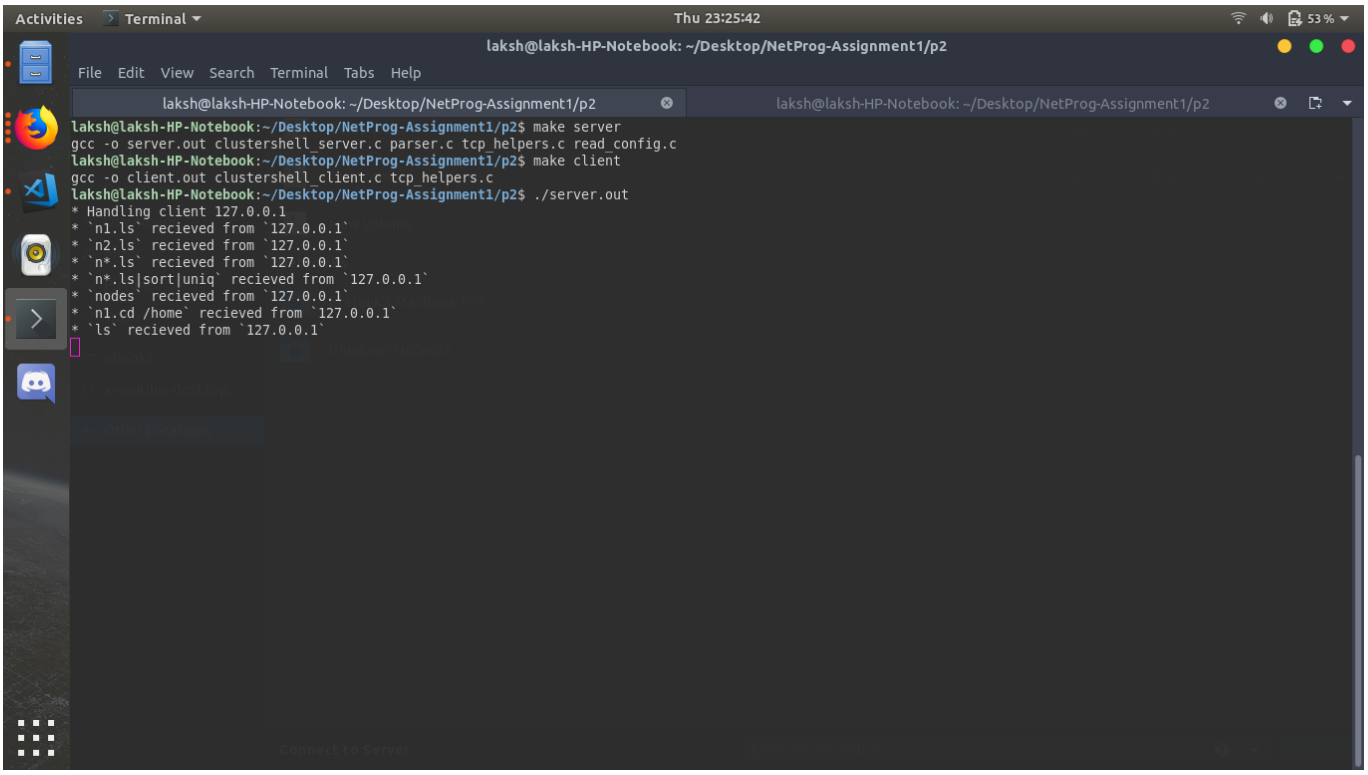
For n clients open n terminals and run

```
make client
./client.out
```

Make sure that server is running before clients are run.

