Advance topics in software engineering

Server Client Implementation Using TCP Protocol

Group-A

Project User Manual

SUBMITTED TO:

Prof. Cristina Ruiz Martin

SUBMITTED BY:

Hitin Sarin – 101134217

Jaspreet Kaur – 101148265

Prerit Sikerwal – 101128282

Saksham Mal – 101132131

Installing GCC Compiler in Ubuntu:

Below are the commands for Installing gcc 7.2 and g++ 7.2 and making them default compiler in Ubuntu machine. Before proceeding with the installation please verify if the machine has the gcc compiler already installed in it. To check that you have g++-7 and gcc-7, Open Ubuntu terminal using "CONTROL+ALT+t" and type on the terminal the following command and press ENTER:

$ g++ -v

$ gcc -v

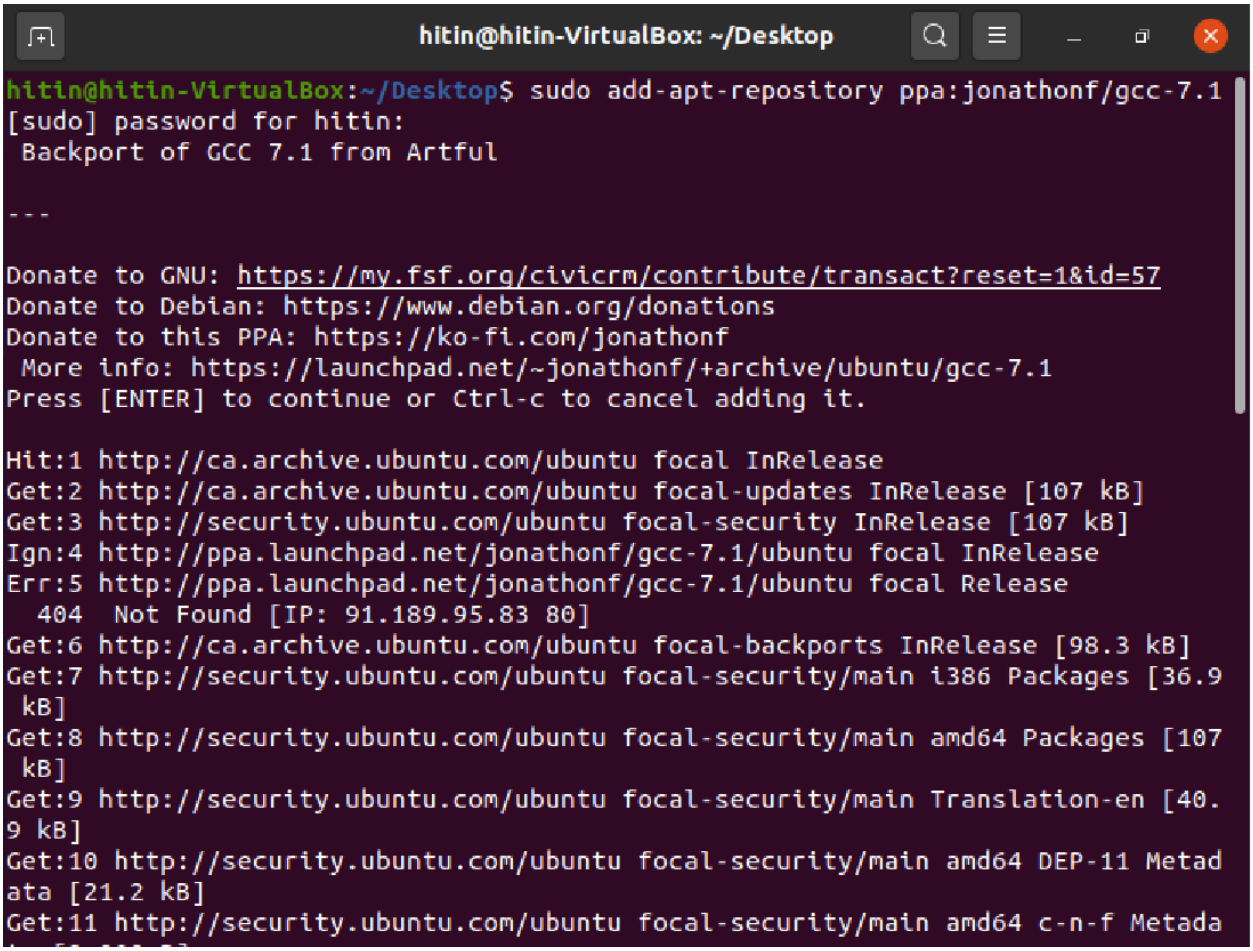
STEPS:

