**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра вычислительной техники**

отчет

**по лабораторной работе №10**

**по дисциплине «Организация процессов и программирование в среде Linux»**

Темы: **ВЗАИМОДЕЙСТВИЕ ПРОЦЕССОВ ЧЕРЕЗ СОКЕТЫ**

|  |  |  |
| --- | --- | --- |
| Студент гр. 8306 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Пеунов В.В. |
| Преподаватель | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Разумовский Г.В. |

Санкт-Петербург,

2021

**Цель работы**

Знакомство с организацией взаимодействия процессов через сокеты

**Задание на лабораторную работу**

Написать две программы (сервер и клиент) , которые обмениваются сообщениями через потоковые сокеты. Клиенты проверяют возможность соединения с сервером и в случае отсутствия соединения или истечения времени ожидания отправки сообщения завершают работу. После соединения с сервером они генерируют случайную последовательность чисел и выводят ее на экран, а затем отсылают серверу. Сервер в течение определенного времени ждет запросы от клиентов и в случае их отсутствия завершает работу. При поступлении запроса от клиента сервер порождает обслуживающий процесс, который принимает последовательность чисел, упорядочивает ее и выводит на экран, а затем отсылает обратно клиенту и завершают работу. Клиент полученную последовательность выводит на экран и заканчивает свою работу

**Описание работы**

Скриншот работы программы представлен на рисунке 1.

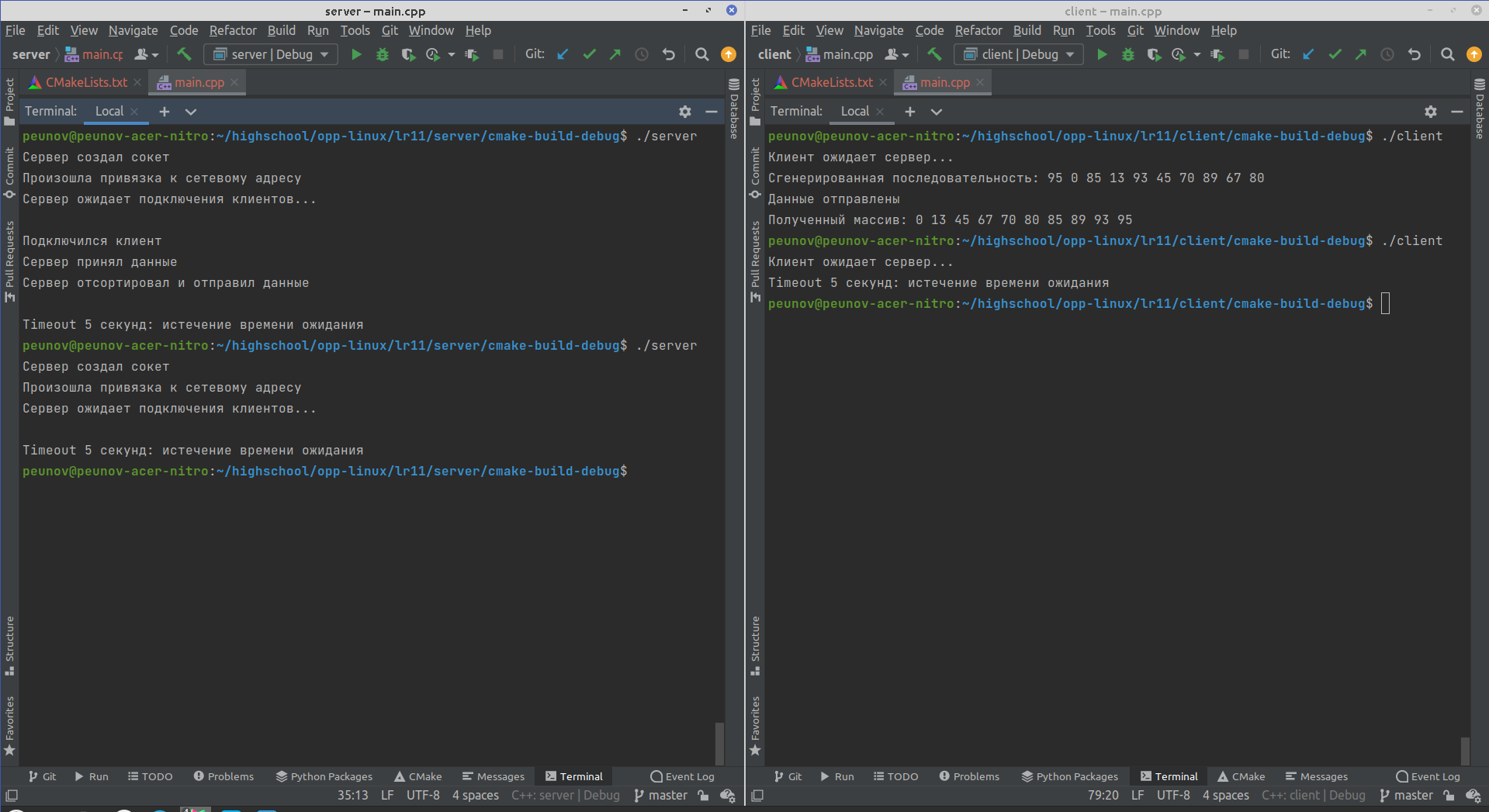


Рисунок 1.