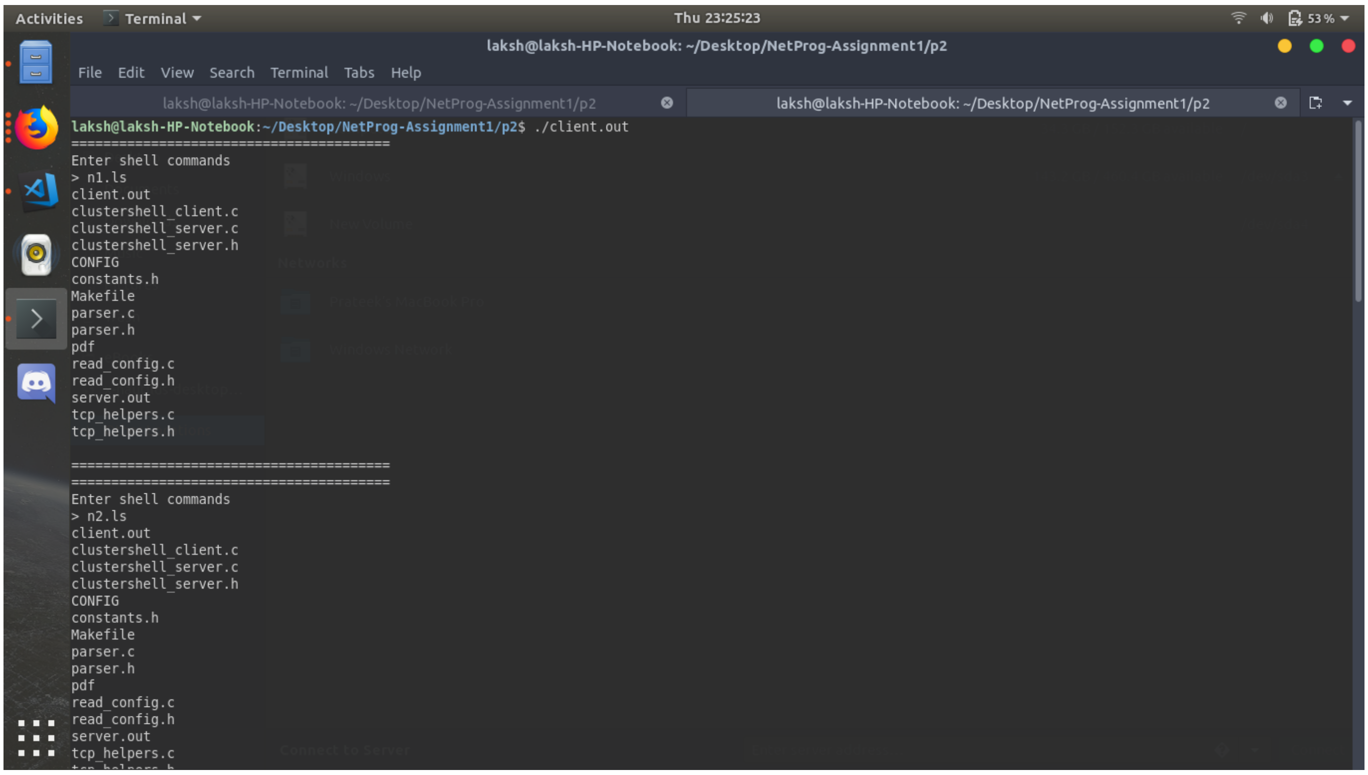
The client is now ready to accept commands from the user.
Logs of all communications are visible on the server.

Screenshots (Server)