1. Open Ubuntu terminal using "CONTROL+ALT+t"

2. Type on the Ubuntu terminal the following command and press ENTER: (Type the administrative password)

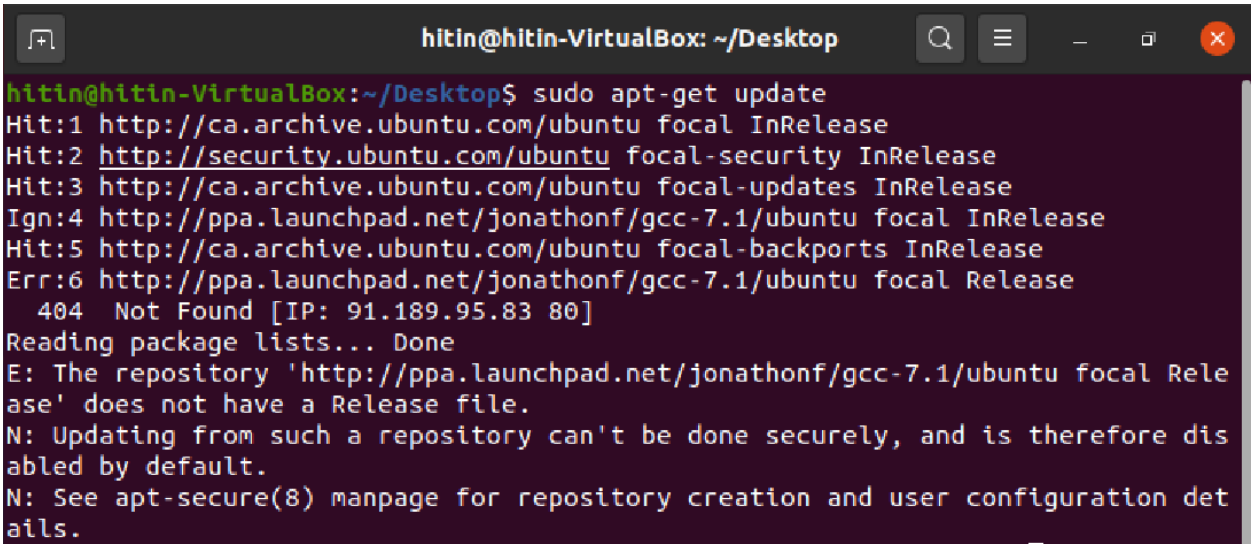
$ sudo add-apt-repository ppa:jonathonf/gcc-7.1

3. Press [ENTER] to continue.



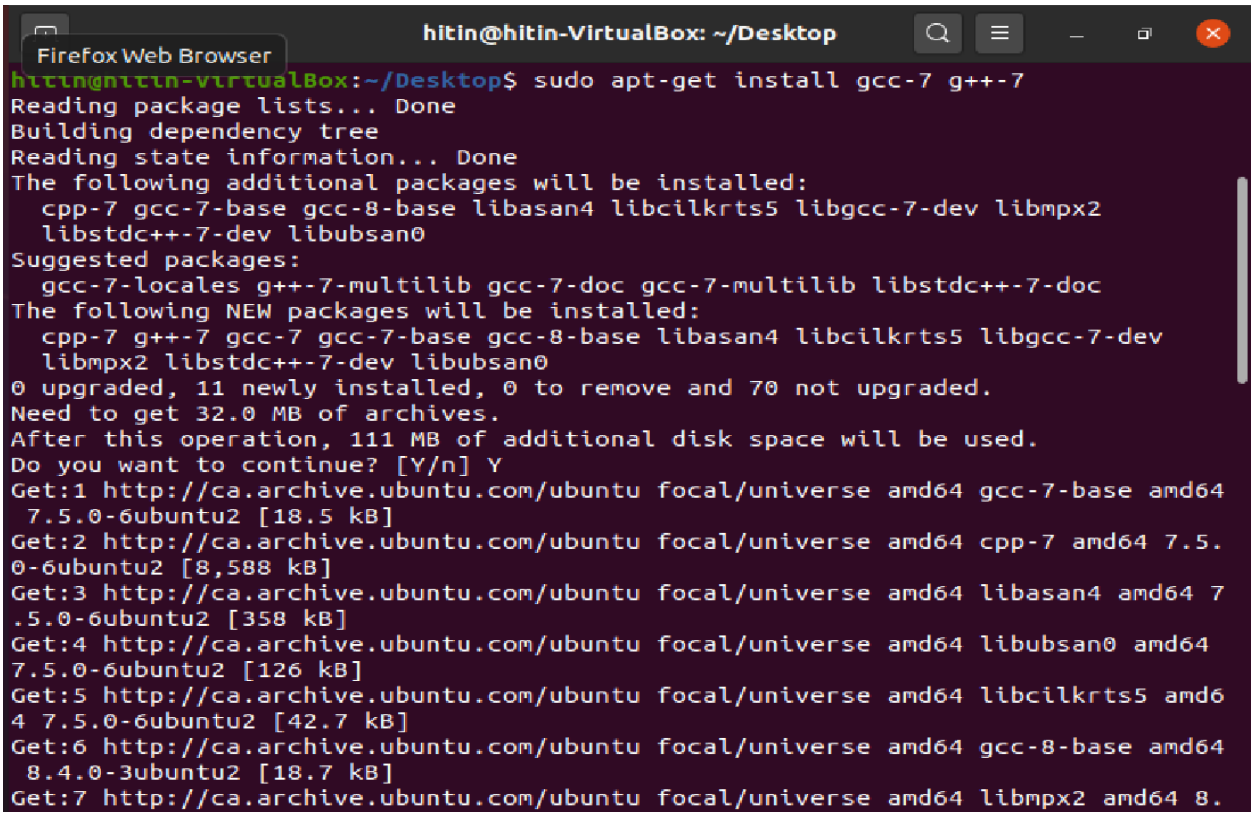
5. Type on the Ubuntu terminal the following command and press ENTER:

