National University of Computer and Emerging Sciences



Lab # 02

# Task 01

## server.c

#include <iostream>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <unistd.h>

#include <cstring>

using namespace std;

int main()

{

char msg[100];

int conn\_sock, comm\_sock, n;

struct sockaddr\_in server\_addr, client\_addr;

socklen\_t client\_addr\_len = sizeof(client\_addr);

// Create socket

conn\_sock = socket(AF\_INET, SOCK\_STREAM, 0);

// Set up server address structure

server\_addr.sin\_family = AF\_INET;

server\_addr.sin\_port = htons(1234); // Use port 1234

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

// Bind the socket

int bind\_result = bind(conn\_sock, (struct sockaddr\*)&server\_addr, sizeof(server\_addr));

// Listen for incoming connections

listen(conn\_sock, 10);

// Accept a connection

client\_addr\_len = sizeof(client\_addr);

comm\_sock = accept(conn\_sock, (struct sockaddr \*)&client\_addr, &client\_addr\_len);

if (comm\_sock < 0)

{

printf("Accept failed.\n");

}

else

{

printf("Connection established with client.\n");

}

// Close sockets

close(comm\_sock);

close(conn\_sock);

return 0;

}

## client.c

#include <arpa/inet.h>

#include <stdio.h>

#include <string.h>

#include <sys/socket.h>

#include <unistd.h>

#define PORT 1234

int main(int argc, char const\* argv[])

{

int status, valread, client\_fd;

struct sockaddr\_in serv\_addr;

client\_fd = socket(AF\_INET, SOCK\_STREAM, 0);

serv\_addr.sin\_family = AF\_INET;

serv\_addr.sin\_port = htons(PORT);

// Convert IPv4 and IPv6 addresses from text to binary form

int addr\_result = inet\_pton(AF\_INET, "127.0.0.1", &serv\_addr.sin\_addr);

status = connect(client\_fd, (struct sockaddr\*)&serv\_addr, sizeof(serv\_addr));

if (status < 0)

{

printf("Connection failed.\n");

}

else

{

printf("Connection successfull.\n");

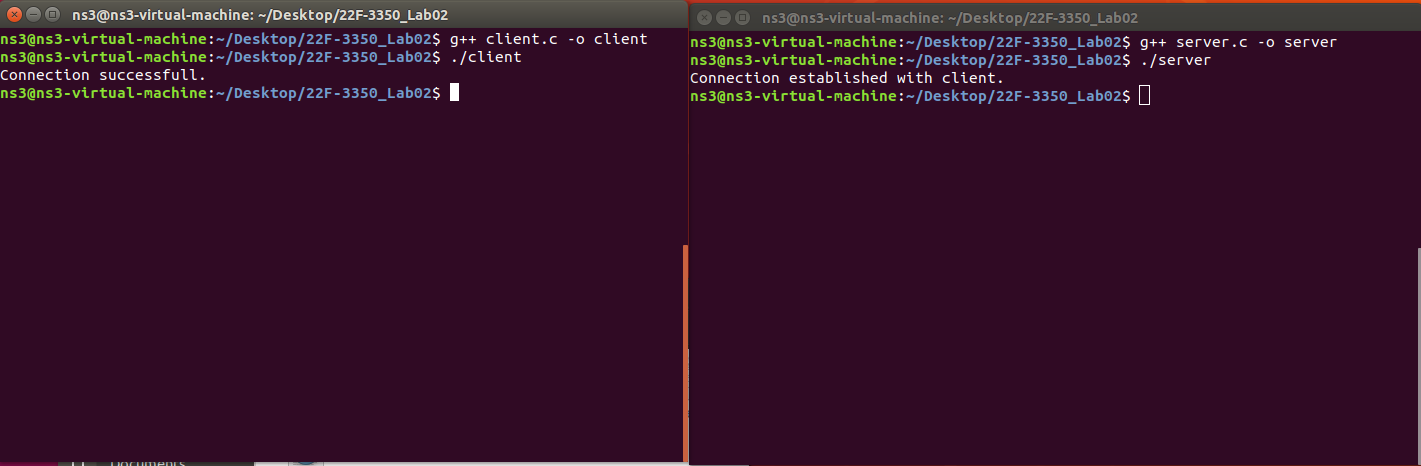
}

close(client\_fd);

