Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина «Операционные среды и системное программирование»

|  |  |
| --- | --- |
|  |  |

**ОТЧЕТ**

к лабораторной работе № 6

на тему «Элементы сетевого программирования»

Выполнил             А. Д. Филипеня

Проверил                          Н. Ю. Гриценко

Минск 2024

**СОДЕРЖАНИЕ**

[1 Цель работы 3](#_Toc157722973)

[2 Краткие теоретические сведения 4](#_Toc157722974)

[3 Полученные результаты 5](#_Toc157722975)

[Выводы 6](#_Toc157722976)

[Список использованных источников 7](#_Toc157722977)

[Приложение А (обязательное) Листинг исходного кода 8](#_Toc157722978)

# **1 ЦЕЛЬ РАБОТЫ**

Целью выполнения лабораторной работы является практическое освоение основ построения и функционирования сетей, стеков протоколов, программных интерфейсов на примере написания многопользовательского чата с использованием транспортного протокола TCP.

# **2 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

Сеть – типичная сложная система (состоит из подсистем, которые, в свою очередь, также являются сложными системами).

Протокол – набор правил и процедур взаимодействия между одноименными уровнями различных систем, обеспечивают корректную связь участников взаимодействия в сети.

Сокеты предоставляют весьма мощный и гибкий механизм межпроцессного взаимодействия. Они могут использоваться для организации взаимодействия программ на одном компьютере, по локальной сети или через Интернет, что позволяет создавать распределённые приложения различной сложности. Кроме того, с их помощью можно организовать взаимодействие с программами, работающими под управлением других операционных систем.[1]

Сокет – это конечная точка сетевых коммуникаций. Он является чем-то вроде портала, через который можно отправлять байты во внешний мир. Приложение просто пишет данные в сокет, а их дальнейшая буферизация, отправка и транспортировка осуществляется используемым стеком протоколов и сетевой аппаратурой. Чтение данных из сокета происходит аналогичным образом.[2]

С каждым сокетом связываются домен, тип и протокол. Эти атрибуты задаются при создании сокета и остаются неизменными на протяжении всего времени его существования.

Домен определяет пространство адресов, в котором располагается сокет, и множество протоколов, которые используются для передачи данных. Чаще других используются домены Unix и Internet, задаваемые константами AF\_UNIX и AF\_INET соответственно (префикс AF означает «семейство адресов»). При задании AF\_UNIX для передачи данных используется файловая система ввода/вывода Unix. В этом случае сокеты используются для межпроцессного взаимодействия на одном компьютере и не годятся для работы по сети. Константа AF\_INET соответствует Internet-домену. Сокеты, размещённые в этом домене, могут использоваться для работы в любой IP-сети. [3]

Для выполнения поставленного задания были использованы определенные инструменты для работы с сокетами, такие как функция socket для создания сокета, функция bind для связывания сокета с адресом, функция listen для установки соединения, функция accept для обслуживания очередного запроса.

Таким образом, можно понять, что для реализации поставленного задания достаточно изучения основ сетевого программирования.

# **3 ПОЛУЧЕННЫЕ РЕЗУЛЬТАТЫ**

В ходе выполнения поставленного задания был разработан упрощенный чат для нескольких пользователей. Был использован транспортный протокол TCP. Была выбрана централизованная архитектура, т.е. выделенный процесс-сервер и процессы-клиенты.

На сервере создается сокет для приема соединений или отдельных сообщений, также осуществляется передача сообщений нескольким клиентам, кроме отправившего сообщение. Со стороны клиента реализовано обнаружение сервера и соединение с ним, ввод пользовательских сообщений и передача их серверу, а также отображение сообщений других пользователей. Результат работы программы представлен на рисунке 3.1.