$ sudo apt-get update



6. Type on the terminal the following command and press ENTER:

$ sudo apt-get install gcc-7 g++-7 (Press Y if asked)



7. Type on the Ubuntu terminal the following command and press ENTER:

$ sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-7 60 --slave /usr/bin/gcc-ar gcc-ar /usr/bin/gcc-ar-7 --slave /usr/bin/gcc-nm gcc-nm /usr/bin/gcc-nm-7 --slave /usr/bin/gccranlib gcc-ranlib /usr/bin/gcc-ranlib-7

```
hitin@hitin-VirtualBox:~/Desktop$ sudo update-alternatives --install /usr/bin/g
cc gcc /usr/bin/gcc-7 60 --slave /usr/bin/gcc-ar gcc-ar /usr/bin/gcc-ar-7 --sla
ve /usr/bin/gcc-nm gcc-nm /usr/bin/gcc-nm-7 --slave /usr/bin/gccranlib gcc-ranl
ib /usr/bin/gcc-ranlib-7
update-alternatives: using /usr/bin/gcc-7 to provide /usr/bin/gcc (gcc) in auto
 mode
hitin@hitin-VirtualBox:~/Desktop$
```

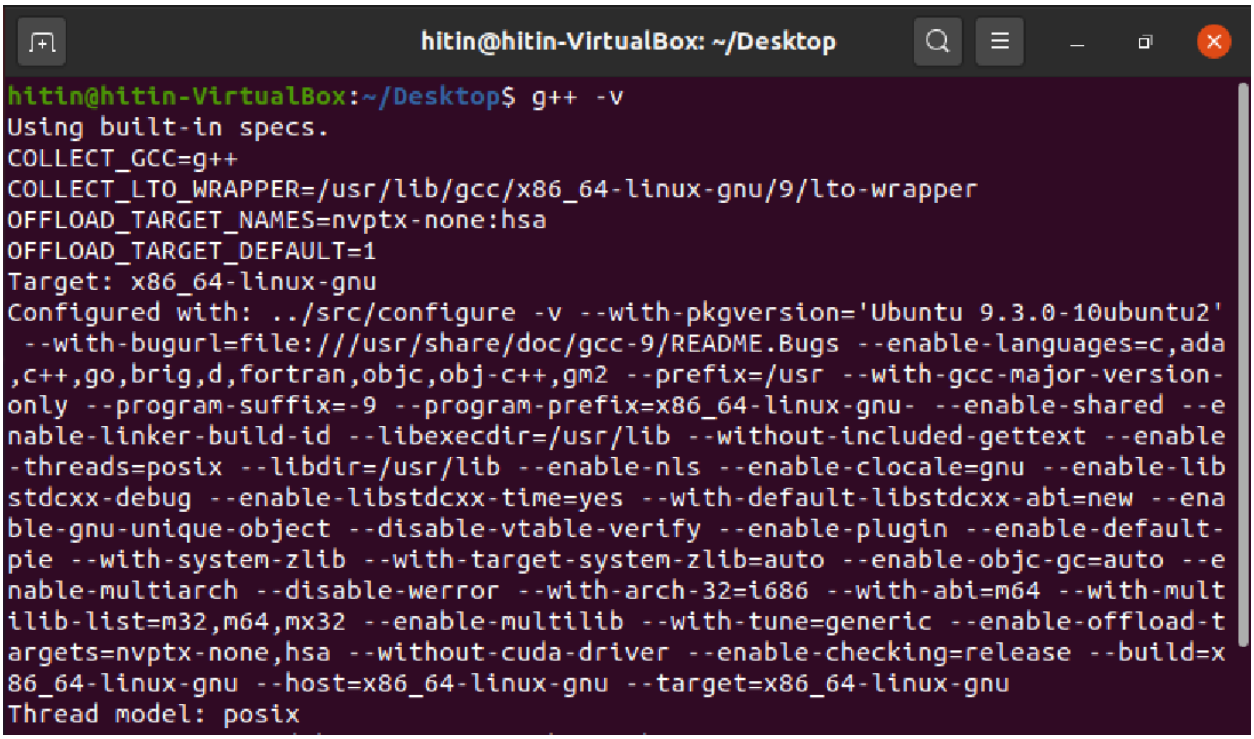
8. Type on the Ubuntu terminal the following command and press ENTER:

