Белорусский Национальный Технический Университет

Кафедра «Инновационные процессы»

Лабораторная работа № 5

 «Клиент-серверное сокетное соединение в сети»

|  |
| --- |
| Выполнил: студент группы 42221  Мельник К.А. Чушпан К.  Проверил:  Станкевич С.Н. |

МИНСК 2023

Цель работы:

Изучить механизм сокетного сетевого взаимодействия в Linux.

Задания:

**Задание 1:**

Напишите программы с клиент-серверной архитектурой по следующим требованиям:

1. Обменяться числами. Сервер возводит получаемое от клиента число в квадрат и возвращает ответ клиенту.

Листинг кода:

Client.c:

#include <stdio.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <unistd.h>

#define PORT 12345

void client(int number) {

int client\_socket;

struct sockaddr\_in server\_addr;

// Создание сокета

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

// Настройка адреса сервера

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

server\_addr.sin\_addr.s\_addr = INADDR\_ANY;

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

// Подключение к серверу connect(client\_socket, (struct sockaddr\*)&server\_addr, sizeof(server\_addr));

// Отправка данных серверу

char buffer[1024];

sprintf(buffer, "%d", number);

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

// Получение результата от сервера

recv(client\_socket, buffer, sizeof(buffer), 0);

printf("Результат для числа %d: %s\n", number, buffer);

// Закрытие сокета

close(client\_socket);

}

int main() {

client(160);

return 0;

}

Server.c:

#include <stdio.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <unistd.h>

#define PORT 12345

int main() {

int server\_socket, client\_socket;

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

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

// Создание сокета

server\_socket = socket(AF\_INET, SOCK\_STREAM, 0);

// Настройка адреса сервера

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

server\_addr.sin\_addr.s\_addr = INADDR\_ANY;

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

// Привязка сокета к адресу

bind(server\_socket, (struct sockaddr\*)&server\_addr, sizeof(server\_addr));

// Ожидание подключения клиента

listen(server\_socket, 1);

printf("Сервер запущен. Ожидание подключения клиента...\n");

// Принятие подключения

client\_socket = accept(server\_socket, (struct sockaddr\*)&client\_addr, &client\_addr\_len);

printf("Подключен клиент\n");

while (1) {

char buffer[2048];

int num, result;

// Получение данных от клиента

recv(client\_socket, buffer, sizeof(buffer), 0);

// Преобразование строки в число

num = atoi(buffer);

// Вычисление квадрата числа

result = num\*num;

// Отправка результата клиенту

sprintf(buffer, "%d", result);

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

}

// Закрытие сокетов

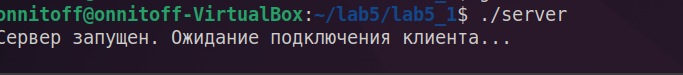
close(client\_socket);

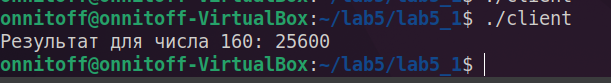
close(server\_socket);

return 0;

}

Работоспособность:





2. Создать два клиента на один и тот же сокет. Используют один и тот же сокет. Первый клиент получает результат счета от 1 до 10, второй – результат счета от 100 до 110 (подумайте, как распознать, какой клиент обратился к серверу).

3. Создать два клиента на один и тот же сервер. Сокеты должны быть разными (именованное соединение в каждом случае должно быть различным). Первый клиент получает результат счета от 1 до 10, второй – от 100 до 110.

**Задание 2:**

Интернет сокеты Напишите программу, которая читает главную страницу из какого-нибудь Web-сервера. Запуск программы должен производиться с консоли.

Лисиинг кода:

#include <stdio.h>

#include <sys/socket.h>

#include <netdb.h>

#include <stdlib.h>

#include <string.h>

#define MAX\_BUFFER\_SIZE 1024

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

if (argc != 2) { // проверяется, что программа была запущена с одним аргументом командной строки

fprintf(stderr, "Использование: %s <URL>\n", argv[0]);

return EXIT\_FAILURE;

}

char url[MAX\_BUFFER\_SIZE]; // url

snprintf(url, MAX\_BUFFER\_SIZE, "%s", argv[1]); // url устанавливается из второгшо аргумента командной строки

// Получаем информацию о хосте url

struct addrinfo hints, \*res;

memset(&hints, 0, sizeof(hints));

hints.ai\_family = AF\_UNSPEC;

hints.ai\_socktype = SOCK\_STREAM;

int status = getaddrinfo(url, "http", &hints, &res);

if (status != 0) {

fprintf(stderr, "Ошибка getaddrinfo: %s\n", gai\_strerror(status));

return EXIT\_FAILURE;

