Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика и системы управления
КАФЕДРА Информационная безопасность (ИУ8)

Безопасность Систем Баз Данных

Отчет
по Лабораторной работе №6
“Работа с ip-socket”

Выполнил:
Евула А. С.,
студент группы ИУ8-63

Проверил:
Зенькович С. А.,
старший преподаватель
кафедры ИУ8

Москва, 2021

ОГЛАВЛЕНИЕ

ЦЕЛЬ РАБОТЫ

Разработать приложения для работы с IP-сокетами.

ОСНОВНАЯ ЧАСТЬ

1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

IP-сокет - механизм обмена данными между процессами, исполняеиыми как на одной машине, так и на разных(передача происходит по сети). Для взаимодействия с помощью стека протоколов TCP/IP используются адреса и порты.
Такая пара определяет сокет. Адрес - 32-битная структура (IPv4) или 128-битная (IPv6). Номер порта - целое число в диапазоне от 0 до 65535 (для TCP).

2. ПРАКТИЧЕСКАЯ ЧАСТЬ

В ходе выполнения лабораторной работы были реализованы сервер и клиент.

1. СЕРВЕР

- создает локальный сокет
- соединяет его с именем сокета
- ожидает подключения клиентов
- устанавливает подключение с ними
- запрашивает имя (идентификация)
- получает и отправляет данные клиенту
- ведет отчет передаваемых данных (логирование в `server.log`)
- при получении сообщения “exit” отключается от клиента

На входной вопрос (сообщение) клиента сервер отвечает случайно фразой по принципу “8-ball”
Для удобного использования были вынесены 2 метода `print_state` (логирует информацию и выводит ее в нужный поток), `send_to_client` (логирует отправляемое сообщение и отправляет его клиенту).

1. СОЗДАНИЕ СОКЕТА И ЕГО НАСТРОЙКА

```
int listenFd = socket(AF_INET, SOCK_STREAM, 0);
if (listenFd < 0)
{
    print_state(log, 1, "ERROR: socket creation failure\n");
    exit(1);
}

int opt = 1;
if (setsockopt(listenFd, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT, &opt,
sizeof(opt)))
{
    print_state(log, 1, "ERROR: socket opt failure\n");
    exit(1);
}
```

2. СВЯЗЫВАНИЕ СОКЕТА С ЛОКАЛЬНЫМ АДРЕСОМ И УСТАНОВКА В РЕЖИМ ПРОСЛУШИВАНИЯ

```
if (bind(listenFd, (const struct sockaddr *)&serverAddr, sizeof(struct
sockaddr_in)) < 0)
{
    print_state(log, 1, "ERROR: bind failure\n");
    exit(1);
}

if (listen(listenFd, 20) < 0)
{
    print_state(log, 1, "ERROR: listening failure\n");
    exit(1);
}
```

3. ФУНКЦИЯ ЗАПИСИ В ПОТОК (С ЗАПИСЬЮ В ЛОГ)

```
template <typename... Args>
void print_state(FILE *log, int code, char *format, Args... args)
{
    fprintf((code == 0) ? stdout : stderr, format, args...);
    fprintf(log, format, args...);
}
```

4. ФУНКЦИЯ ОТПРАВКИ СООБЩЕНИЯ КЛИЕНТУ (С ЗАПИСЬЮ В ЛОГ)

```
void send_to_client(FILE *log, int &client, char *msg)
{
    char buffer[BUF_SIZE] = {};

    print_state(log, 0, "[%s] %s\n", IP, msg);

    sprintf(buffer, "%s\n", msg);
    write(client, buffer, strlen(buffer));
}
```

Исходный код в Приложение А и Приложение В (массив возможных ответов на вопрос клиента).

2. КЛИЕНТ

- создает локальный сокет
- соединяет его с именем сокета
- устанавливает подключение с сервером
- получает и отправляет данные серверу
- при вводе "exit" отключается от сервера

1. СОЗДАНИЕ СОКЕТА

```
int clientFd = socket(AF_INET, SOCK_STREAM, 0);
if (clientFd < 0)
{
    fprintf(stderr, "ERROR: socket creation failure\n");
    exit(1);
}
```

2. НАСТРОЙКА СОКЕТА И ПОДКЛЮЧЕНИЕ К СЕРВЕРУ

```
int opt = 1;
if (setsockopt(clientFd, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT, &opt,
sizeof(opt)))
{
    fprintf(stderr, "ERROR: socket opt failure\n");
    exit(1);
}
struct sockaddr_in serverAddr{};
memset(&serverAddr, 0, sizeof(struct sockaddr_in));
serverAddr.sin_family = AF_INET;
serverAddr.sin_port = htons(PORT);

if (inet_pton(AF_INET, IP, &serverAddr.sin_addr) <= 0)
{
    fprintf(stderr, "ERROR: server not found\n");
    exit(1);
}

if (connect(clientFd, (const struct sockaddr *)&serverAddr, sizeof(struct
sockaddr_in)) < 0)
{
    fprintf(stderr, "ERROR: connection failure\n");
    exit(1);
}
```

Исходный код в Приложение Б.