$ sudo update-alternatives --install /usr/bin/g++ g++ /usr/bin/g++-7 60 --slave /usr/bin/g++-ar g++-ar /usr/bin/g++-ar-7 --slave /usr/bin/g++-nm g++-nm /usr/bin/g++-nm-7 --slave /usr/bin/g++- ranlib g++-ranlib /usr/bin/g++-ranlib-7

Similarly, the above command is executed.

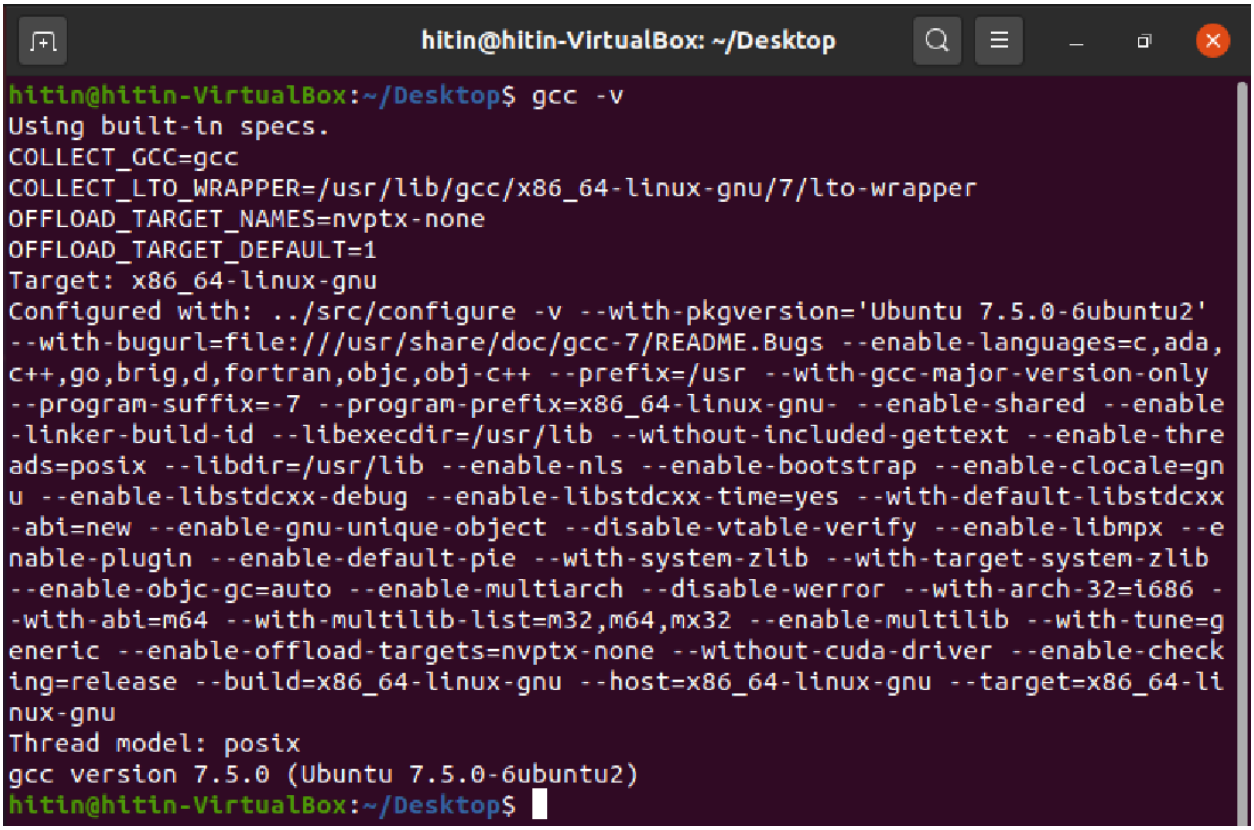
9. To check that you have g++-7 type on the terminal the following command and press ENTER:

