Give an overview of the the client-server program

The Client-Server program allows one or more clients to connect to a server and communicate in text to the server. The server will display the clients message in the terminal and send the client a confirmation message that the server has recieved the message. The server will also check for new connections, if a new client connects then all the other clients will be notified. The server can also blacklist certain client ip’s and prevent said ip’s from joining.

Describe how it works (at the high level, textual)

On the server side:

The server creates a socket which listens to connections from the server side, if a client wants to connect then the server will accept the connection using accept(). If the accept succeeds then the client recieves a welcome message. The server will then broadcast a message to all other clients that a new client has connected, finally the new client will be added to the set of active file descpritors in the server.

If an already connected client has a message for the server the server will recognize that the data has arrived on an already connected socket. It will then read said data, print it out in the console and send back a confirmation message to the client.

On the client side:

The client uses connect() in order to connect to the server, if connection is successful the client goes through the file descriptors and listens to inputs from either a socket or stdin. If a message is arriving from the server the client will receive it and print it out, else the client will wait for the user to type a message and send it to the server.

Show how to use the program

Setting up:

1. Open a terminal and go to the /DVA218/lab2_files directory.
2. Type "make clean && make"

Using server:

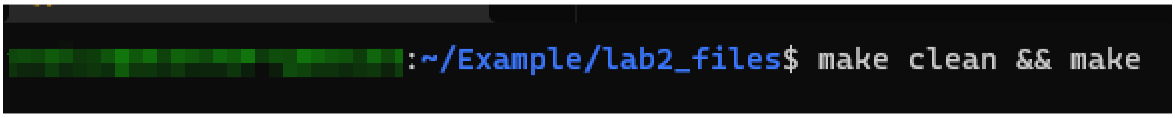
1. Enter command "./server" into the terminal to start the server.

Using client:

1. Enter command "./client {ip_address_of_server}" into the terminal, where the ip of the server that you want to connect to should replace te {ip_address_of_server} bracket.
2. You can now start sending messages to the server and will get a reply when the server has successfully received your message. You will also get updates when other client connect to the server.

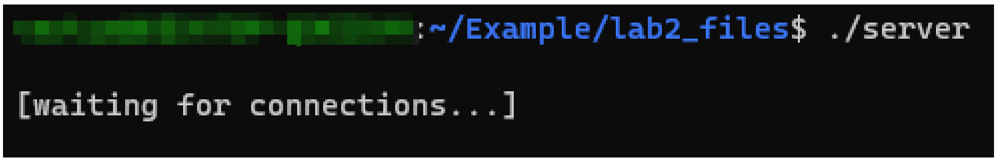
Example images of setting up and connecting:

1. Setting up files



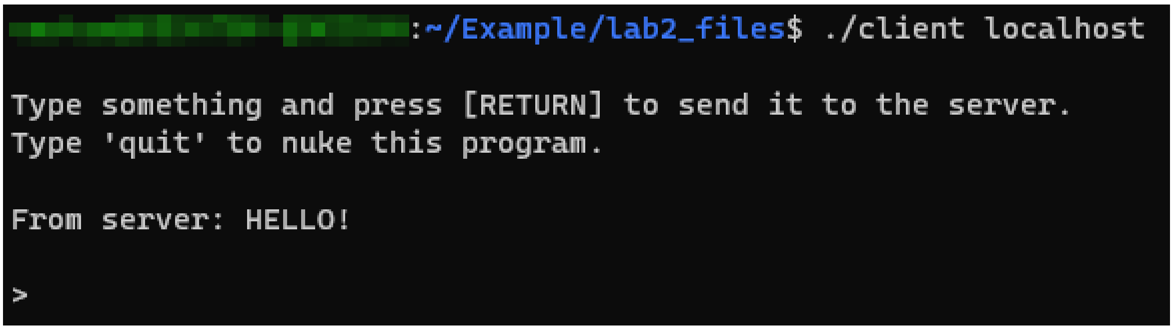
```
[redacted]:~/Example/lab2_files$ make clean && make
```

2. Starting server



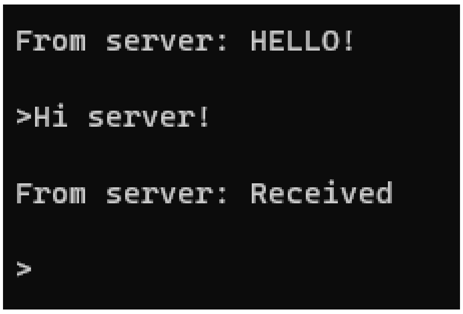
```
[redacted]:~/Example/lab2_files$ ./server

[waiting for connections...]
```

