**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 № 11 по курсу

«Операционные системы»

**Локальные 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 и использования локальных виртуальных соединений между процессами, используя метод сокетов.

**Ход выполнения работы:**

Создаем клиент client.c который будет выводить текст своего файла и server.c.

Текст программ:

<server.c>

#define \_GNU\_SOURCE

#include <sys/types.h>

#include <sys/un.h>

#include <stdlib.h>

#include <fcntl.h>

#include <ctype.h>

#include <linux/socket.h>

#include <sys/stat.h>

#include <stdio.h>

#include <sys/socket.h>

#define LEN 100

#define MAXCLIENT 6

struct ucred cr;

int main(){

int cl= sizeof(cr);

struct sockaddr\_un server;

int lstner;

char wtmp[] = "/home/newstorage/5893\_pam/lab11/lab11POT.server";

int sendbuff;

lstner = socket(AF\_UNIX, SOCK\_STREAM, 0);

if (lstner < 0) {

perror("Server error socket");

exit(1);

}

unlink(wtmp);

bzero(&server, sizeof(server));

server.sun\_family = AF\_UNIX;

strcpy(server.sun\_path, wtmp);

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

{

perror("Error bind");

exit(2);

}

listen(lstner, MAXCLIENT);

int sock, res;

int bytes\_read;

char buf[LEN];

while (1) {

sock = accept(lstner, NULL, NULL);

if (sock < 0) {

perror("Error accept");

continue;

}

while(1) {

getsockopt(sock, SOL\_SOCKET, SO\_PEERCRED, &cr, &cl);

bytes\_read = recv(sock, buf, LEN, 0);

if (bytes\_read <= 0)

break;

printf("FROM PID %d MESS: %s\n", cr.pid, buf);

}

close(sock);

}

return(0);

}

<client.c>

#include <sys/types.h>

#include <sys/socket.h>

#include <sys/un.h>

#include <stdlib.h>

#include <fcntl.h>

#include <unistd.h>

#include <stdio.h>

#define BUFSIZE 100

char buf[BUFSIZE];

char buf\_msg[BUFSIZE];

ssize\_t nread;

char wtmp[]="/home/newstorage/5893\_pam/lab11/lab11POT.server";

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

{

if(argc !=2)

{

perror("Error not found name file");

exit(1);

}

int sock, fi1;

struct sockaddr\_un server;

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

if(sock<0)

{

perror("Error socket");

exit(1);

}

bzero(&server,sizeof(server));

server.sun\_family = AF\_UNIX;

strcpy(server.sun\_path,wtmp);

int LEN = sizeof(server);

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

{

perror("Error connect");

exit(2);

}

if((fi1 = open(argv[1],O\_RDONLY)) ==-1)

{

perror("Error open file");

exit(1);

}

while((nread = read(fi1,buf\_msg,BUFSIZE))>0)

{

send(sock,buf\_msg,sizeof(buf\_msg),0);

}

printf("Sended: %s\n", buf\_msg);

close(sock);

close(fi1);

return 0;

}

2. Также создаем второй клиент(client2.c) и все компилируем.

3. Запускаем сервер в оперативном режиме.

4. Запускаем два наших клиента в фоновом режиме передав в качестве параметра два разных файла.

Текст файла file:

Potlog

My first

client

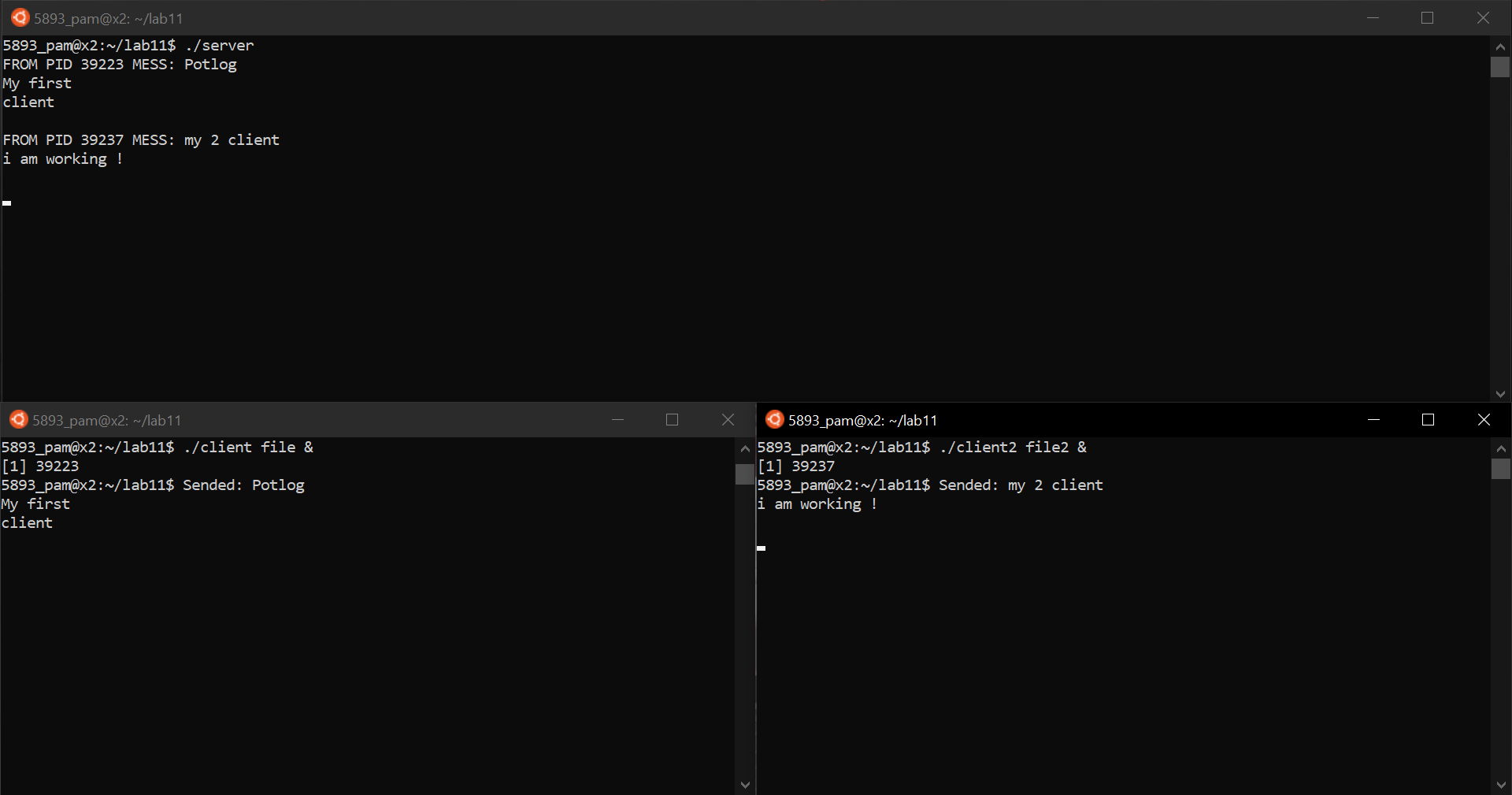
Текст файла file2:

My 2 client

i am working

Результат работы программ:

(Cверху доп. Окно сервера. Снизу справа и слева два окна клиента)



Для того что бы понять где какой клиент, выводим PID каждого клиента.

После работы закрываем все ранее запущенные процессы.

Вывод: На практике получили навыки работы создания и использования локальных виртуальных соединений между процессами при использовании метода сокетов.

На практике узнали чем отличается tcp от udp протокола.