**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра Вычислительной техники**

**ОТЧЕТ**

**по лабораторной работе №11**

**по дисциплине «Организация процессов и программирование**

**В среде Linux»**

**Тема: Взаимодействие процессов через сокеты**

|  |  |  |
| --- | --- | --- |
| Студент гр. 9308 |  | Яловега Н.В. |
| Преподаватель |  | Разумовский Г.В. |

Санкт-Петербург

2022

**Цель работы**

Знакомство с организацией сокеты, системными функциями, обеспечивающими управление сокетами, и их использованием для решения задач межпроцессного взаимодействия.

**Задание**

1. 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.

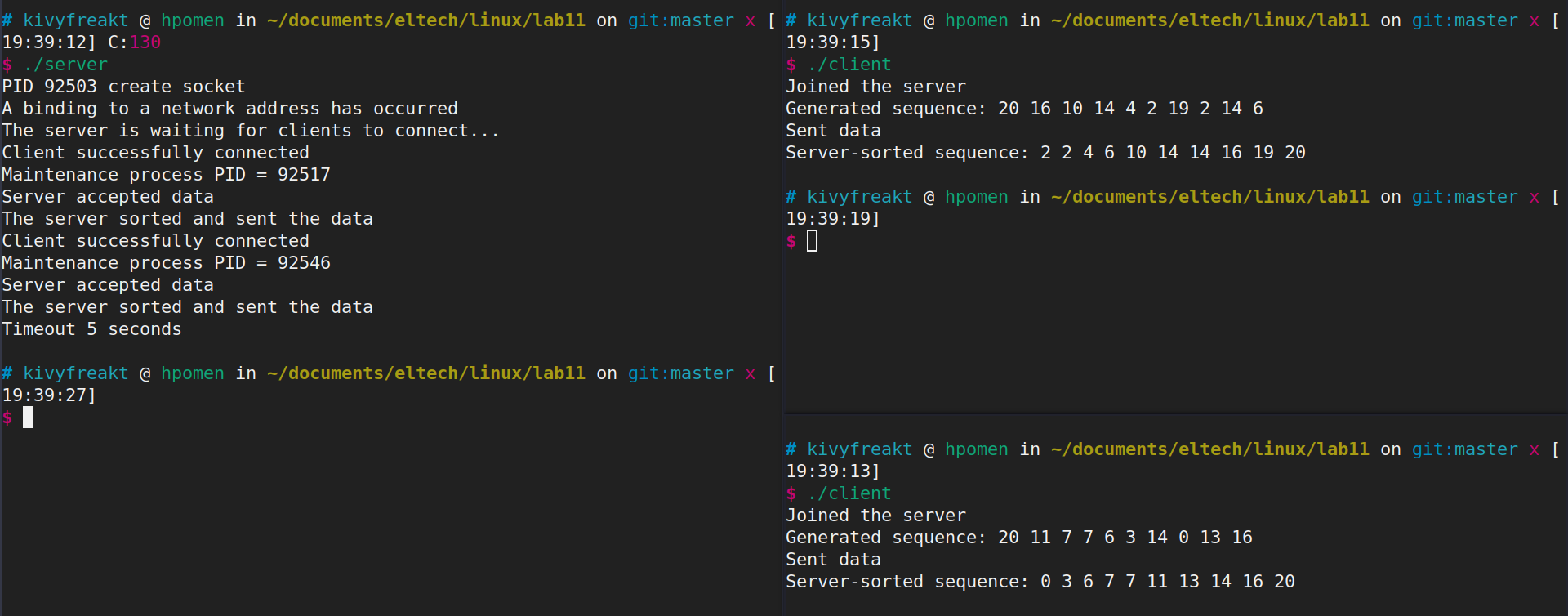
2. Откомпилировать программы и запустить их.

**Порядок выполнения работы**

Разработано 2 программы: сервер и клиент. Сервер открывает сокет, привязывает его к своему сетевому адресу и ждет запросы от клиентов в течение 5 секунд (с помощью функции select). Если запрос клиента поступил, сервер порождает обслуживающий процесс, который принимает от клиента последовательность чисел, сортирует ее и возвращает клиенту. Если после обслуживания последнего клиента, сервер не получает запрос в течение 5 секунд, он завершает свою работу.

При запуске клиент пытается присоединиться к серверу в течение 5 секунд. Если соединение установлено, клиент генерирует случайную последовательность чисел и отправляет ее серверу. Сервер присылает в ответ отсортированный массив чисел, клиент выводит его на экран.

Скриншот выполнения программ представлены на рис 1.