}

// Создаем сокет

int sockfd = socket(res->ai\_family, res->ai\_socktype, res->ai\_protocol);

if (sockfd == -1) {

perror("Ошибка создания сокета");

freeaddrinfo(res);

return EXIT\_FAILURE;

}

// Устанавливаем соединение

if (connect(sockfd, res->ai\_addr, res->ai\_addrlen) == -1) {

perror("Ошибка соединения");

close(sockfd);

freeaddrinfo(res);

return EXIT\_FAILURE;

}

// Отправляем запрос на сервер

char request[MAX\_BUFFER\_SIZE];

snprintf(request, MAX\_BUFFER\_SIZE, "GET / HTTP/1.1\r\nHost: %s\r\n\r\n", url); send(sockfd, request, strlen(request), 0);

// Получаем и выводим ответ от сервера

char buffer[MAX\_BUFFER\_SIZE];

ssize\_t bytes\_received;

while ((bytes\_received = recv(sockfd, buffer, MAX\_BUFFER\_SIZE - 1, 0)) > 0) {

buffer[bytes\_received] = '\0';

printf("%s", buffer);

}

// Освобождаем ресурсы

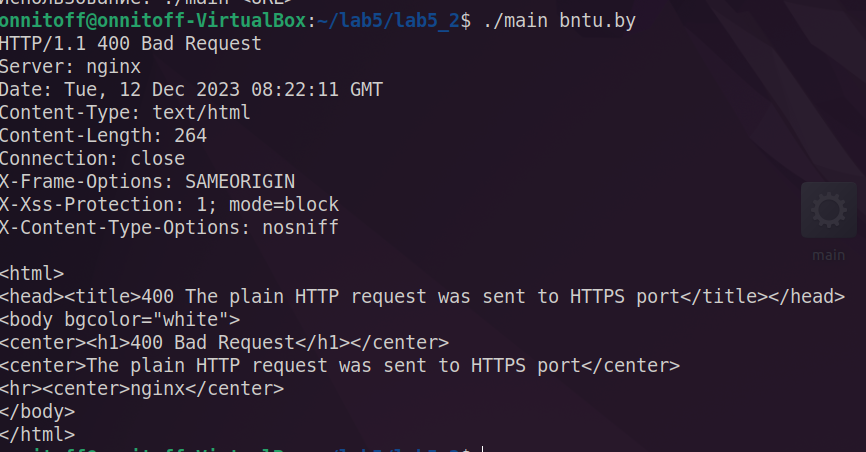
close(sockfd);

freeaddrinfo(res);

return EXIT\_SUCCESS;

}

Работоспособность:



**Задание 3:**

Дейтаграммные сокеты Напишите программу с клиент-серверной архитектурой для передачи данных через дейтаграммные сокеты. Количество клиентов должно соответствовать количеству членов вашей команды, но не менее двух. Семейства протоколов и способы взаимодействия сокетов-клиентов должны отличаться. Запустите программы в разных терминалах и определите какие программыклиенты получили данные. Объясните полученный результат.

Листинг кода:

server\_udp.c:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <arpa/inet.h>

#define PORT\_UDP 12345

#define BUFFER\_SIZE 1024

int main() {

int udp\_socket, length;

char buffer[BUFFER\_SIZE];

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

// Создание UDP сокета

udp\_socket = socket(AF\_INET, SOCK\_DGRAM, 0);

if (udp\_socket == -1) {

perror("UDP socket creation failed");

exit(EXIT\_FAILURE);

}

// Заполнение структуры адреса сервера

memset(&server\_addr, 0, sizeof(server\_addr));

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

server\_addr.sin\_addr.s\_addr = INADDR\_ANY;

server\_addr.sin\_port = htons(PORT\_UDP);

// Привязка сокета к адресу и порту

if (bind(udp\_socket, (const struct sockaddr\*)&server\_addr, sizeof(server\_addr)) == -1) {

perror("UDP socket bind failed");

close(udp\_socket);

exit(EXIT\_FAILURE);

}

printf("UDP Server listening on port %d...\n", PORT\_UDP);

while (1) {

// Ожидание данных от клиента

length = sizeof(client\_addr);

memset(buffer, 0, BUFFER\_SIZE);

recvfrom(udp\_socket, (char\*)buffer, BUFFER\_SIZE, 0, (struct sockaddr\*)&client\_addr, &length);

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

}

close(udp\_socket);

return 0;

}

Client\_udp.c:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <arpa/inet.h>

#define PORT\_UDP 12345

#define BUFFER\_SIZE 1024

int main() {

int udp\_socket;

char buffer[BUFFER\_SIZE];

struct sockaddr\_in server\_addr;

// Создание UDP сокета

udp\_socket = socket(AF\_INET, SOCK\_DGRAM, 0);

if (udp\_socket == -1) {

perror("UDP socket creation failed");

exit(EXIT\_FAILURE);