$ g++ -v



10. To check that you have gcc-7 type on the terminal the following command and press ENTER:

$ gcc -v



Downloading Project:

1) Install Git (User can use Git bash or any other software of choice to clone the project).

2) Create a new folder for the project. Right Click inside the folder to open a git bash.

Right Click + “Git Bash here”.

3) Type the following command in the git terminal:

$ git clone https://github.com/jaskaur7/Group_A_SERVER-CLIENT_USING_TCP.git

OR

Go to the link https://github.com/jaskaur7/Group_A_SERVER-CLIENT_USING_TCP.git and click on Clone or Download button and then download the zip folder of the software into the machine.

Running the Project:

Compiling and running a C program

All the main code is defined under '/src directory', the header files are included under '/include directory' and the test files are under '/test directory'.

1. Open the downloaded folder (where project files are placed) cloned from github repository.

2. Set the prompt in the folder and then right click to open a new terminal. The terminal will be in the same directory as the project files.

3. Under the project path, a file named 'makefile' exists in which all the functions (client, serverSingle, serverMulti, test_client and test_server) have been compiled using the below set of commands.

```
================================================================================================
# ------------------------------------------------
# Generic Makefile
#
# @author Hitin Sarin - hitinsarin@cmail.carleton.ca
# @author Jaspreet Kaur - jaspreetkaur4@cmail.carleton.ca
# @author Saksham Mal - sakshammal@cmail.caretlon.ca
# @author Prerit Sikerwal - preritsikerwal@cmail.carleton.ca
#

# Date  : 2020-06-23
#
# Changelog :
# 2020-06-23: Makefile
#
# ------------------------------------------------

CC = gcc
LIBS = -pthread

all: client serverSingle serverMulti test_client test_server message

serverMulti: serverMulti.o queue.o common.o
		$(CC) $(LIBS) ./build/serverMulti.o ./build/queue.o ./build/common.o -o ./bin/serverMulti

serverSingle: serverSingle.o common.o
		$(CC) ./build/serverSingle.o ./build/common.o -o ./bin/serverSingle

client: client.o common.o
		$(CC) ./build/client.o ./build/common.o -o ./bin/client

test_client: test_client.o common.o
		$(CC) ./build/test_client.o ./build/common.o -o ./bin/test_client

test_server: test_server.o common.o
		$(CC) ./build/test_server.o ./build/common.o -o ./bin/test_server

serverMulti.o: ./src/server-multithread/server.c ./include/server_multi.h ./include/queue.h ./include/common.h
		$(CC) -c ./src/server-multithread/server.c -o ./build/serverMulti.o

serverSingle.o: ./src/server-singlethread/server.c  ./include/server_single.h ./include/common.h
		$(CC) -c ./src/server-singlethread/server.c -o ./build/serverSingle.o
```

```
client.o: ./src/client/client.c ./include/client.h ./include/common.h
		$(CC) -c ./src/client/client.c -o ./build/client.o

queue.o: ./lib/queue.c ./include/queue.h
		$(CC) -c ./lib/queue.c -o ./build/queue.o

test_client.o: ./test/src/test_client.c ./include/client.h ./include/common.h
	$(CC) -c ./test/src/test_client.c -o ./build/test_client.o

test_server.o: ./test/src/test_server.c ./include/server_single.h ./include/server_multi.h ./include/common.h
		$(CC) -c ./test/src/test_server.c -o ./build/test_server.o

common.o: ./src/common.c ./include/common.h
		$(CC) -c ./src/common.c -o ./build/common.o

message:
	@echo ""
	@echo "		Now you can run this project by folowing the steps in the User_Manual1.0."
	@echo "		bin/./client 127.0.0.1 8080"
	@echo "		bin/./serverSingle 8080"

.PHONY: clean

clean:
		rm -f ./build/*.o ./bin/serverSingle ./bin/serverMulti ./bin/client ./bin/test_client ./bin/test_server
==========================================================================================================
```