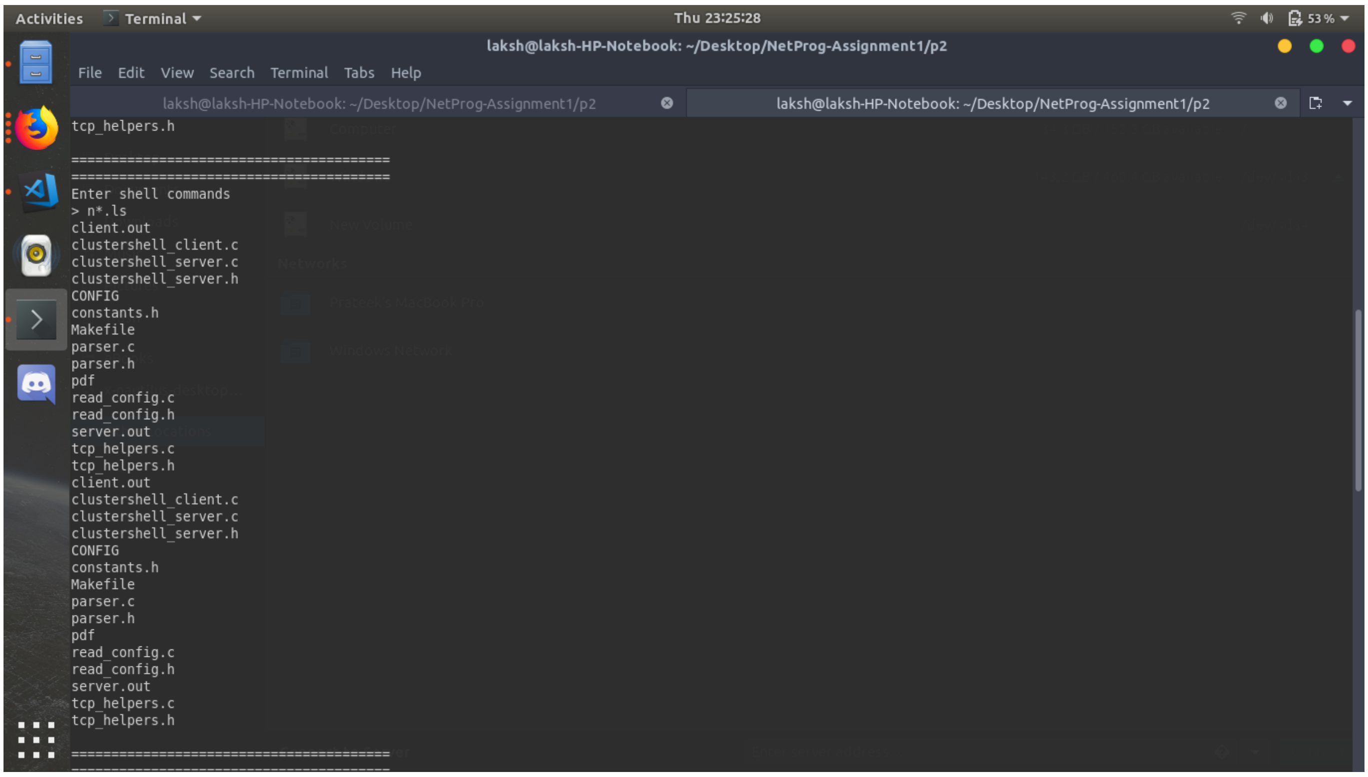

Activities
Terminal
Thu 23:25:42
53 %
laksh@laksh-HP-Notebook: ~/Desktop/NetProg-Assignment1/p2
File Edit View Search Terminal Tabs Help
laksh@laksh-HP-Notebook: ~/Desktop/NetProg-Assignment1/p2
laksh@laksh-HP-Notebook: ~/Desktop/NetProg-Assignment1/p2

```
laksh@laksh-HP-Notebook:~/Desktop/NetProg-Assignment1/p2$ make server
gcc -o server.out clustershell_server.c parser.c tcp_helpers.c read_config.c
laksh@laksh-HP-Notebook:~/Desktop/NetProg-Assignment1/p2$ make client
gcc -o client.out clustershell_client.c tcp_helpers.c
laksh@laksh-HP-Notebook:~/Desktop/NetProg-Assignment1/p2$ ./server.out
* Handling client 127.0.0.1
* `n1.ls` recieved from `127.0.0.1`
* `n2.ls` recieved from `127.0.0.1`
* `n*.ls` recieved from `127.0.0.1`
* `n*.ls|sort|uniq` recieved from `127.0.0.1`
* `nodes` recieved from `127.0.0.1`
* `n1.cd /home` recieved from `127.0.0.1`
* `ls` recieved from `127.0.0.1`
```