3. ПРИМЕР ВЗАИМОДЕЙСТВИЯ КЛИЕНТА И СЕРВЕРА

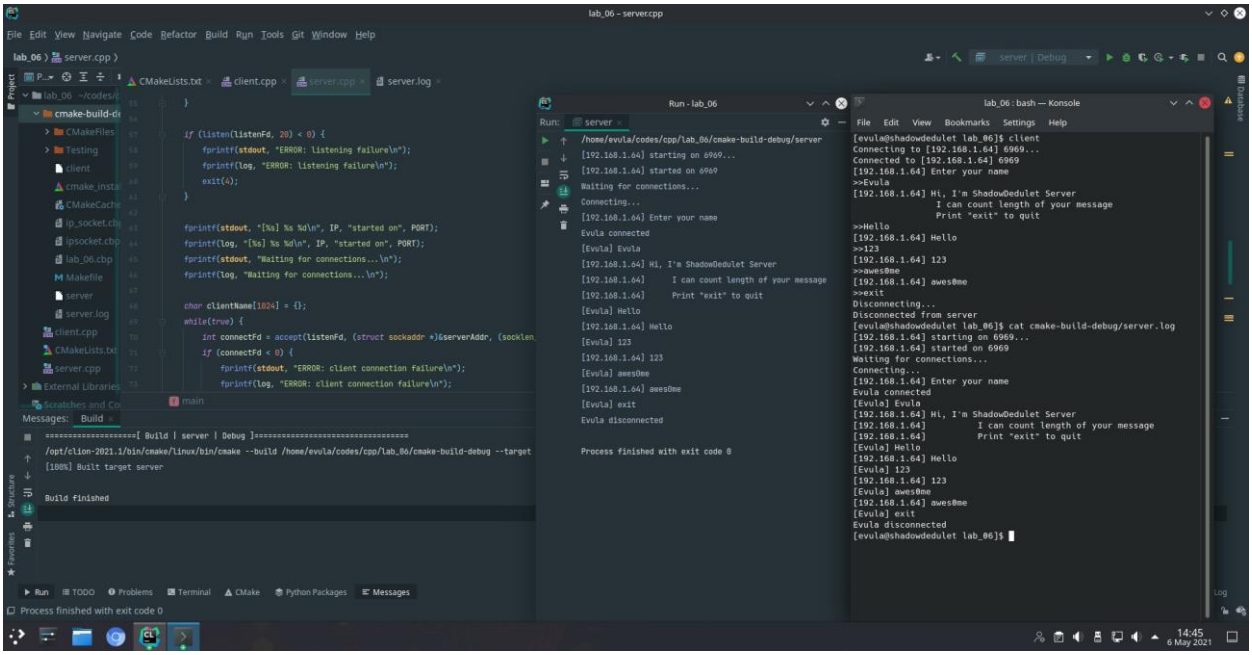

Рисунок 1 – пример использования unix-сокетов

ВЫВОДЫ

Были получены знания о IP-сокетах и принципах взаимодействия сервер-клиент.

ПРИЛОЖЕНИЕ А

```
#include <cstdio>
#include <cstdlib>
#include <sys/socket.h>
#include <netinet/in.h>
#include <unistd.h>
#include <cstring>
#include <string>
#include "Answers.h"

#define IP "192.168.1.67"
#define PORT 6969
#define BUF_SIZE 1024

template <typename... Args>
void print_state(FILE *log, int code, char *format, Args... args)
{
    fprintf((code == 0) ? stdout : stderr, format, args...);
    fprintf(log, format, args...);
}

void send_to_client(FILE *log, int &client, char *msg)
{
    char buffer[BUF_SIZE] = {};

    print_state(log, 0, "[%s] %s\n", IP, msg);

    sprintf(buffer, "%s\n", msg);
    write(client, buffer, strlen(buffer));
}

int main(int argc, char *argv[])
{
    FILE *log;
    log = fopen("./server.log", "w");

    srand(time(nullptr));

    print_state(log, 0, "[%s] starting on %d...\n", IP, PORT);

    int listenFd = socket(AF_INET, SOCK_STREAM, 0);
    if (listenFd < 0)
    {
        print_state(log, 1, "ERROR: socket creation failure\n");
        exit(1);
    }

    int opt = 1;
    if (setsockopt(listenFd, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT, &opt,
sizeof(opt)))
    {
```

```
        print_state(log, 1, "ERROR: socket opt failure\n");
        exit(1);
    }

    struct sockaddr_in serverAddr
    {
    };
    int serverAddrLen = sizeof(serverAddr);

    serverAddr.sin_family = AF_INET;
    serverAddr.sin_addr.s_addr = INADDR_ANY;
    serverAddr.sin_port = htons(PORT);

    if (bind(listenFd, (const struct sockaddr *)&serverAddr, sizeof(struct
sockaddr_in)) < 0)
    {
        print_state(log, 1, "ERROR: bind failure\n");
        exit(1);
    }

    if (listen(listenFd, 20) < 0)
    {
        print_state(log, 1, "ERROR: listening failure\n");
        exit(1);
    }

    print_state(log, 0, "[%s] started on %d\n", IP, PORT);
    print_state(log, 0, "Waiting for connections...\n");

    char clientName[BUF_SIZE] = {};
    while (true)
    {
        int connectFd = accept(listenFd, (struct sockaddr *)&serverAddr,
(socklen_t *)&serverAddrLen);
        if (connectFd < 0)
        {
            print_state(log, 1, "ERROR: client connection failure\n");
        }
        print_state(log, 0, "Connecting...\n");

        send_to_client(log, connectFd, "Enter your name");
        read(connectFd, clientName, BUF_SIZE);
        print_state(log, 0, "%s connected\n", clientName);
        print_state(log, 0, "[%s] %s\n", clientName, clientName);

        send_to_client(log, connectFd, "Hi, I'm ShadowDedulet Server");
        send_to_client(log, connectFd, "\t\tI can count length of your
message");
        send_to_client(log, connectFd, "\t\tPrint \"exit\" to quit");

        while (true)
        {
            char buffer[BUF_SIZE] = {};
            read(connectFd, buffer, BUF_SIZE);
```

```
            print_state(log, 0, "[%s] %s\n", clientName, buffer);

            if (std::string(buffer) == "exit")
            {
                fprintf(log, "%s disconnected\n", clientName);
                close(connectFd);
                break;
            }

            size_t index = rand() % Answers::answers.size();
            char *answer = Answers::answers[index];
            send_to_client(log, connectFd, answer);
        }
        break;
    }
    close(listenFd);
    exit(0);
}
```