4. Now type the ‘make all’ command inside the terminal and press enter.

$ make all

```
hitin@hitin-VirtualBox:~/Desktop/ProjectC$ make all
gcc -c ./src/client/client.c -o ./build/client.o
gcc -c ./src/common.c -o ./build/common.o
gcc ./build/client.o ./build/common.o -o ./bin/client
gcc -c ./src/server-singlethread/server.c -o ./build/serverSingle.o
gcc ./build/serverSingle.o ./build/common.o -o ./bin/serverSingle
gcc -c ./src/server-multithreaded/server.c -o ./build/serverMulti.o
gcc -c ./lib/queue.c -o ./build/queue.o
gcc -pthread ./build/serverMulti.o ./build/queue.o ./build/common.o -o ./bin/se
rverMulti
gcc -c ./test/src/test_client.c -o ./build/test_client.o
gcc ./build/test_client.o ./build/common.o -o ./bin/test_client
gcc -c ./test/src/test_server.c -o ./build/test_server.o
gcc ./build/test_server.o ./build/common.o -o ./bin/test_server
```

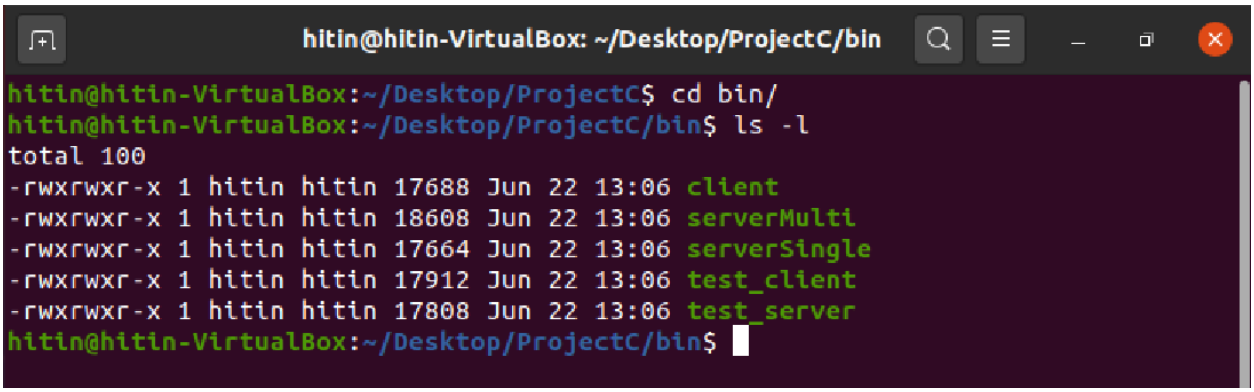
5. Once the ‘make all’ command is executed, switch to the ‘bin/ directory’ using the below command:

$ cd bin/ (This will change the directory to bin directory)

Now type the ‘list’ command using the syntax mentioned below:

$ ls -l (This will list the files under bin directory)

(P.S. The list command will show all the main files: client, serverSingle, serverMulti, test_client and test_server in the executable forms).

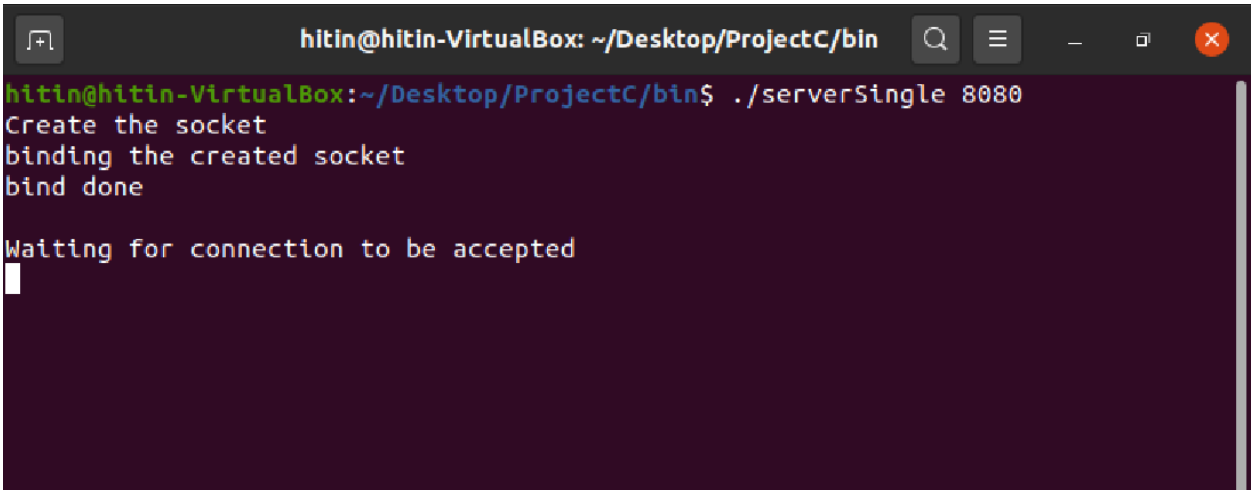


6. Now, to build the connection between Client and Single-thread Server using TCP protocol as proposed in Release-1, run the Single-thread Server file in one machine/tab and the client file in another machine/tab.