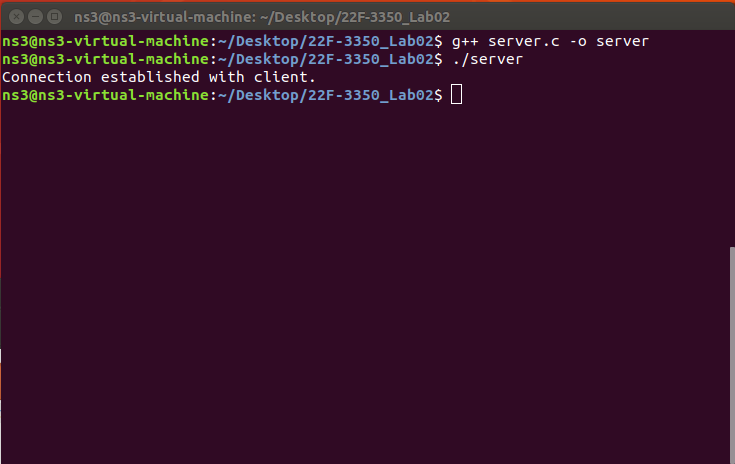
return 0;

}

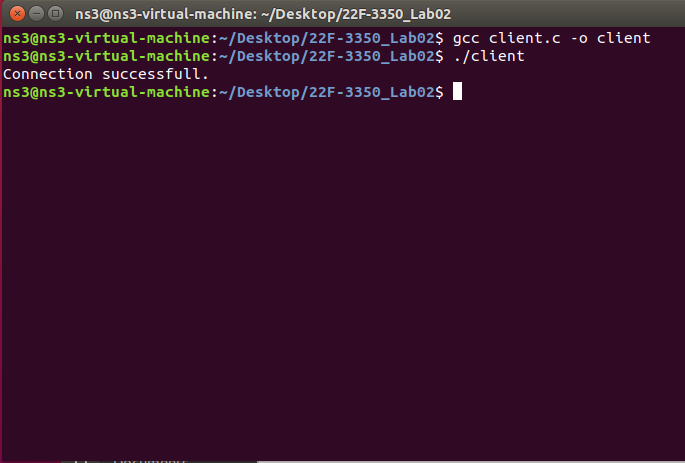
## Output (Both):



## Server:



## Client:



# Task 02

## server.c

#include <iostream>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <unistd.h>

#include <cstring>

#include <ctime>

using namespace std;

int main() {

char buffer[1024];

int conn\_sock, comm\_sock;

struct sockaddr\_in server\_addr, client\_addr;

socklen\_t client\_addr\_len = sizeof(client\_addr);

// Create socket

conn\_sock = socket(AF\_INET, SOCK\_STREAM, 0);

if (conn\_sock < 0) {

cerr << "Socket creation failed." << endl;

return -1;

}

// Set up server address structure

server\_addr.sin\_family = AF\_INET;

server\_addr.sin\_port = htons(1234);

server\_addr.sin\_addr.s\_addr = INADDR\_ANY; // Listen on all interfaces

// Bind the socket

if (bind(conn\_sock, (struct sockaddr\*)&server\_addr, sizeof(server\_addr)) < 0) {

cerr << "Bind failed." << endl;

close(conn\_sock);

return -1;

}

// Listen for incoming connections

listen(conn\_sock, 5);

cout << "Server is listening on port 1234..." << endl;

// Accept a connection

client\_addr\_len = sizeof(client\_addr);

comm\_sock = accept(conn\_sock, (struct sockaddr \*)&client\_addr, &client\_addr\_len);

if (comm\_sock < 0) {

cerr << "Accept failed." << endl;

close(conn\_sock);

return -1;

}

cout << "Connection established with client." << endl;

while (true) {

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

int n = recv(comm\_sock, buffer, sizeof(buffer) - 1, 0);

if (n <= 0) {

cout << "Client disconnected." << endl;

break;

}

buffer[n] = '\0'; // Null-terminate the received string

cout << "Received: " << buffer << endl;

if (strcmp(buffer, "HELLO") == 0) {

const char\* response = "Hello, Client!";

send(comm\_sock, response, strlen(response), 0);