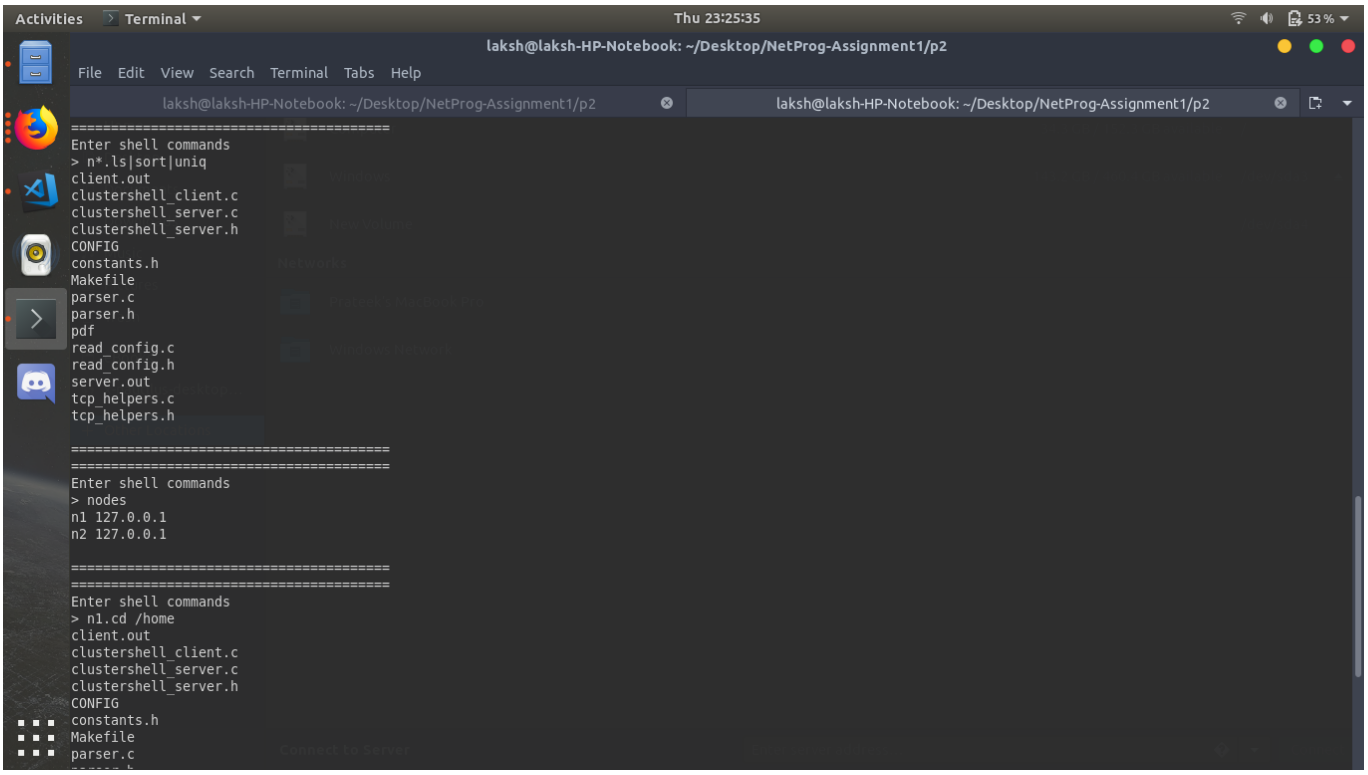
Screenshots (Client)



