ЛАБОРАТОРНАЯ РАБОТА №10.

Клиент-серверное сокетное соединение в сети.

ЦЕЛЬ. Изучить механизм сокетного сетевого взаимодействия в LINUX.

ЗАДАНИЕ.

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

server.cpp

```
#include <sys/types.h>
#include <sys/socket.h>
#include <stdio.h>
#include <sys/un.h>
#include <unistd.h>
#include <stdlib.h>
#include <iostream>

using namespace std;

int main()
{
    int server_sockfd, clientsock_fd;

    int server_len, client_len;

    struct sockaddr_un server_address, client_address;
    unlink("server_socket");
    server_sockfd = socket(AF_UNIX, SOCK_STREAM, 0);

    server_address.sun_family = AF_UNIX;
    strcpy(server_address.sun_path, "server_socket");

    server_len = sizeof(server_address);
    bind(server_sockfd, (struct sockaddr*) & server_address, server_len);

    listen(server_sockfd, 5);

    int num;

    client_len = sizeof(client_address);
    clientsock_fd = accept(server_sockfd, (struct sockaddr*)
                                &client_address, (socklen_t*)&client_len);

    while (1)
    {
        cout << "Server is waiting" << endl;
        read(clientsock_fd, &num, sizeof(int));
        cout << "Got from client: " << num << endl;
        num *= num;
        write(clientsock_fd, &num, sizeof(int));
        sleep(1);
    }

    close(clientsock_fd);
}
```

Client.cpp

```cpp
#include <sys/types.h>
#include <sys/socket.h>
#include <stdio.h>
#include <sys/un.h>
#include <stdlib.h>
#include <unistd.h>
#include <iostream>

using namespace std;

int main()
{
    int sockfd;
    int len, result;
    int num = 10;

    struct sockaddr_un address;

    sockfd = socket(AF_UNIX, SOCK_STREAM, 0);

    address.sun_family = AF_UNIX;

    strcpy(address.sun_path, "server_socket");

    len = sizeof(address);

    result = connect(sockfd, (struct sockaddr *) & address, len);

    if(result == 1)
    {
        printf("Error in connection\n");
        exit(-1);
    }

    for(int i = 0; i < 4; i++)
    {
        write(sockfd, &num, sizeof(int));
        sleep(1);

        read(sockfd, &num, sizeof(int));

        cout << "Message from server: " << num << endl;
    }

    cout << "Connection is over: " << num << endl;
    close(sockfd);
    exit(0);

}
```

```
gost@samurai:~/University/3_1/LinuxLabs/lab04/task02$ cd ../task01
gost@samurai:~/University/3_1/LinuxLabs/lab04/task01$ ls
client.cpp  run  server.cpp
gost@samurai:~/University/3_1/LinuxLabs/lab04/task01$ g++ server.cpp
gost@samurai:~/University/3_1/LinuxLabs/lab04/task01$ ./a.out
Server is waiting
Got from client: 10
Server is waiting
Got from client: 100
Server is waiting
Got from client: 10000
Server is waiting
Got from client: 100000000
Server is waiting
Got from client: 1874919424
gost@samurai:~/University/3_1/LinuxLabs/lab04/task01$
```

```
gost@samurai:~/University/3_1/LinuxLabs/lab04/task01$ ls
client.cpp  run  server.cpp
gost@samurai:~/University/3_1/LinuxLabs/lab04/task01$ g++ client.cpp
gost@samurai:~/University/3_1/LinuxLabs/lab04/task01$ ./a.out
Message from server: 100
Message from server: 10000
Message from server: 100000000
Message from server: 1874919424
Connection is over: 1874919424
gost@samurai:~/University/3_1/LinuxLabs/lab04/task01$
```

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).

```
#include <sys/types.h>
#include <sys/socket.h>
#include <stdio.h>
#include <sys/un.h>
#include <unistd.h>
#include <stdlib.h>
#include <iostream>
#include <thread>

using namespace std;


void polling(int id, int server_sockfd, sockaddr_un client_address, int client_len, int clientsock_fd)
{
    cout << id << " started";
    int num;

    client_len = sizeof(client_address);
    clientsock_fd = accept(server_sockfd, (struct sockaddr*)
                                    &client_address, (socklen_t*)&client_len);
    while (1)
    {
        cout << "Server is waiting" << endl;
        read(clientsock_fd, &num, sizeof(int));
        cout << "Got from client "<< id << ": " << num << endl;
        num++;
        write(clientsock_fd, &num, sizeof(int));
        sleep(1);
    }


}
```

```
int main()
{
    int server_sockfd, clientsock_fd;

    int server_len, client_len;

    struct sockaddr_un server_address, client_address;
    unlink("server_socket");
    server_sockfd = socket(AF_UNIX, SOCK_STREAM, 0);

    server_address.sun_family = AF_UNIX;
    strcpy(server_address.sun_path, "server_socket");

    server_len = sizeof(server_address);
    bind(server_sockfd, (struct sockaddr*) & server_address, server_len);

    listen(server_sockfd, 5);

    thread thread_client_1(polling, 1, server_sockfd, client_address, client_len, clientsock_fd);
    thread thread_client_2(polling, 2, server_sockfd, client_address, client_len, clientsock_fd);

    thread_client_1.join();
    thread_client_2.join();
}
```

Два client.cpp разбиты на _1 и _2 разница только в начальном значении. Код схож с первой задачей.