3. Connecting client - (using *localhost* as example)



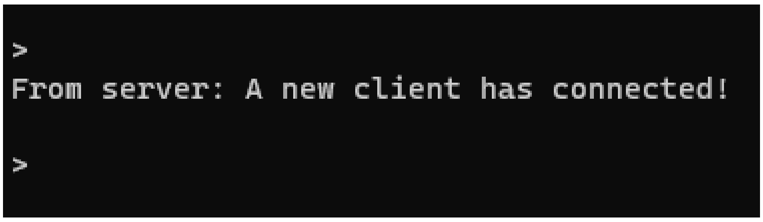
```
[redacted]:~/Example/lab2_files$ ./client localhost

Type something and press [RETURN] to send it to the server.
Type 'quit' to nuke this program.

From server: HELLO!

>
```

4. Sending a message



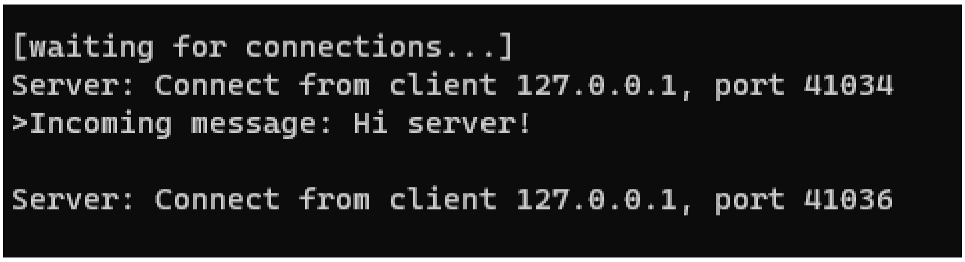
```
From server: HELLO!

>Hi server!

From server: Received

>
```

5. Notification if another client has connected



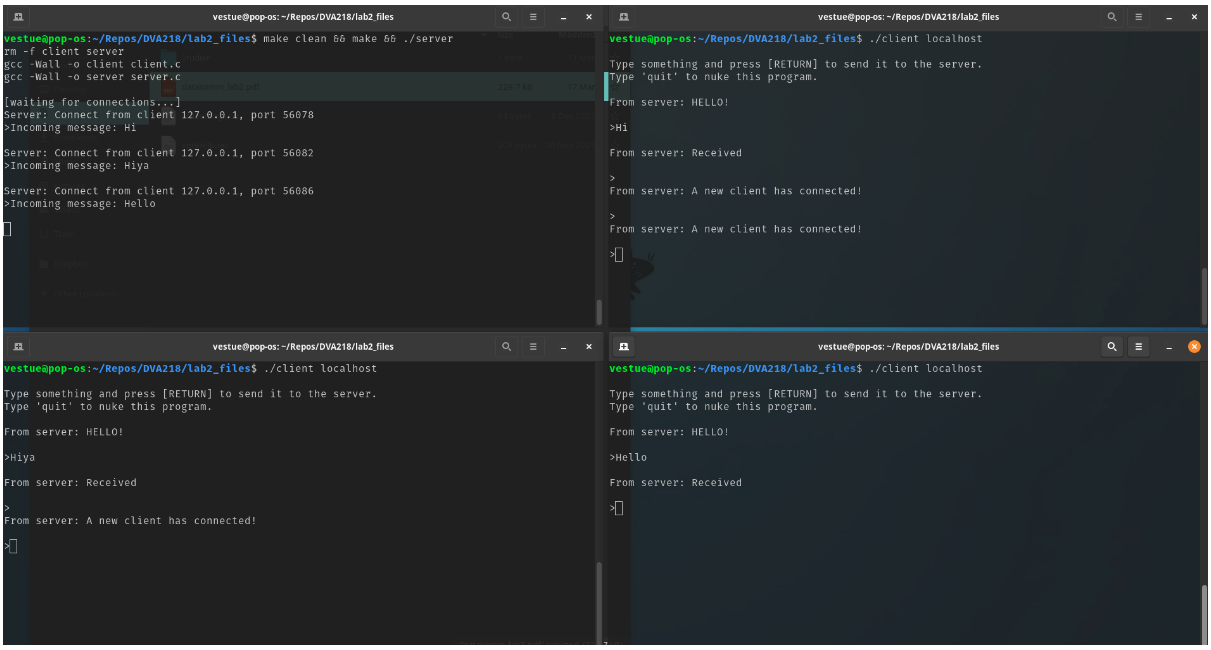
```
>
From server: A new client has connected!

>
```

6. How the connection is seen Server-side



```
[waiting for connections...]
Server: Connect from client 127.0.0.1, port 41034
>Incoming message: Hi server!

Server: Connect from client 127.0.0.1, port 41036
```