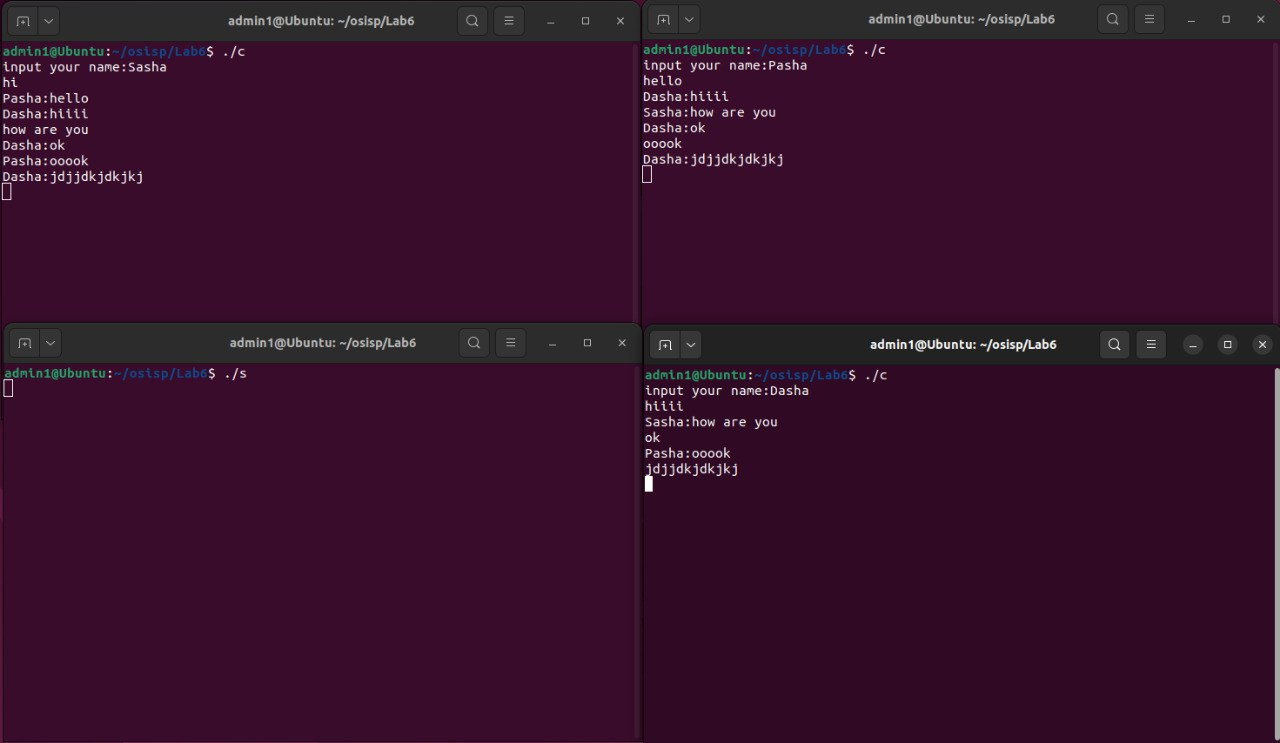


Рисунок 3.1 – Результат работы программы

Таким образом, в ходе лабораторной работы был реализован упрощенный чат для нескольких пользователей с использованием транспортного протокола TCP и централизованной архитектуры.

# **ВЫВОДЫ**

Многопользовательский чат, разработанный в ходе выполнения работы, реализует принципы работы с сокетами. Были использованы транспортный протокол TCP, централизованная архитектура. На сервере создается сокет для приема соединений или отдельных сообщений, осуществляется передача сообщений нескольким клиентам, кроме отправителя сообщения. Со стороны клиента реализовано обнаружение сервера и соединение с ним, ввод пользовательских сообщений и передача их серверу, а также отображение сообщений других пользователей.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

[1] Тейнсли, Д. Linux и другие UNIX-подобные операционные системы: программирование. / Д. Тейнсли. – СПб.: БВХ-Петербург, 2006. – 1056 с.

[2] Щупаков, Ю. Руководство по командам и shell-программированию, операционная система Linux. / Ю. Щупаков. – СПб: Питер, 2008. – 592 с.

[3] UnixTutorials [Электронный ресурс]. – Режим доступа: https://prog /unix-tutorials/. – Дата доступа: 06.02.2024.