Step1 – On first machine/tab, run 'serverSingle' file with port number 8080 using below command:

$ ./serverSingle <Port Number>

After this command is executed, server is ready to accept connections after creating and binding the created socket.

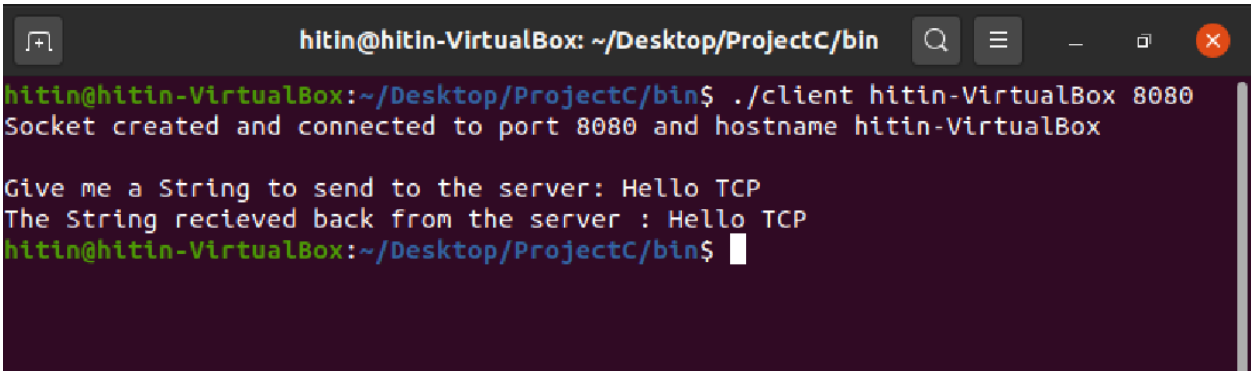


Step2- On second machine/tab, under the same 'bin/ directory', run client file with port number 8080 and hostname/IP address of the Single-thread Server using below command:

$ ./client <Hostname/IP Address> <Port Number>

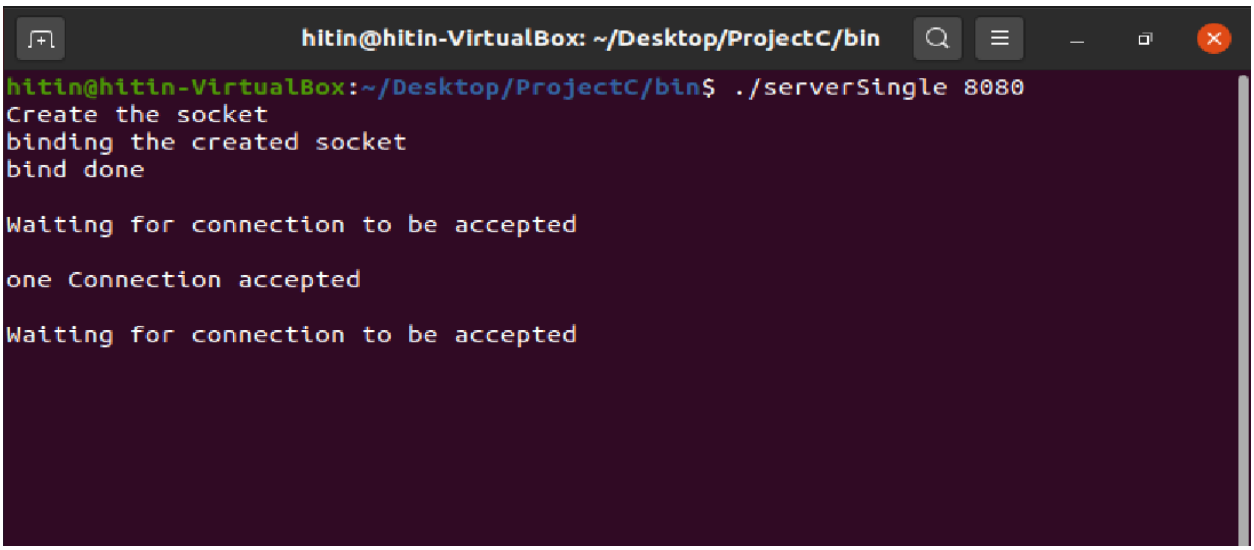
After executing the client file, a connection with the provided Server hostname/IP address is built prompting a message “Give me a string to send to the server:”

Once a message string is sent to the server, same string is received back from server stating the connection has been accepted by the server.