} else if (strcmp(buffer, "TIME") == 0) {

time\_t now = time(0);

char\* dt = ctime(&now);

send(comm\_sock, dt, strlen(dt), 0);

} else if (strcmp(buffer, "EXIT") == 0) {

const char\* response = "Goodbye!";

send(comm\_sock, response, strlen(response), 0);

break;

} else {

const char\* response = "Unknown command.";

send(comm\_sock, response, strlen(response), 0);

}

}

// Close sockets

close(comm\_sock);

close(conn\_sock);

return 0;

}

## client.c

#include <arpa/inet.h>

#include <stdio.h>

#include <string.h>

#include <sys/socket.h>

#include <unistd.h>

#define PORT 1234

int main() {

int client\_fd;

struct sockaddr\_in serv\_addr;

char buffer[1024];

// Create socket

client\_fd = socket(AF\_INET, SOCK\_STREAM, 0);

if (client\_fd < 0) {

perror("Socket creation failed");

return -1;

}

serv\_addr.sin\_family = AF\_INET;

serv\_addr.sin\_port = htons(PORT);

// Convert IPv4 and IPv6 addresses from text to binary form

if (inet\_pton(AF\_INET, "127.0.0.1", &serv\_addr.sin\_addr) <= 0)

{

perror("Invalid address");

return -1;

}

// Connect to the server

if (connect(client\_fd, (struct sockaddr\*)&serv\_addr, sizeof(serv\_addr)) < 0) {

perror("Connection failed");

return -1;

}

printf("Connected to the server.\n");

while (1) {

printf("Enter command (HELLO, TIME, EXIT): ");

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

fgets(buffer, sizeof(buffer), stdin);

buffer[strcspn(buffer, "\n")] = 0; // Remove newline character

// Send command to server

send(client\_fd, buffer, strlen(buffer), 0);

// Receive response from server

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

int valread = recv(client\_fd, buffer, sizeof(buffer) - 1, 0);

if (valread > 0) {

buffer[valread] = '\0'; // Null-terminate the received string

printf("Server: %s\n", buffer);

}

if (strcmp(buffer, "Goodbye!") == 0) {

break; // Exit if the server says goodbye

}

}

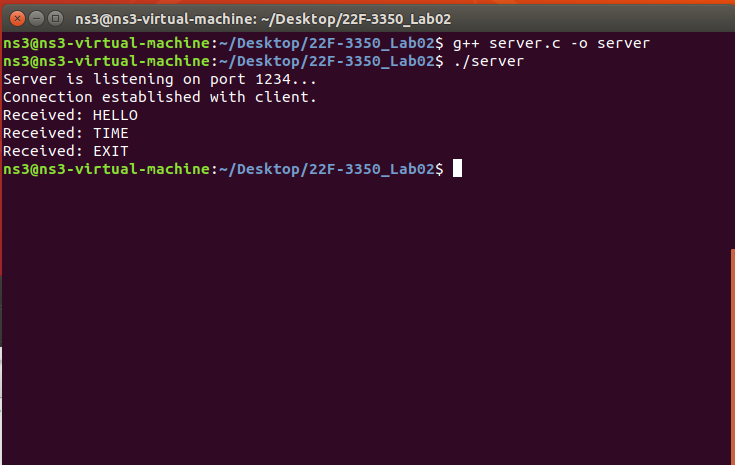
// Close socket

close(client\_fd);

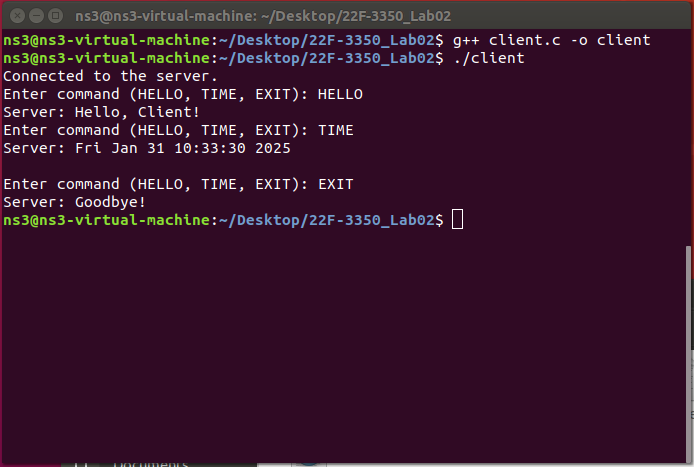
return 0;