## **ПРИЛОЖЕНИЕ А**

## **(обязательное)**

## **Листинг исходного кода**

Листинг 1 – Программный код server.c

#include <sys/types.h>

#include <sys/socket.h>

#include <sys/wait.h>

#include <stdio.h>

#include <stdlib.h>

#include <errno.h>

#include <string.h>

#include <sys/un.h>

#include <sys/time.h>

#include <sys/ioctl.h>

#include <unistd.h>

#include <netinet/in.h>

#include <pthread.h>

#define COUNT 5

int socket\_fd[COUNT];

void pthread\_function(int client\_fd){

char message[1500];

char buf[1024];

int i,recvbytes;

char name[20];

recvbytes = recv(client\_fd, name, 20, 0);

name[recvbytes]=':';

name[recvbytes+1]='\0';

while(1){

if((recvbytes = recv(client\_fd, buf, 1024, 0))==-1){

perror("recv error");

exit(1);

}

if(recvbytes==0){

printf("%sbye!\n",name);

break;

}

buf[recvbytes]='\0';

for(i = 0; i < COUNT; ++i){

if(socket\_fd[i]==-1 || socket\_fd[i] == client\_fd) {

continue;

}

else {

message[0]='\0';

strcat(message,name);

strcat(message,buf);

if(send(socket\_fd[i], message, strlen(message), 0) == -1){

perror("send error");

exit(1);

}

}

}

}

close(client\_fd);

for(i=0;i<COUNT;i++){

if(socket\_fd[i]==client\_fd){

socket\_fd[i]=-1;

}

}

pthread\_exit(NULL);

}

int main(){

int i;

for(i=0;i<COUNT;i++){

socket\_fd[i]=-1;

}

pthread\_t id;

int sockfd,client\_fd;

socklen\_t sin\_size;

struct sockaddr\_in my\_addr;

struct sockaddr\_in remote\_addr;

if((sockfd = socket(AF\_INET,SOCK\_STREAM,0))==-1){

perror("socket");

exit(1);

}

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

my\_addr.sin\_port=htons(12345);

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

bzero(&(my\_addr.sin\_zero),8);

int yes = 1;

if (setsockopt(sockfd, SOL\_SOCKET, SO\_REUSEADDR, &yes, sizeof(int)) == -1) {

perror("setsockopt");

exit(1);

}

if(bind(sockfd,(struct sockaddr \*)&my\_addr,sizeof(struct sockaddr))==-1){

perror("bind");

exit(1);

}

if(listen(sockfd,10)==-1){

perror("listen");

exit(1);

}

i=0;

while(1){

sin\_size=sizeof(struct sockaddr\_in);

if((client\_fd=accept(sockfd,(struct sockaddr \*)&remote\_addr,&sin\_size))==-1){

perror("accept");

exit(1);

}

while(socket\_fd[i]!=-1)

i=(i+1)%COUNT;

socket\_fd[i]=client\_fd;

pthread\_create(&id,NULL,(void \*)pthread\_function,(int \*)client\_fd);

}

}

Листинг 2 – Программный код client.c

#include <sys/types.h>

#include <sys/socket.h>

#include <sys/wait.h>

#include <stdio.h>

#include <stdlib.h>

#include <errno.h>

#include <string.h>

#include <sys/un.h>

#include <sys/time.h>

#include <sys/ioctl.h>

#include <unistd.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <pthread.h>

char recv\_buf[1500],send\_buf[1024];

// Функция ввода потока, отвечающая за отображение полученной информации

void pthread\_function(int sockfd){

int recvbytes;

while(1){

if((recvbytes = recv(sockfd, recv\_buf, 1500, 0))==-1){

perror("recv error");

exit(1);

}else{

recv\_buf[recvbytes]='\0';

printf("%s\n", recv\_buf);

}

}

}

int main(void){

pthread\_t id;

int sockfd;

struct sockaddr\_in server\_addr;

// настройки параметров

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

server\_addr.sin\_port = htons(12345);

server\_addr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

if((sockfd = socket(AF\_INET, SOCK\_STREAM, 0))==-1){

perror("socket error");

exit(1);

}

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

perror("connect error");

exit(1);

}

char name[20];

printf("input your name:");

scanf("%s",name);

send(sockfd,name,strlen(name),0);

pthread\_create(&id,NULL,(void \*)pthread\_function,(int \*)sockfd);

while(1){

gets(send\_buf);

if(send(sockfd,send\_buf,strlen(send\_buf),0)==-1){

perror("send error");

exit(1);

}

sleep(1);

}

close(sockfd);

pthread\_cancel(id);

return 0;

}