**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 № 9 по курсу

«Операционные системы»

**Датаграммные локальные каналы**

**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 датаграммных информационных каналов между процессами при использовании метода сокетов.

**Ход выполнения работы:**

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

Текст программ:

**<server.c>**

//<server.c>

#include <sys/types.h>

#include <sys/socket.h>

#include <sys/un.h>

#include <stdlib.h>

#include <stdio.h>

#include <errno.h>

#define LBUF 100

#define LEN sizeof(struct sockaddr\_un)

char bufer[LBUF];

main(){

struct sockaddr\_un server, client;

int n, sock, s\_len;

int c\_len = LEN;

if((sock = socket(AF\_UNIX, SOCK\_DGRAM, 0)) < 0){

printf("Server: impossible to create socket.\n");

exit(1);

}

unlink("/home/newstorage/5893\_pam/lab9/lab9.server");

bzero(&server, LEN);

server.sun\_family = AF\_UNIX;

strcpy(server.sun\_path, "/home/newstorage/5893\_pam/lab9/lab9.server");

s\_len = sizeof(server.sun\_family) + strlen(server.sun\_path);

if(bind(sock, (struct sockaddr \*)&server, s\_len) < 0){

printf(strerror(errno));

printf("Server: error in binding.\n");

exit(1);

}

for(;;){

n = recvfrom(sock, bufer, LBUF, 0, (struct sockaddr \*)&client, &c\_len);

if(n < 0){

printf("Server: error in recieving.\n");

continue;

}

printf("From %s: %s\n", client.sun\_path, bufer);

}

}

**<client1.c>**

//<client1.c>

#include <sys/types.h>

#include <sys/socket.h>

#include <sys/un.h>

#include <stdio.h>

#include <stdlib.h>

#define LBUF 100

#define LEN sizeof(struct sockaddr\_un)

char bufer[LBUF];

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

if(argc != 2){

printf("Need file name.\n");

exit(1);

}

char \*estr;

FILE \*fil;

if((fil = fopen(argv[1], "r")) == NULL){

printf("Cannot open the file.\n");

exit(1);

}

struct sockaddr\_un server, client;

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

if((sock = socket(AF\_UNIX, SOCK\_DGRAM, 0)) < 0){

printf("Client: impossible to create socket.\n");

exit(1);

}

bzero(&client, LEN);

client.sun\_family = AF\_UNIX;

strcpy(client.sun\_path, "/home/newstorage/5893\_pam/lab9/client1Pot");

//mktemp(client.sun\_path);

c\_len = sizeof(client.sun\_family) + strlen(client.sun\_path);

if(bind(sock, (struct sockaddr \*)&client, c\_len) < 0){

printf("Client: error in binding\n");

exit(1);

}

bzero(&server, LEN);

server.sun\_family = AF\_UNIX;

strcpy(server.sun\_path, "/home/newstorage/5893\_pam/lab9/lab9.server");

s\_len = sizeof(server.sun\_family) + strlen(server.sun\_path);

while(1){

estr = fgets(bufer, sizeof(bufer), fil);

if(estr != NULL){

if(sendto(sock, bufer, LBUF, 0, (struct sockaddr \*)&server, s\_len) != LBUF){

printf("Client: error in sending.\n");

exit(1);

}

}

else exit(1); //условие выхода из бесконечного цикла

}

fclose(fil);

close(sock);

unlink(client.sun\_path);

exit(0);

}

2. Также создаем второй клиент(client2.c) поменяв ему поле client.sun\_path.

("/home/newstorage/5893\_pam/lab9/POT1")

3. Запускаем сервер в оперативном режиме.

