**Министерство науки и высшего образования Российской Федерации**

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

**«Томский государственный университет систем управления и радиоэлектроники»**

Кафедра компьютерных систем в управлении и проектировании (КСУП)

Отчет по лабораторной работе № 12 по курсу

«Операционные системы»

**Сетевые виртуальные соединения**

**Выполнил:**

студент гр. 589-3

\_\_\_\_\_\_\_\_\_\_\_\_\_Потлог А.M.

«\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2020г.

**Принял:**

к.т.н., доцент

\_\_\_\_\_\_\_\_Коцубинский В.П.

«\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2020г.

Томск 2020

**Цель работы:**

Целью выполнения настоящей лабораторной работы является получение навыков создания на основе сокетов виртуальных TCP соединений, используемых для связи между удаленными процессами.

**Ход выполнения работы:**

1. Пишем код server.c и client.c и компилируем.

**<server.c>**

#include <sys/types.h>

#include <string.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <stdlib.h>

#include <stdio.h>

#include <ctype.h>

#include <time.h>

#include <fcntl.h>

#include <unistd.h>

#include <signal.h>

#include <arpa/inet.h>

#define LBUF 40

#define LBUF2 20

#define LEN sizeof(struct sockaddr\_in)

#define LOGFILE "server.log"

#define SENDFILE "server.send"

char buffer[LBUF];

char buffer2[LBUF2];

char timemsg[10];

char filebuffer[100];

int nsock;

void s\_action(int sig)

{

close(nsock);

printf("SERVER: got SIGPIPE.\n");

}

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

{

struct sockaddr\_in client, server;

int n, sock, s\_len, fil, fil2;

int c\_len = LEN;

time\_t t;

struct tm \*timeinfo;

static struct sigaction sact;

sact.sa\_handler = s\_action;

sigfillset(&(sact.sa\_mask));

sigaction(SIGPIPE, &sact, NULL);

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

{

printf("SERVER: unable to create socket.\n");

exit(1);

}

bzero(&server, LEN);

server.sin\_family = AF\_INET;

server.sin\_port = 1305;

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

int optval = 1;

setsockopt(sock, SOL\_SOCKET, SO\_REUSEADDR, &optval,

sizeof(optval));

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

{

printf("SERVER: unable to bind socket.\n");

exit(1);

}

if ((fil = open(LOGFILE, O\_RDWR | O\_CREAT | O\_TRUNC, 0660)) <

0)

{

printf("SERVER: unable to init file.\n");

exit(1);

}

close(fil);

if ((listen(sock, 5)) < 0)

{

printf("SERVER: unable to create queue.\n");

exit(1);

}

for (;;)

{

if ((nsock = accept(sock, (struct sockaddr \*) &client,

&c\_len)) < 0)

{

printf("SERVER: unable to pull from

queue.\n");

continue;

}

if (fork() == 0)

{

memset(buffer, 0, LBUF);

n = read(nsock, buffer, LBUF);

if (n < 0)

{

printf("SERVER: recieving error.\n");

exit(1);

}

time(&t);

timeinfo = localtime(&t);

strftime(timemsg, 10, "%T", timeinfo);

sprintf(filebuffer, "From %s:%d at %s: %s\n",

inet\_ntoa(client.sin\_addr), client.sin\_port, timemsg, buffer);

if ((fil = open(LOGFILE, O\_RDWR | O\_APPEND,

0660)) < 0)

{

printf("SERVER: unable to open file

again.\n");

exit(1);

}

write(fil, filebuffer, strlen(filebuffer));

close(fil);

if ((fil2 = open(SENDFILE, O\_RDONLY, 0660)) <

0)

{

printf("SERVER: unable to open .send

file");

exit(1);

}

while((n = read(fil2, buffer2, LBUF2)) > 0)

{

if ((write(nsock, buffer2, n)) != n)

{

printf("SERVER: unable to

send.\n");

exit(1);

}

}

close(fil2);

close(nsock);

exit(0);

}

close(nsock);

}

}

**<client.c>**

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <stdlib.h>

#include <stdio.h>

#include <ctype.h>

#include <string.h>

#define LEN sizeof(struct sockaddr\_in)

