**GUESS THE NUMBER GAME**

**TCP Server**

import socket

import sys

import signal

import time

import random

server\_running = True

# CTRL+C shutdown command signal handle

def handle\_shutdown\_signal(signal, frame):

global server\_running

print("Shutting down the server...")

server\_running = False

time.sleep(3)

sys.exit(0)

def main():

# register shutdown signal handler for CTRL+C

signal.signal(signal.SIGINT, handle\_shutdown\_signal)

# create TCP socket and bind the server to ip and port

try:

server = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

server.bind(('127.0.0.1', 8080))

server.listen(5)

print("Server waiting and waiting...")

# loop while server is up

while server\_running:

secret\_number = random.randint(1, 100)

# timeout server after 5 minutes if no interraction from client

server.settimeout(60 \* 5)

# accept and handle client connection

try:

client, addr = server.accept()

except socket.timeout:

print("Connection timed out.")

break

print(f"Connection accepted: {addr[0]}:{addr[1]}")

# solve the problem with data from client

response = "Number Guessing Game! Guess a number between 1 and 100."

client.send(response.encode('utf-8'))

print(f'Number to be guessed - {secret\_number}')

# loop for values until you guess

while True:

# receive the guess from the client

guess = client.recv(1024).decode('utf-8')

print(f'client guess: {guess}')

if not guess.isnumeric():

response = "ERROR!!! NEED NUMBER!!!"

print("invalid client input")

continue

# if number guessed we out

if int(guess) == secret\_number:

print(int(guess))

print(int(guess) == secret\_number)

response = "YAAAY! YA DID IT CHAMPION!"

break

# checking client to server data and decoding the data to the correct format

if int(guess) < secret\_number:

response = "higher!"

else:

response = "LOWER!"

client.send(response.encode('utf-8'))

client.send(response.encode('utf-8'))

client.close()

except Exception as e:

raise

finally:

server.close()

print("Server closed.")

main()

**TCP Client**

#include <iostream>

#include <string>

#include <cstring>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <unistd.h>

int main()

{

// create TCP socket and init the server socket snippet

int clientSocket = socket(AF\_INET, SOCK\_STREAM, 0);

if (clientSocket == -1)

{

std::cerr << "Error on creating socket.\n";

return 1;

}

sockaddr\_in serverAddress;

serverAddress.sin\_family = AF\_INET;

serverAddress.sin\_port = htons(8080);

serverAddress.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

// server connection

if (connect(clientSocket, (struct sockaddr\*)&serverAddress, sizeof(serverAddress)) == -1)

{

std::cerr << "Server connection error.\n";

return 1;

}

while (true)

{

// allocating memory so it doesnt kill server

char buffer[1024];

memset(buffer, 0, sizeof(buffer));

int bytesReceived = recv(clientSocket, buffer, sizeof(buffer), 0);

if (bytesReceived == -1)

{

std::cerr << "Error receiving data from server.\n";

return 1;

}

std::cout << "Server: " << buffer << "\n";

if (strstr(buffer, "YAAAY! YA DID IT CHAMPION!") != nullptr)

break;

// user input

std::string guess;

std::cout << "Your guess: ";

std::getline(std::cin, guess);

// send data to server

if (send(clientSocket, guess.c\_str(), guess.size(), 0) == -1)

{

std::cerr << "Error sending data to server.\n";

return 1;

}

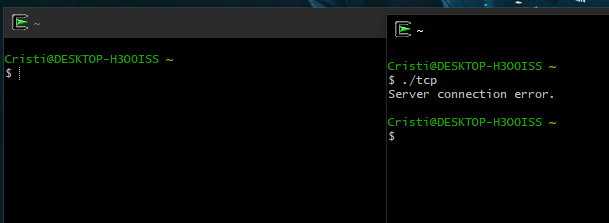
}

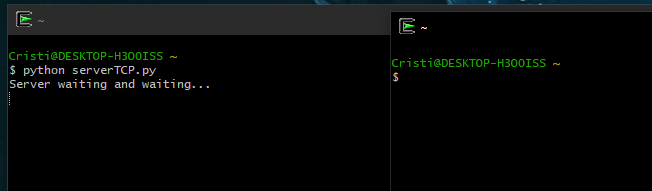
close(clientSocket);

return 0;

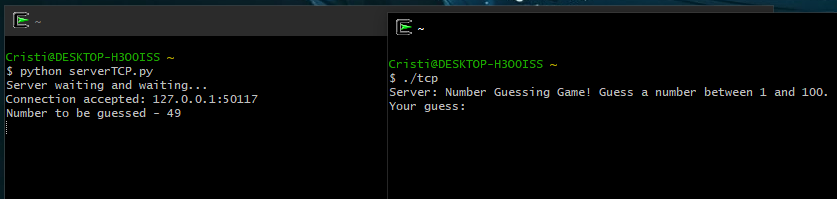
}

Server error since the server is not opened.