}

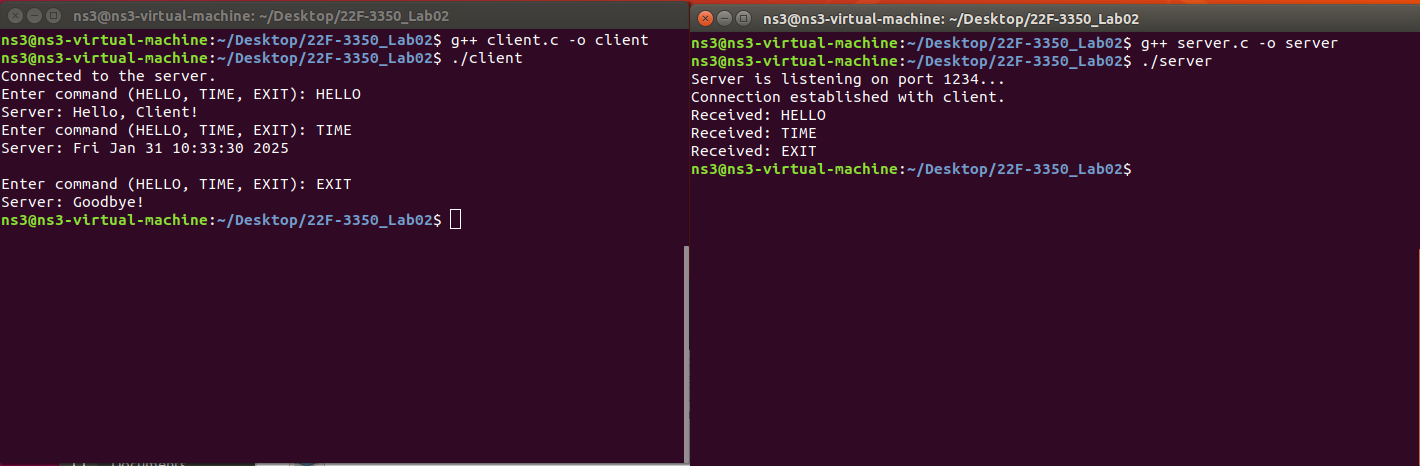
## Server:



## Client:



## Output (Both):



# Task 03

## server.c

#include <iostream>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <unistd.h>

#include <cstring>

#include <fstream>

using namespace std;

int main()

{

char filename[100];

int block\_size;

int conn\_sock, comm\_sock;

struct sockaddr\_in server\_addr, client\_addr;

socklen\_t client\_addr\_len = sizeof(client\_addr);

// Create socket

conn\_sock = socket(AF\_INET, SOCK\_STREAM, 0);

if (conn\_sock < 0) {

perror("Socket creation failed");

return -1;

}

// Set up server address structure

server\_addr.sin\_family = AF\_INET;

server\_addr.sin\_port = htons(1234);

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

// Bind the socket

int bind\_result = bind(conn\_sock, (struct sockaddr\*)&server\_addr, sizeof(server\_addr));

if (bind\_result < 0) {

perror("Bind failed");

return -1;

}

// Listen for incoming connections

listen(conn\_sock, 10);

// Accept a connection

client\_addr\_len = sizeof(client\_addr);

comm\_sock = accept(conn\_sock, (struct sockaddr \*)&client\_addr, &client\_addr\_len);

if (comm\_sock < 0) {

printf("Accept failed.\n");

return -1;

} else {

printf("Connection established with client.\n");

}

// Receive filename and block size from the client

recv(comm\_sock, filename, sizeof(filename), 0);

recv(comm\_sock, &block\_size, sizeof(block\_size), 0);

// Print the requested filename

cout << "Requested filename: " << filename << endl;