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.

```
Got from client 2: 104
Server is waiting
Got from client 1: 15
Server is waiting
Got from client 2: 105
Server is waiting
Got from client 1: 16
Server is waiting
Got from client 2: 106
Server is waiting
Got from client 1: 17
Server is waiting
Got from client 2: 107
Server is waiting
Got from client 1: 18
Server is waiting
Got from client 2: 108
Server is waiting
Got from client 1: 19
Server is waiting
Got from client 2: 109
Server is waiting
Got from client 1: 110
gost@samurai:~/University/3_1/LinuxLabs/lab04/task03$
```

```
gost@samurai: ~/University/3_1/LinuxLabs/lab04/task03
gost@samurai:~/University/3_1/LinuxLabs/lab04/task03$ g++ client_1.cpp
gost@samurai:~/University/3_1/LinuxLabs/lab04/task03$ ./a.out
Message from server: 11
Message from server: 12
Message from server: 13
Message from server: 14
Message from server: 15
Message from server: 16
Message from server: 17
Message from server: 18
Message from server: 19
Message from server: 20
Connection is over: 20
gost@samurai:~/University/3_1/LinuxLabs/lab04/task03$
```

```
gost@samurai:~/Univers
gost@samurai:~/Univers
Message from server: 1
Message from server: 1
Message from server: 1
Message from server: 1
Message from server: 1
Message from server: 1
Message from server: 1
Message from server: 1
Message from server: 1
Connection is over: 11
gost@samurai:~/Univers
```

```cpp
#include <sys/types.h>
#include <sys/socket.h>
#include <stdio.h>
#include <sys/un.h>
#include <unistd.h>
#include <stdlib.h>
#include <iostream>

using namespace std;

int main()
{
    int server_sockfd, clientsock_fd_1, clientsock_fd_2;

    int server_len, client_len_1, client_len_2;

    struct sockaddr_un server_address, client_address_1, client_address_2;
    unlink("server_socket");
    server_sockfd = socket(AF_UNIX, SOCK_STREAM, 0);

    server_address.sun_family = AF_UNIX;
    strcpy(server_address.sun_path, "server_socket");

    server_len = sizeof(server_address);
    bind(server_sockfd, (struct sockaddr*) & server_address, server_len);

    listen(server_sockfd, 5);

    int num;

    client_len_1 = sizeof(client_address_1);
    clientsock_fd_1 = accept(server_sockfd, (struct sockaddr*)
                                &client_address_1, (socklen_t*)&client_len_1);

    client_len_2 = sizeof(client_address_2);
    clientsock_fd_2 = accept(server_sockfd, (struct sockaddr*)
                                &client_address_2, (socklen_t*)&client_len_2);
```

```cpp
    while (1)
    {
        cout << "Server is waiting" << endl;
        read(clientsock_fd_1, &num, sizeof(int));
        cout << "Got from client 1: " << num << endl;
        num++;
        write(clientsock_fd_1, &num, sizeof(int));
        sleep(1);

        cout << "Server is waiting" << endl;
        read(clientsock_fd_2, &num, sizeof(int));
        cout << "Got from client 2: " << num << endl;
        num++;
        write(clientsock_fd_2, &num, sizeof(int));
        sleep(1);
    }

    close(clientsock_fd_1);
    close(clientsock_fd_2);


}
```

```
Got from client 2: 104
Server is waiting
Got from client 1: 15
Server is waiting
Got from client 2: 105
Server is waiting
Got from client 1: 16
Server is waiting
Got from client 2: 106
Server is waiting
Got from client 1: 17
Server is waiting
Got from client 2: 107
Server is waiting
Got from client 1: 18
Server is waiting
Got from client 2: 108
Server is waiting
Got from client 1: 19
Server is waiting
Got from client 2: 109
Server is waiting
Got from client 1: 110
gost@samurai:~/University/3_1/LinuxLabs/lab04/task03$
```

```
gost@samurai: ~/University/3_1/LinuxLabs/lab04/task03
gost@samurai:~/University/3_1/LinuxLabs/lab04/task03$ g++ client_1.cpp
gost@samurai:~/University/3_1/LinuxLabs/lab04/task03$ ./a.out
Message from server: 11
Message from server: 12
Message from server: 13
Message from server: 14
Message from server: 15
Message from server: 16
Message from server: 17
Message from server: 18
Message from server: 19
Message from server: 20
Connection is over: 20
gost@samurai:~/University/3_1/LinuxLabs/lab04/task03$
```

```
server.cpp .../task01   client_2.cpp   client_1.cpp
gost@samurai: ~/University/3_1/LinuxLabs/lab04/task03
gost@samurai:~/University/3_1/LinuxLabs/lab04/task03$ g++ client_2.cpp
gost@samurai:~/University/3_1/LinuxLabs/lab04/task03$ ./a.out
Message from server: 101
Message from server: 102
Message from server: 103
Message from server: 104
Message from server: 105
Message from server: 106
Message from server: 107
Message from server: 108
Message from server: 109
Message from server: 110
Connection is over: 110
gost@samurai:~/University/3_1/LinuxLabs/lab04/task03$
```