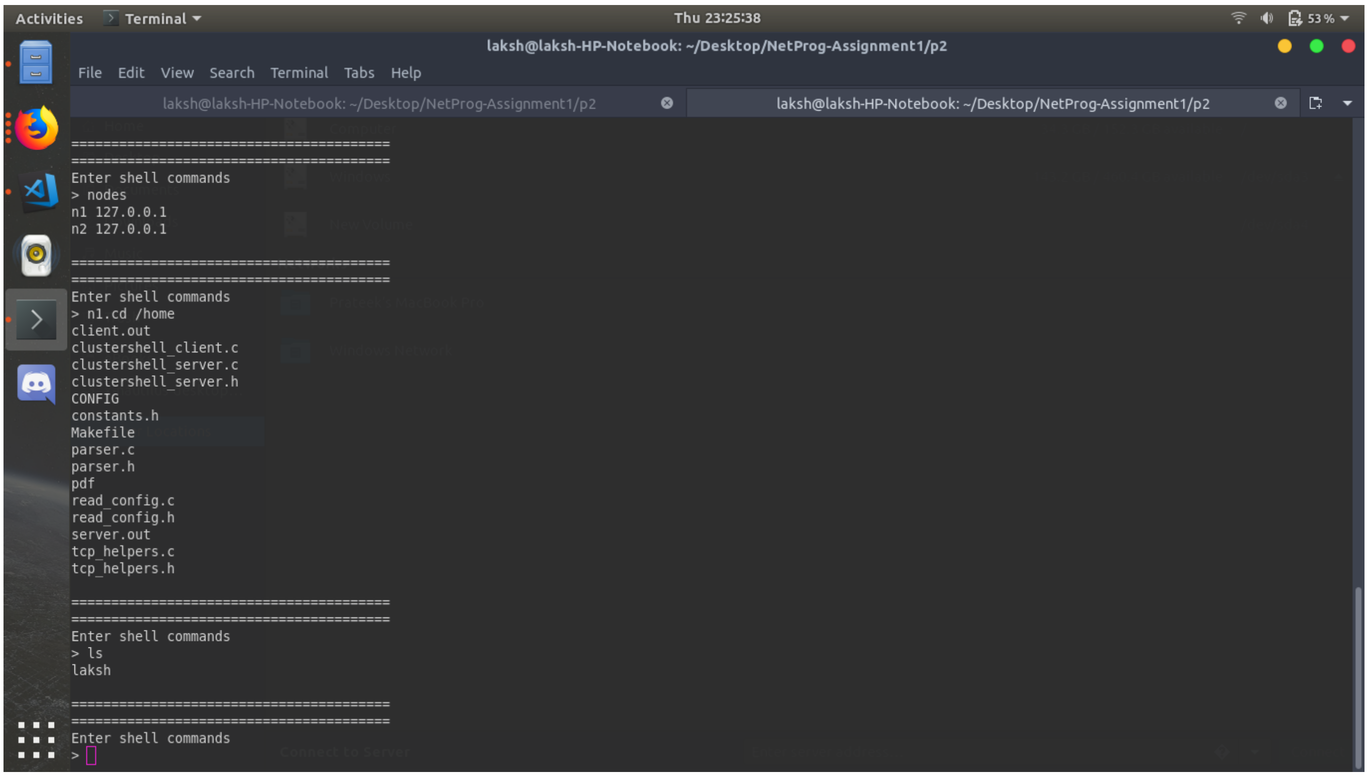
```
laksh@laksh-HP-Notebook:~/Desktop/NetProg-Assignment1/p2$ ./client.out
========================================
Enter shell commands
> n1.ls
client.out
clustershell_client.c
clustershell_server.c
clustershell_server.h
CONFIG
constants.h
Makefile
parser.c
parser.h
pdf
read_config.c
read_config.h
server.out
tcp_helpers.c
tcp_helpers.h

========================================
========================================
Enter shell commands
> n2.ls
client.out
clustershell_client.c
clustershell_server.c
clustershell_server.h
CONFIG
constants.h
Makefile
parser.c
parser.h
pdf
read_config.c
read_config.h
server.out
tcp_helpers.c
```



```
tcp_helpers.h

========================================
========================================
Enter shell commands
> n*.ls
client.out
clustershell_client.c
clustershell_server.c
clustershell_server.h
CONFIG
constants.h
Makefile
parser.c
parser.h
pdf
read_config.c
read_config.h
server.out
tcp_helpers.c
tcp_helpers.h
client.out
clustershell_client.c
clustershell_server.c
clustershell_server.h
CONFIG
constants.h
Makefile
parser.c
parser.h
pdf
read_config.c
read_config.h
server.out
tcp_helpers.c
tcp_helpers.h

========================================
```


Activities
Terminal
Thu 23:25:35
laksh@laksh-HP-Notebook: ~/Desktop/NetProg-Assignment1/p2
File Edit View Search Terminal Tabs Help

```
========================================
Enter shell commands
> n*.ls|sort|uniq
client.out
clustershell_client.c
clustershell_server.c
clustershell_server.h
CONFIG
constants.h
Makefile
parser.c
parser.h
pdf
read_config.c
read_config.h
server.out
tcp_helpers.c
tcp_helpers.h

========================================
========================================
Enter shell commands
> nodes
n1 127.0.0.1
n2 127.0.0.1

========================================
========================================
Enter shell commands
> n1.cd /home
client.out
clustershell_client.c
clustershell_server.c
clustershell_server.h
CONFIG
constants.h
Makefile
parser.c
```


Activities
Terminal
Thu 23:25:38
laksh@laksh-HP-Notebook: ~/Desktop/NetProg-Assignment1/p2
File Edit View Search Terminal Tabs Help

```
========================================
========================================
Enter shell commands
> nodes
n1 127.0.0.1
n2 127.0.0.1

========================================
========================================
Enter shell commands
> n1.cd /home
client.out
clustershell_client.c
clustershell_server.c
clustershell_server.h
CONFIG
constants.h
Makefile
parser.c
parser.h
pdf
read_config.c
read_config.h
server.out
tcp_helpers.c
tcp_helpers.h

========================================
========================================
Enter shell commands
> ls
laksh

========================================
========================================
Enter shell commands
> 
```

Assumptions

- Commands starting with `./` are assumed to be relative to the current directory, and hence are not searched in `PATH` variable.