Meanwhile on the single-thread server side, a message stating “One Connection accepted” is prompted which is further waiting to accept a new connection.

P.S. The Single-thread server will take one connection request at a time.



7. Now, for executing the Multi-thread server with given number of threads and size of queue on Tab-1 and multiple clients on Tab-2, Tab-3 and Tab-4, follow the below mentioned steps.

Step1- On Tab-1 , run the Multi-thread server with the given number of threads and size of queue.

$ ./serverMulti <port number> <number of threads> <size of queue>

```
hitin@hitin-VirtualBox:~/Desktop/ProjectC/bin$ ./serverMulti 8080 10 10
Create the socket
Socket created
bind done
Thread 140372377753344:         Waiting for Connection
Thread 140372369360640:         Waiting for Connection
Thread 140372360967936:         Waiting for Connection
Thread 140372352575232:         Waiting for Connection
Thread 140372344182528:         Waiting for Connection
Thread 140372327397120:         Waiting for Connection
Thread 140372302219008:         Waiting for Connection
Thread 140372319004416:         Waiting for Connection
Thread 140372335789824:         Waiting for Connection
Thread 140372310611712:         Waiting for Connection

Main thread is Accepting Connections
```

Step2 – Run the client file on multiple tabs for connecting the multi-thread server simultaneously using below command:

$ ./client <hostname/IP address> <port number>

As shown below, different connection requests (with different string messages) have been sent to the multi-thread server.

Tab-2

```
hitin@hitin-VirtualBox: ~/Desktop/ProjectC/bin
hitin@hitin-VirtualBox:~/Desktop/ProjectC/bin$ ./client hitin-VirtualBox 8080
Socket created and connected to port 8080 and hostname hitin-VirtualBox

Give me a String to send to the server: hello
The String recieved back from the server : hello
hitin@hitin-VirtualBox:~/Desktop/ProjectC/bin$
```

Tab-3

```
hitin@hitin-VirtualBox:~/Desktop/ProjectC/bin$ ./client hitin-VirtualBox 8080
Socket created and connected to port 8080 and hostname hitin-VirtualBox

Give me a String to send to the server: hi
The String recieved back from the server : hi
hitin@hitin-VirtualBox:~/Desktop/ProjectC/bin$
```

Tab-4

```
hitin@hitin-VirtualBox:~/Desktop/ProjectC/bin$ ./client hitin-VirtualBox 8080
Socket created and connected to port 8080 and hostname hitin-VirtualBox

Give me a String to send to the server: Hey
The String recieved back from the server : Hey
hitin@hitin-VirtualBox:~/Desktop/ProjectC/bin$
```

Meanwhile on the multi-thread server-end, multiple client requests from the above-mentioned tabs have been completed prompting a message “Connection completed”.

The server then waits for the completion of pending connections.

```
hitin@hitin-VirtualBox:~/Desktop/ProjectC/bin$ ./serverMulti 8080 10 10
Create the socket
Socket created
bind done
Thread 140372377753344:         Waiting for Connection
Thread 140372369360640:         Waiting for Connection
Thread 140372360967936:         Waiting for Connection
Thread 140372352575232:         Waiting for Connection
Thread 140372344182528:         Waiting for Connection
Thread 140372327397120:         Waiting for Connection
Thread 140372302219008:         Waiting for Connection
Thread 140372319004416:         Waiting for Connection
Thread 140372335789824:         Waiting for Connection
Thread 140372310611712:         Waiting for Connection

Main thread is Accepting Connections


Connection Completed


Main thread is Accepting Connections
Handler 140372377753344: Processing :

Connection Completed
```

```
Main thread is Accepting Connections
Handler 140372377753344: Processing :

Connection Completed


Main thread is Accepting Connections
Handler 140372369360640: Processing :

Connection Completed


Main thread is Accepting Connections
Handler 140372360967936: Processing :Thread 140372360967936:    Waiting for Con
nection
Thread 140372369360640:         Waiting for Connection
Thread 140372377753344:         Waiting for Connection
```