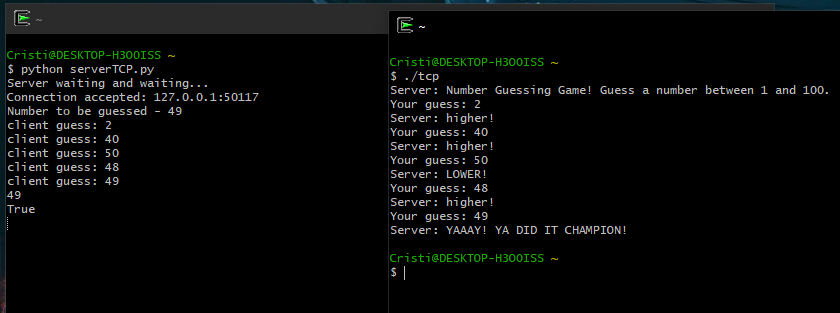
****

****

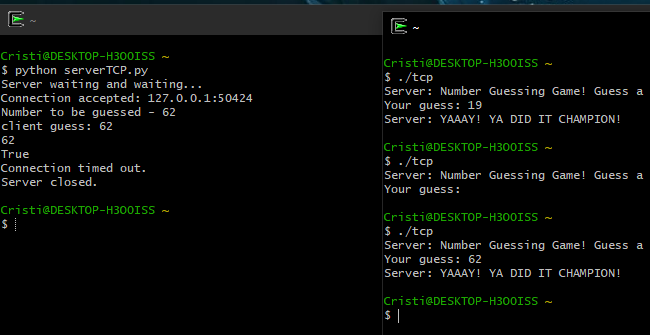
Server opened and the connection is accepted.

****

The game is working. We guessed the number.

****

Connection timed out after 5 minutes(forgot to show the time)

****

**UDP Server**

import socket

import sys

import signal

import time

import random

server\_running = True

# CTRL+C shutdown command signal handle

def handle\_shutdown\_signal(signal, frame):

global server\_running

print("Shutting down the server...")

server\_running = False

time.sleep(3)

sys.exit(0)

def main():

# register shutdown signal handler for CTRL+C

signal.signal(signal.SIGINT, handle\_shutdown\_signal)

# create UDP socket and bind the server to ip and port

try:

server = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

server.bind(('127.0.0.1', 8080))

print("Server waiting and waiting...")

# loop while server is up

while server\_running:

secret\_number = random.randint(1, 100)

# timeout server after 5 minutes if no interraction from client

server.settimeout(60 \* 5)

# wait for "start" message from the client

try:

data, addr = server.recvfrom(1024)

except socket.timeout:

print("Connection timed out.")

break

if data.decode('utf-8') == "start":

print(f"Connection accepted: {addr[0]}:{addr[1]}")

response = "Number Guessing Game! Guess a number between 1 and 100."

server.sendto(response.encode('utf-8'), addr)

print(f'Number to be guessed - {secret\_number}')

else:

print(f"Invalid initial message received from {addr[0]}:{addr[1]}")

# loop for values until you guess

while True:

# receive the guess from the client

try:

data, addr = server.recvfrom(1024)

guess = data.decode('utf-8')

except socket.timeout:

print("Client connection timed out.")

break

print(f'client guess: {guess}')

if not guess.isnumeric():

response = "ERROR!!! NEED NUMBER!!!"

print("Invalid client input")

else:

if int(guess) == secret\_number:

response = "YAAAY! YA DID IT CHAMPION!"

server.sendto(response.encode('utf-8'), addr)

break

elif int(guess) < secret\_number:

response = "higher!"

else:

response = "LOWER!"

server.sendto(response.encode('utf-8'), addr)

except Exception as e:

print("Error:", e)

finally:

server.close()

print("Server closed.")

main()

**UDP Client**

#include <iostream>

#include <string>

#include <cstring>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <unistd.h>

int main()

{

// create UDP socket and init the server socket snippet

int clientSocket = socket(AF\_INET, SOCK\_DGRAM, 0);

if (clientSocket == -1)