**Вывод:** Знакомство с организацией взаимодействия процессов через сокеты

**Приложение A. Клиент**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<sys/socket.h>

#include<arpa/inet.h>

#include <fcntl.h>

#include <unistd.h>

#include <time.h>

#include <iostream>

#define MSG\_LEN 10

using namespace std;

int \*arr;

void generate\_nums(){

arr = (int\*)malloc(MSG\_LEN);

printf("Сгенерированная последовательность: ");

for(int i = 0; i < MSG\_LEN; i++){

arr[i] = rand() % 100;

printf("%d ", arr[i]);

}

printf("\n");

}

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

srand(time(nullptr));

struct timeval tv;

int server\_reply[1024];

int wv, delta, connection;

fd\_set fds;

struct sockaddr\_in server;

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

server.sin\_family = AF\_INET;

server.sin\_port = htons( 8889);

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

if (socket\_descriptor == -1){

printf("Ошибка при создании сокета\n");

}

printf("Клиент ожидает сервер...\n");

time\_t start = time(NULL);

do {

delta = time(NULL) - start;

connection = connect(socket\_descriptor, (struct sockaddr \*) &server, sizeof(server));

}

while (delta < 5 && connection < 0);

if(connection < 0){

cout << "Timeout 5 секунд: истечение времени ожидания" << endl;

exit(2);

}

generate\_nums();

if (send(socket\_descriptor, arr, MSG\_LEN \* 4, 0) < 0) {

cout << "Отправка не удалась" << endl;

exit(EXIT\_FAILURE);

}

cout << "Данные отправлены" << endl;

FD\_ZERO(&fds);

FD\_SET(socket\_descriptor, &fds);

tv.tv\_sec=5;

tv.tv\_usec=0;

wv = select(FD\_SETSIZE, &fds, nullptr, nullptr, &tv);

if(wv == 0){

cout << "Истечение времени ожидания!" << endl;

exit(EXIT\_FAILURE);

}

recv(socket\_descriptor, server\_reply, 1024, 0);

cout << "Полученный массив: ";

for (int i = 0; i < MSG\_LEN; i++) {

cout << server\_reply[i] << " ";

}

cout << endl;

return 0;

}

**Приложение Б. Сервер**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<sys/socket.h>

#include<arpa/inet.h>

#include <fcntl.h>

#include <unistd.h>

#include <iostream>

using namespace std;

int socket\_descriptor, attached\_socket;

//char \*message;

int reply[1024];

#define MSG\_LEN 10

int cmp(const void \*a, const void \*b) {

return \*(int\*)a - \*(int\*)b;

}

void service\_process(){

//pid\_t pid;

switch(fork()) {

case 0:

close(socket\_descriptor);

recv(attached\_socket, reply , 1024 , 0);

cout << "Сервер принял данные" << endl;

qsort(reply, MSG\_LEN, sizeof(int), cmp);

cout << "Сервер отсортировал и отправил данные" << endl << endl;

send(attached\_socket, reply, MSG\_LEN\*4, 0);

close(attached\_socket);

exit(EXIT\_SUCCESS);

default:

break;

}

}

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

{

struct sockaddr\_in server , client;

int rv, cv;

fd\_set fds;

struct timeval tv;

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

if (socket\_descriptor == -1){

cout << "Ошибка при создании сокета" << endl;

return 0;

}

cout << "Сервер создал сокет" << endl;

server.sin\_family = AF\_INET;

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

server.sin\_port = htons( 8889 );

if(bind(socket\_descriptor, (struct sockaddr \*)&server, sizeof(server)) < 0){

cout << "Ошибка при bind" << endl;

exit(EXIT\_FAILURE);

}

cout << "Произошла привязка к сетевому адресу" << endl;

listen(socket\_descriptor, 5);

cout << "Сервер ожидает подключения клиентов..." << endl << endl;

while(true){

FD\_ZERO(&fds);

FD\_SET(socket\_descriptor, &fds);

tv.tv\_sec = 5;

tv.tv\_usec = 0;

cv = select(FD\_SETSIZE, &fds, nullptr, nullptr, &tv);

if(cv == 0){

cout << "Timeout 5 секунд: истечение времени ожидания" << endl;

break;

} else {

attached\_socket = accept(socket\_descriptor, nullptr, nullptr);

if(attached\_socket < 0){

cout << "Ошибка в функции accept" << endl;

exit(3);

}

cout << "Подключился клиент" << endl;

service\_process();

}

}

close(socket\_descriptor);

return 0;

}