}

// Заполнение структуры адреса сервера

memset(&server\_addr, 0, sizeof(server\_addr));

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

server\_addr.sin\_port = htons(PORT\_UDP);

server\_addr.sin\_addr.s\_addr = INADDR\_ANY;

// Отправка данных серверу

printf("Enter message for UDP Server: ");

fgets(buffer, BUFFER\_SIZE, stdin);

sendto(udp\_socket, (const char\*)buffer, strlen(buffer), 0, (const struct sockaddr\*)&server\_addr, sizeof(server\_addr));

close(udp\_socket);

return 0;

}

Server\_tcp.c:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <arpa/inet.h>

#define PORT\_TCP 12346

#define BUFFER\_SIZE 1024

int main() {

int tcp\_socket, new\_socket, length;

char buffer[BUFFER\_SIZE];

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

// Создание TCP сокета

tcp\_socket = socket(AF\_INET, SOCK\_STREAM, 0);

if (tcp\_socket == -1) {

perror("TCP socket creation failed");

exit(EXIT\_FAILURE);

}

// Заполнение структуры адреса сервера

memset(&server\_addr, 0, sizeof(server\_addr));

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

server\_addr.sin\_addr.s\_addr = INADDR\_ANY;

server\_addr.sin\_port = htons(PORT\_TCP);

// Привязка сокета к адресу и порту

if (bind(tcp\_socket, (const struct sockaddr\*)&server\_addr, sizeof(server\_addr)) == -1) {

perror("TCP socket bind failed");

close(tcp\_socket);

exit(EXIT\_FAILURE);

}

// Прослушивание сокета

if (listen(tcp\_socket, 5) == -1) {

perror("TCP socket listen failed");

close(tcp\_socket);

exit(EXIT\_FAILURE);

}

printf("TCP Server listening on port %d...\n", PORT\_TCP);

while(1) {

// Ожидание подключения клиента

length = sizeof(client\_addr);

new\_socket = accept(tcp\_socket, (struct sockaddr\*)&client\_addr, (socklen\_t\*)&length);

// Чтение данных от клиента

memset(buffer, 0, BUFFER\_SIZE);

read(new\_socket, buffer, BUFFER\_SIZE);

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

close(new\_socket);

}

close(tcp\_socket);

return 0;

}

Client\_tcp.c:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <arpa/inet.h>

#define PORT\_TCP 12346

#define BUFFER\_SIZE 1024

int main() {

int tcp\_socket;

char buffer[BUFFER\_SIZE];

struct sockaddr\_in server\_addr;

// Создание TCP сокета

tcp\_socket = socket(AF\_INET, SOCK\_STREAM, 0);

if (tcp\_socket == -1) {

perror("TCP socket creation failed");

exit(EXIT\_FAILURE);

}

// Заполнение структуры адреса сервера

memset(&server\_addr, 0, sizeof(server\_addr));

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

server\_addr.sin\_port = htons(PORT\_TCP);

server\_addr.sin\_addr.s\_addr = INADDR\_ANY;

// Подключение к серверу

if (connect(tcp\_socket, (struct sockaddr\*)&server\_addr, sizeof(server\_addr)) == -1) {

perror("TCP socket connect failed");

close(tcp\_socket);

exit(EXIT\_FAILURE);

}

// Отправка данных серверу

printf("Enter message for TCP Server: ");

fgets(buffer, BUFFER\_SIZE, stdin);

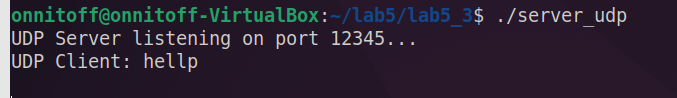
write(tcp\_socket, buffer, strlen(buffer));

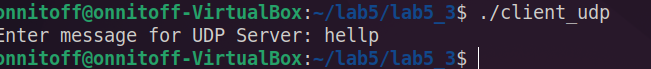
close(tcp\_socket);

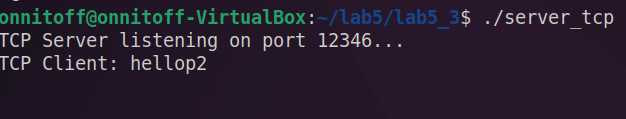
return 0;

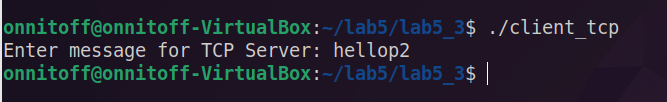
}

Работоспособность:









Вывод:

Различия в протоколах (UDP и TCP) отражаются в их характеристиках передачи данных: UDP - без установления соединения, немаркированные пакеты; TCP - установление соединения, гарантированная доставка, управление потоком и порядком передачи данных.