4. Запускаем два наших клиента передав в качестве параметра два разных файла.(каждая строка файла будет отправляться отдельной датаграммой)

Текст файла file1:

potlog

5893 pervi client

Текст файла file2:

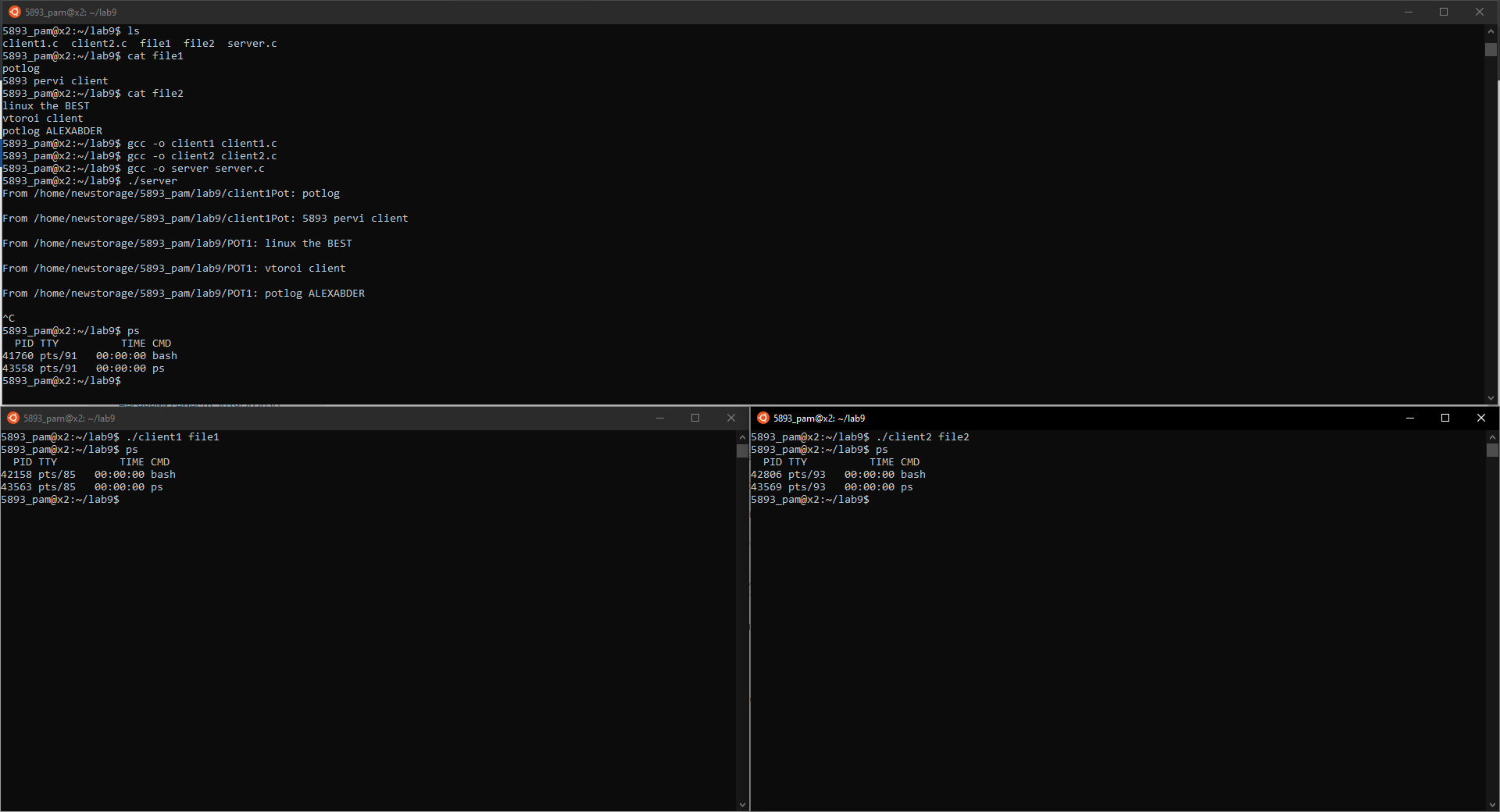
linux the BEST

vtoroi client

potlog ALEXABDER

Результат работы программ:

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



Для того что бы понять где какой клиент, выводим имя-путь сокета каждого клиента, сервер его получает после отправки датаграммы клиентом, вместе с полезной информацией(служебная инф.). (имя сокета должно быть разным в пределах системы).

5. После окончания работы закрываем сервер и проверяем нет ли у нас лишних процессов, с помощью утилиты ps. (клиенты закрываются сами, как видно на скрине).

Вывод: На практике получили навыки работы создания и использования локальных датаграммных информационных каналов между процессами при использовании метода сокетов.