#define LBUF 70

char buffer[LBUF];

char \*msg = "message";

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

{

struct sockaddr\_in client, server;

int n, sock, s\_len, c\_len;

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

{

printf("CLIENT: unable to create socket.\n");

exit(1);

}

server.sin\_family = AF\_INET;

server.sin\_addr.s\_addr = inet\_addr(argv[1]);

server.sin\_port = atoi(argv[2]);

if ((connect(sock, (struct sockaddr \*) &server, LEN)) <

0)

{

printf("CLIENT: unable to connect.\n");

exit(1);

}

if ((write(sock, msg, strlen(msg))) != strlen(msg))

{

printf("CLIENT: unable to send.\n");

exit(1);

}

while((n = read(sock, buffer, LBUF)) > 0)

{

printf(buffer);

memset(buffer, 0, LBUF);

}

if (n < 0)

{

printf("CLIENT: unable to recieve.\n");

exit(1);

}

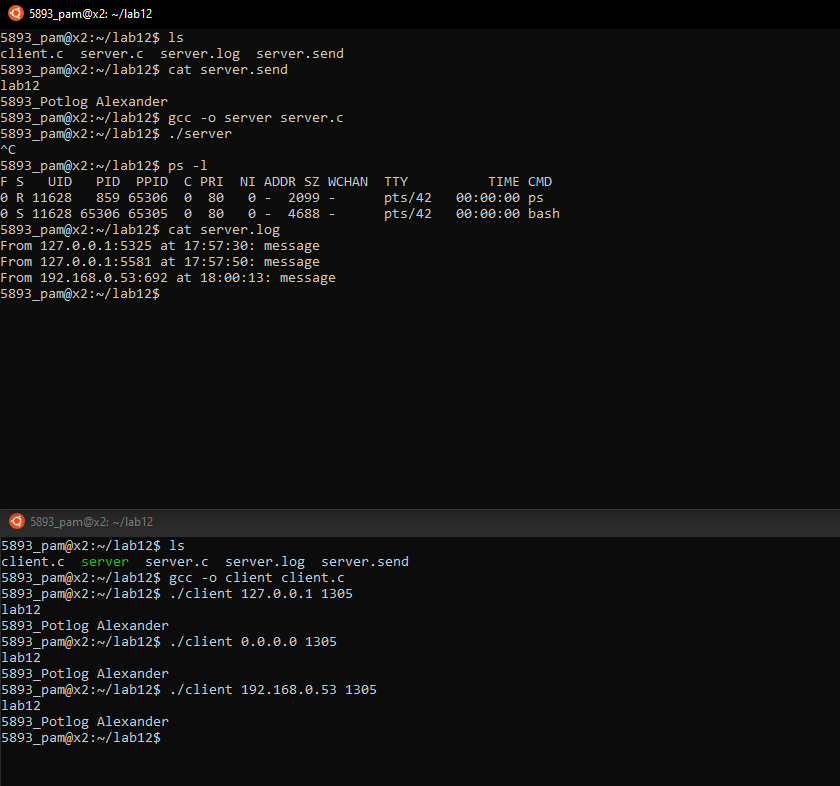
close(sock);

exit(0);

}

1. Совместная отладка своих клиентской и серверной частей в локальном режиме. В процессе данной отладки клиентская и серверная части располагаются на одном и том же хосте, хотя и используют сетевой *TCP*-канал. При этом *TCP*-модуль (входит в состав ОС), обнаружив в качестве адреса назначения свой собственный *IP*-адрес, никакой отправки сообщения по сети не выполняет, а направляет его в указанный локальный сокет.

Результат работы программ:



Вывод: получили навыки создания на основе сокетов виртуальных TCP-соедниний, используемых для связи между удаленными процессами.

127.0.0.1 — это адрес интернет-протокола loopback (IP), также называемый «localhost». Адрес используется для установления соединения с тем же компьютером, который используется конечным пользователем.

В контексте серверов 0.0.0.0 означает «все адреса IPv4 на локальном компьютере».

В последнем случае использовали конкретный ip адресс нашего хоста который мы узнали с помощью команды ip addr show.