

Client/Server Chat Program Testing

Wilson Sue & Charmie Jung
November 18, 2023

Features	2
Reading commands	2
Separating commands	2
Parsing commands	2
Executing commands	2
Exiting	2
Test Results	3
Examples	4
./build/chat -a <ip address> <port>	4
./build/chat -c <ip address> <port>	5
./build/chat -c <ip address> <port> < <file path>	5

Features

Reading commands

Commands are implicitly read from the standard input (stdin) in the client mode, where user messages are sent to the server. The server, meanwhile, listens to incoming connections and handles client messages. There is no explicit command reading mechanism built into the server.

Separating commands

In the client mode, each line entered by the user is considered a separate command (message) and is sent to the server. The server then broadcasts this message to other connected clients. There is no explicit separation of commands into smaller units, and no error is reported for an empty command (all whitespace).

Parsing commands

The program does not involve a complex command parsing mechanism. The main function parses command-line arguments to determine the mode (server or client), server address, and port. The parsed components are:

- Mode of chat program (-a for server, -c for client)
- IP address of the server
- Port number for the connection
- Optionally IO redirection

Redirection can include the ~ directory which will be expanded to the users home directory.

Executing commands

The program executes based on the mode specified in the command-line arguments and the ip address passed in along with the port. Optionally you can add IO redirection.

Exiting

When a command exists it will either display the exit status or if the command cannot be found, an error message.

Test Results

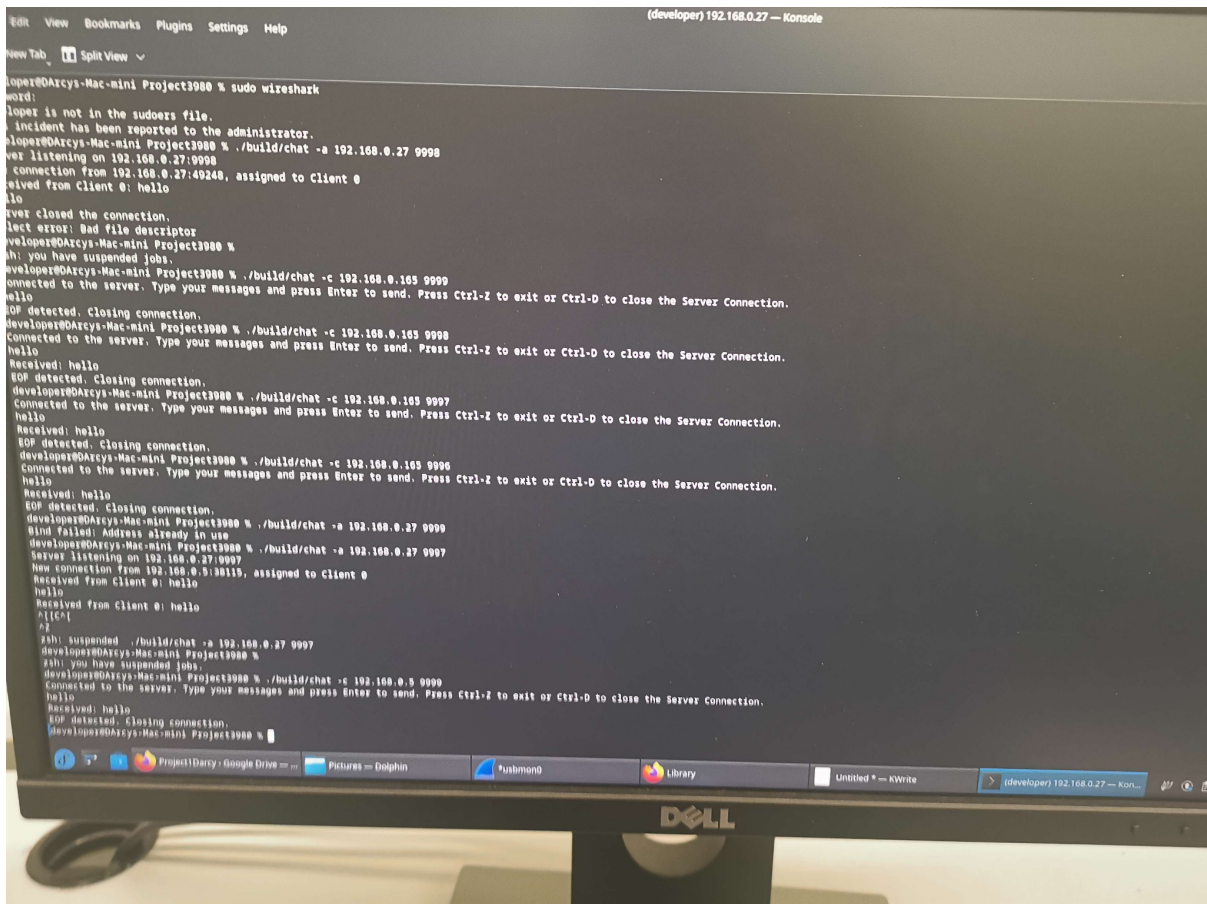
Command	Description	Status	Example
<code>./build/chat -a <ip address> <port></code>	Empty command just goes back to read another command	Passed	Example 1
<code>./build/chat -c <ip address> <port></code>	Print the working directory	Passed	Example 2
<code>./build/chat -c <ip address> <port> < <file path></code>	Go back a directory	Passed	Example 3

Examples

`./build/chat -a <ip address> <port>`

```
[wilsonsue@fedora Project3980]$ ./build/chat -a 192.168.0.165 9999
Server listening on 192.168.0.165:9999
New connection from 192.168.0.3:45884, assigned to Client 0
Hello
Received from Client 0: Hello
^Z
[1]+  Stopped                  ./build/chat -a 192.168.0.165 9999
[wilsonsue@fedora Project3980]$ ^C
[wilsonsue@fedora Project3980]$
exit
There are stopped jobs.
```

`./build/chat -c <ip address> <port>`



`./build/chat -c <ip address> <port> < <file path>`

File used to test:

```
1 testing1 testing2
2 alksnjdjasnzd
```

Using the command listed above

```
[wilsonsue@node-1w7jr9pqxdq7w7wvu12c901gt Project3980]$ ./build/chat -c 192.168.1.70 9999 < testFile.txt
Connected to the server. Type your messages and press Enter to send. Press Ctrl-Z to exit or Ctrl-D to close the Server Connection.
EOF detected. Closing connection.
[wilsonsue@node-1w7jr9pqxdq7w7wvu12c901gt Project3980]$ S
```

Output

```
[wilsonsue@fedora Project3980]$ cd ..  
[wilsonsue@fedora CLionProjects]$ cd Project3980  
[wilsonsue@fedora Project3980]$ ./build/chat -a 192.168.0.165 9999  
Bind failed: Address already in use  
[wilsonsue@fedora Project3980]$ ./build/chat -a 192.168.0.165 9999  
Server listening on 192.168.0.165:9999  
New connection from 192.168.0.3:43376, assigned to Client 0  
Received from Client 0: testing1 testing2  
alksnjdjasnzdServer closed the connection.  
Select error: Bad file descriptor  
[wilsonsue@fedora Project3980]$
```