// Open the file

ifstream file(filename);

if (!file) {

perror("File not found or cannot be opened");

const char\* msg = "File not found.\n";

send(comm\_sock, msg, strlen(msg), 0);

} else {

char \*buffer = new char[block\_size];

while (file.read(buffer, block\_size) || file.gcount() > 0) {

int bytes\_sent = file.gcount();

send(comm\_sock, buffer, bytes\_sent, 0);

cout << "Sending block of size: " << bytes\_sent << endl; // Print the size of data sent

}

delete[] buffer;

file.close();

const char\* msg = "\nFile transfer complete.\n";

send(comm\_sock, msg, strlen(msg), 0);

}

// Close sockets

close(comm\_sock);

close(conn\_sock);

return 0;

}

## client.c

#include <arpa/inet.h>

#include <stdio.h>

#include <string.h>

#include <sys/socket.h>

#include <unistd.h>

#include <stdlib.h> // Include this header for atoi

#define PORT 1234

int main(int argc, char const\* argv[])

{

int status, client\_fd;

struct sockaddr\_in serv\_addr;

if (argc != 3) {

printf("Usage: %s <filename> <block\_size>\n", argv[0]);

return -1;

}

const char\* filename = argv[1];

int block\_size = atoi(argv[2]); // Convert block size from string to int

client\_fd = socket(AF\_INET, SOCK\_STREAM, 0);

if (client\_fd < 0) {

perror("Socket creation failed");

return -1;

}

serv\_addr.sin\_family = AF\_INET;

serv\_addr.sin\_port = htons(PORT);

// Convert IPv4 and IPv6 addresses from text to binary form

if (inet\_pton(AF\_INET, "127.0.0.1", &serv\_addr.sin\_addr) <= 0) {

perror("Invalid address/ Address not supported");

return -1;

}

status = connect(client\_fd, (struct sockaddr\*)&serv\_addr, sizeof(serv\_addr));

if (status < 0) {

printf("Connection failed.\n");

return -1;

} else {

printf("Connection successful.\n");

}

// Send filename and block size to the server

send(client\_fd, filename, strlen(filename), 0);

send(client\_fd, &block\_size, sizeof(block\_size), 0);

// Receive file content in blocks

char buffer[block\_size + 1];

int bytes\_received;

printf("Receiving file content:\n");

while ((bytes\_received = recv(client\_fd, buffer, block\_size, 0)) > 0) {

buffer[bytes\_received] = '\0'; // Null-terminate the buffer

printf("%s", buffer); // Print the received data

}

if (bytes\_received < 0) {

perror("Receive failed");

} else if (bytes\_received == 0) {

printf("No more data received from the server.\n");

}

printf("\nFile transfer complete.\n"); // Indicate completion

close(client\_fd);

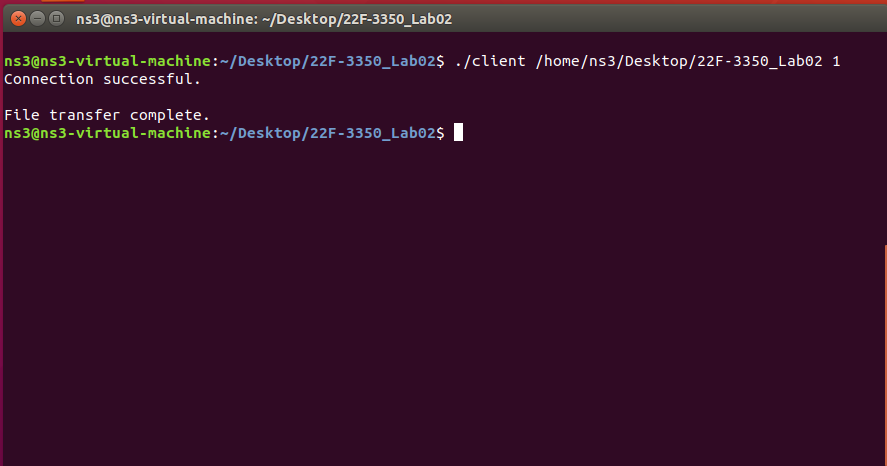
return 0;

}

## Server:



## Client:



## Output (Both):