Showcase of connecting



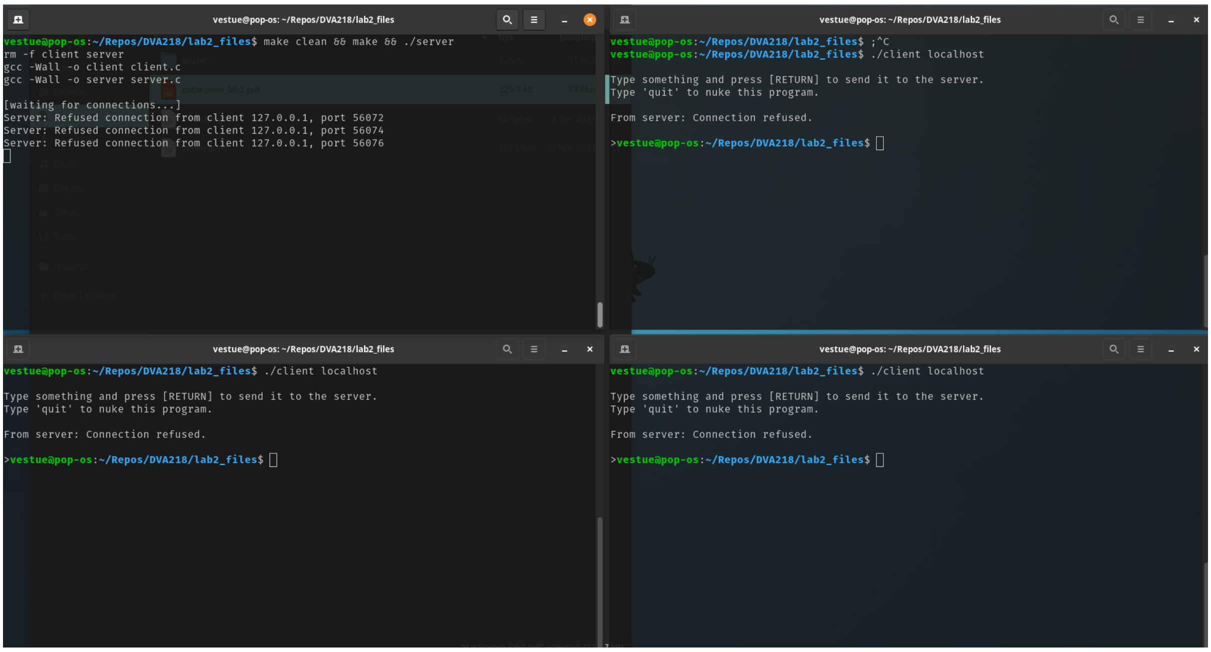
```
vestue@pop-os:~/Repos/DVA218/lab2_files$ make clean && make && ./server
rm -f client server
gcc -Wall -o client client.c
gcc -Wall -o server server.c

[waiting for connections...]
Server: Connect from client 127.0.0.1, port 56078
>Incoming message: Hi

Server: Connect from client 127.0.0.1, port 56082
>Incoming message: Hiya

Server: Connect from client 127.0.0.1, port 56086
>Incoming message: Hello
```

```
vestue@pop-os:~/Repos/DVA218/lab2_files$ ./client localhost

Type something and press [RETURN] to send it to the server.
Type 'quit' to nuke this program.

From server: HELLO!

>Hi

From server: Received

>
From server: A new client has connected!

>
From server: A new client has connected!

>
```

```
vestue@pop-os:~/Repos/DVA218/lab2_files$ ./client localhost

Type something and press [RETURN] to send it to the server.
Type 'quit' to nuke this program.

From server: HELLO!

>Hiya

From server: Received

>
From server: A new client has connected!

>
```

```
vestue@pop-os:~/Repos/DVA218/lab2_files$ ./client localhost

Type something and press [RETURN] to send it to the server.
Type 'quit' to nuke this program.

From server: HELLO!

>Hello

From server: Received

>
```

Connecting with a blocked client



vestue@pop-os: ~/Repos/DVA218/lab2_files
vestue@pop-os: ~/Repos/DVA218/lab2_files
vestue@pop-os: ~/Repos/DVA218/lab2_files
vestue@pop-os: ~/Repos/DVA218/lab2_files

```
vestue@pop-os:~/Repos/DVA218/lab2_files$ make clean && make && ./server
rm -f client server
gcc -Wall -o client client.c
gcc -Wall -o server server.c

[waiting for connections...]
Server: Refused connection from client 127.0.0.1, port 56072
Server: Refused connection from client 127.0.0.1, port 56074
Server: Refused connection from client 127.0.0.1, port 56076
```

```
vestue@pop-os:~/Repos/DVA218/lab2_files$ ;^C
vestue@pop-os:~/Repos/DVA218/lab2_files$ ./client localhost

Type something and press [RETURN] to send it to the server.
Type 'quit' to nuke this program.

From server: Connection refused.

>vestue@pop-os:~/Repos/DVA218/lab2_files$
```

```
vestue@pop-os:~/Repos/DVA218/lab2_files$ ./client localhost

Type something and press [RETURN] to send it to the server.
Type 'quit' to nuke this program.

From server: Connection refused.

>vestue@pop-os:~/Repos/DVA218/lab2_files$
```

```
vestue@pop-os:~/Repos/DVA218/lab2_files$ ./client localhost

Type something and press [RETURN] to send it to the server.
Type 'quit' to nuke this program.

From server: Connection refused.

>vestue@pop-os:~/Repos/DVA218/lab2_files$
```