**Исходный код**

**server.cpp**

#include <iostream>

#include <stdlib.h>

#include <string.h>

#include <sys/socket.h>

#include <arpa/inet.h>

#include <fcntl.h>

#include <unistd.h>

#define SEQUENCE\_LEN 10

#define MSG\_LEN 1024

int sock, attached\_socket;

int reply[MSG\_LEN];

int timeout;

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

void service\_process();

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

{

struct sockaddr\_in server, client;

int rv;

struct timeval tv;

if(argc < 2)

{

std::cout << "Incorrect number of arguments!\n" << std::endl;

std::cout << "Example: " << argv[0] << " delay \n";

return 1;

}

timeout = atoi(argv[1]);

tv.tv\_sec = timeout; // установка timeout

tv.tv\_usec = 0;

// создание сокета

// AF\_INET - домен для взаимодействия через сеть по протоколу TCP/IP

// SOCK\_STREAM - тип сокета потоковый

// 0 - автоматический выбор протокола (для потокового IPPROTO\_TCP)

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

if (sock == -1)

{

std::cout << "Error socket creation\n";

}

std::cout << "PID " << getpid() << " create socket\n";

server.sin\_family = AF\_INET; // тип домена

server.sin\_addr.s\_addr = INADDR\_ANY; // адрес (0.0.0.0)

server.sin\_port = htons(8888); // порт

// привязка сокета к адресу и номеру порта

// sock - дескриптор сокета

// server - cтруктура

if(bind(sock,(struct sockaddr \*) &server , sizeof(server)) < 0)

{

std::cout << "can't bind\n";

return -1;

}

std::cout << "A binding to a network address has occurred\n";

// перевод сокета в пассивное ожидание (создаем очередь)

// sock - дескриптор сокета сервера

// 5 - длина очереди

listen(sock, 5);

std::cout << "The server is waiting for clients to connect...\n";

fd\_set readfds; // набор дескрипторов на чтение

FD\_ZERO(&readfds); // очистка набора на чтение

FD\_SET(sock, &readfds); // добавление дескриптора в набор

while(1)

{

// определение состояния сокета

// sock+1 - кол-во опрашиваемых дескрипторов сокетов

// readfds набор дескрипторов, которые следует проверять на готовность к чтению

// tv - timeout

rv = select(sock+1, &readfds, NULL, NULL, &tv);

if(rv > 0)

{

// создание нового присоединенного сокета

// sock - дескриптор сокета сервера

// client - содержит указатель на структуру с адресом сервера

attached\_socket = accept(sock, (struct sockaddr \*)&client, (socklen\_t\*)&client);

if(attached\_socket > 0){

std::cout << "Client successfully connected\n";

service\_process();

}

}

else

{

std::cout << "Timeout " << timeout << " seconds\n";

break;

}

tv.tv\_sec = timeout;

}

// закрытие принимающего сокета

close(sock);

return 0;

}

void service\_process()

{

// порождение обслуживающего потока

if (fork() == 0)

{

printf("Maintenance process PID = %d\n", getpid());

// закрытие принимающего сокета

close(sock);

// ожидание получения информации. чтение информации

if(recv(attached\_socket, reply, MSG\_LEN, 0) > 0)

{

// обработка информации

printf("Server accepted data\n");

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

printf("The server sorted and sent the data\n");

}

// отправка ответа

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

// завершение соединения

close(attached\_socket);

// окончание работы

exit(EXIT\_SUCCESS);

}

}

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

{

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

}

**client.cpp**

#include <iostream>

#include <stdlib.h>

#include <string.h>

#include <sys/socket.h>

#include <arpa/inet.h>

#include <fcntl.h>

#include <unistd.h>

#include <sys/time.h>

#define SEQUENCE\_LEN 10

#define MSG\_LEN 1024

int \*arr;

int timeout;

void generate\_sequence();

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

{

int sock;

struct sockaddr\_in server;

int server\_reply[MSG\_LEN];

int rv;

struct timeval tv;

if(argc < 2)

{

std::cout << "Incorrect number of arguments!\n" << std::endl;

std::cout << "Example: " << argv[0] << " delay \n";

return 1;

}

timeout = atoi(argv[1]);

tv.tv\_sec = timeout;

tv.tv\_usec = 0;

srand(time(NULL));

// создание сокета

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

if (sock == -1)

{

std::cout << "Error creating socket\n";

return -1;

}

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

server.sin\_family = AF\_INET;

server.sin\_port = htons(8888);

fd\_set readfds;

FD\_ZERO(&readfds);

FD\_SET(sock, &readfds);

rv = select(sock+1, &readfds, NULL, NULL, &tv);

if(rv > 0)

{

int start\_time = time(NULL);

int duration = 0;

do

{

if(connect(sock, (struct sockaddr \*)&server, sizeof(server)) >= 0)

{

std::cout << "Joined the server\n";

generate\_sequence();

// посылка данных

if(send(sock, arr, SEQUENCE\_LEN\*4 , 0) < 0)

{

std::cout << "Failed\n";

exit(EXIT\_FAILURE);

}

std::cout << "Sent data\n";

// получение ответа от сервера

if(recv(sock, server\_reply , MSG\_LEN , 0) < 0)

{

std::cout << "Failed\n";

exit(EXIT\_FAILURE);

}

std::cout << "Server-sorted sequence: ";

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

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

std::cout << std::endl;

break;

}

else

duration = time(NULL) - start\_time;

}

while(duration < timeout);

if(duration >= timeout)

std::cout << "Timeout " << timeout << " seconds" << std::endl;

}

return 0;

}

void generate\_sequence()

{

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

std::cout << "Generated sequence: ";

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

{

arr[i] = rand() % 21;

std::cout << arr[i] << " ";

}

std::cout << std::endl;

}**Вывод**

В ходе работы были изучены механизмы организации сокетов, системными функциями, обеспечивающими управление сокетами, и их использованием для решения задач межпроцессного взаимодействия в операционной системе linux.