{

std::cerr << "Error on creating socket.\n";

return 1;

}

sockaddr\_in serverAddress;

serverAddress.sin\_family = AF\_INET;

serverAddress.sin\_port = htons(8080);

serverAddress.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

// send a "start" message to server to init the game

std::string message = "start";

if (sendto(clientSocket, message.c\_str(), message.size(), 0, (struct sockaddr\*)&serverAddress, sizeof(serverAddress)) == -1) {

std::cerr << "Error sending data to server.\n";

return 1;

}

while (true)

{

// allocating memory so it doesnt kill server…and receive data from server

char buffer[1024];

memset(buffer, 0, sizeof(buffer));

socklen\_t serverAddrLen = sizeof(serverAddress);

int bytesReceived = recvfrom(clientSocket, buffer, sizeof(buffer), 0, (struct sockaddr\*)&serverAddress, &serverAddrLen);

if (bytesReceived == -1)

{

std::cerr << "Error receiving data from server.\n";

return 1;

}

std::cout << "Server: " << buffer << "\n";

if (strstr(buffer, "YAAAY! YA DID IT CHAMPION!") != nullptr)

break;

// user input

std::string guess;

std::cout << "Your guess: ";

std::getline(std::cin, guess);

// send data to server

if (sendto(clientSocket, guess.c\_str(), guess.size(), 0, (struct sockaddr\*)& serverAddress, sizeof(serverAddress)) == -1)

{

std::cerr << "Error sending data to server.\n";

return 1;

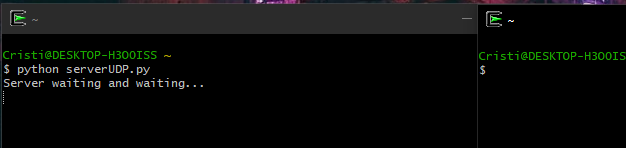
}

}

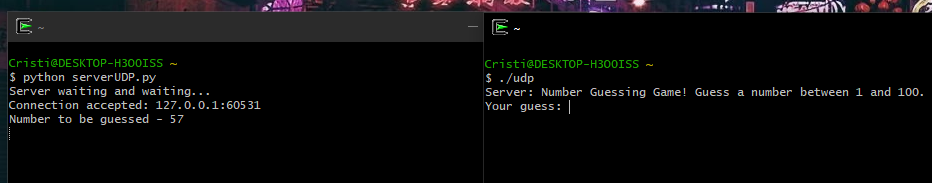
close(clientSocket);

return 0;

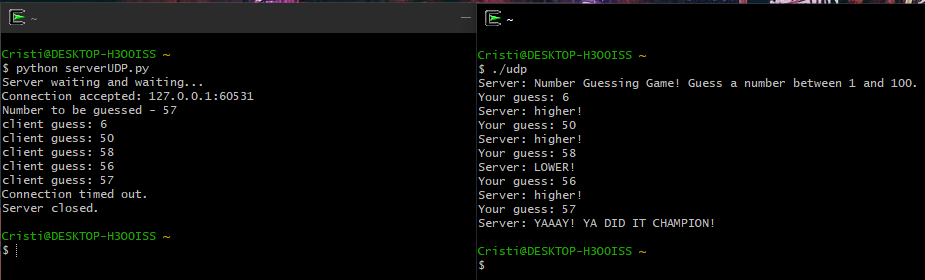
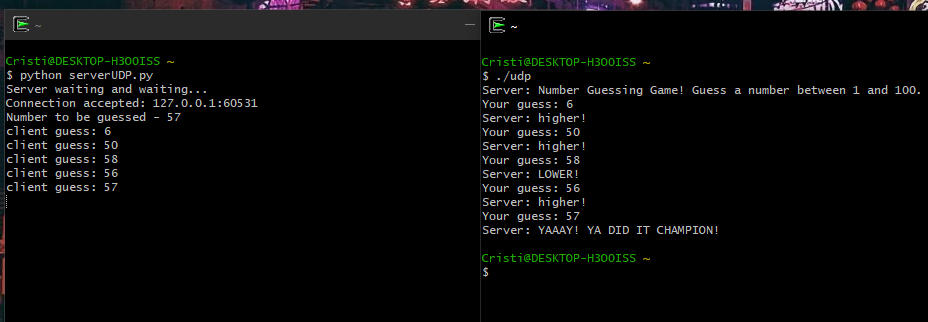
}

****

Server running and the connection is established.

****

Connection timed out after 5 minutes.

****

Game working as intended. We guessed the number.