ПРИЛОЖЕНИЕ Б

```
#include <cstdio>
#include <cstdlib>
#include <sys/socket.h>
#include <netinet/in.h>
#include <unistd.h>
#include <cstring>
#include <arpa/inet.h>

#define IP "192.168.1.67"
#define PORT 6969
#define BUF_SIZE 1024

int main(int argc, char *argv[])
{
    fprintf(stdout, "Connecting to [%s] %d...\n", IP, PORT);

    int clientFd = socket(AF_INET, SOCK_STREAM, 0);
    if (clientFd < 0)
    {
        fprintf(stderr, "ERROR: socket creation failure\n");
        exit(1);
    }

    int opt = 1;
    if (setsockopt(clientFd, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT, &opt,
sizeof(opt)))
    {
        fprintf(stderr, "ERROR: socket opt failure\n");
        exit(1);
    }
    struct sockaddr_in serverAddr
    {
    };
    memset(&serverAddr, 0, sizeof(struct sockaddr_in));
    serverAddr.sin_family = AF_INET;
    serverAddr.sin_port = htons(PORT);

    if (inet_pton(AF_INET, IP, &serverAddr.sin_addr) <= 0)
    {
        fprintf(stderr, "ERROR: server not found\n");
        exit(1);
    }

    if (connect(clientFd, (const struct sockaddr *)&serverAddr, sizeof(struct
sockaddr_in)) < 0)
    {
        fprintf(stderr, "ERROR: connection failure\n");
        exit(1);
    }
```

```
    fprintf(stdout, "Connected to [%s] %d\n", IP, PORT);

    ssize_t len;
    while (true)
    {
        char read_buf[BUF_SIZE] = {}, write_buf[BUF_SIZE] = {};
        len = read(clientFd, read_buf, BUF_SIZE);
        if (len <= 0)
        {
            fprintf(stdout, "Disconnecting...\n");
            break;
        }

        fprintf(stdout, "[%s] %s>>", IP, read_buf);
        fscanf(stdin, "%[^\n]%*c", write_buf);
        write(clientFd, write_buf, strlen(write_buf));
    }

    fprintf(stdout, "Disconnected from server\n");
    close(clientFd);
    exit(0);
}
```

ПРИЛОЖЕНИЕ В

```
#ifndef LAB_05_ANSWERS_H
#define LAB_05_ANSWERS_H

#include <vector>

struct Answers
{
    static const std::vector<char *> answers;
};
const std::vector<char *> Answers::answers = {
        "It is certain (Бесспорно)",
        "It is decidedly so (Предрешено)",
        "Without a doubt (Никаких сомнений)",
        "You may rely on it (Можешь быть уверен в этом)",
        "As I see it, yes (Мне кажется — «да»)",
        "Most likely (Вероятнее всего)",
        "Outlook good (Хорошие перспективы)",
        "Yes (Да)",
        "Reply hazy, try again (Пока не ясно, попробуй снова)",
        "Ask again later (Спроси позже)",
        "Better not tell you now (Лучше не рассказывать)",
        "Concentrate and ask again (Сконцентрируйся и спроси опять)",
        "Don’t count on it (Даже не думай)",
        "My reply is no (Мой ответ — «нет»)",
        "Outlook not so good (Перспективы не очень хорошие)",
        "Very doubtful (Весьма сомнительно)"};

#endif //LAB_05_ANSWERS_H
```