Лабораторная работа 15

Отчёт

Новосельцев Данила Сергеевич

Содержание

Лабораторная работа 15

Новосельцев.Д.С. НФИбд-02-20

Цель работы: приобретение практических навыков работы с сокетами.

Ход работы:

1.Изучил приведённые в тексте программы server.c и client.c.

common.h:

#ifndef **COMMON_H** #define **COMMON_H** #include <stdio.h> #include <stdlib.h> #include <string.h> #include <errno.h> #include <sys/types.h> #include <sys/stat.h> #include <fcntl.h> #define FIFO_NAME “/tmp/fifo” #define MAX_BUFF 80 #endif

server.c:

#include “common.h” int main() { int readfd; /* дескриптор для чтения из FIFO */ *int n; char buff[MAX_BUFF]; /* буфер для чтения данных из FIFO */ /* баннер */ printf(“FIFO Server...”); /* создаем файл FIFO с открытыми для всех * правами доступа на чтение и запись */ if(mknod(FIFO_NAME, S_IFIFO | 0666, 0) < 0) { fprintf(stderr, “%s: Невозможно создать FIFO (%s)”,* ***FILE****, strerror(errno)); exit(-1); } /* откроем FIFO на чтение */ if((readfd = open(FIFO_NAME, O_RDONLY)) < 0) { fprintf(stderr, “%s: Невозможно открыть FIFO (%s)”,* ***FILE****, strerror(errno)); exit(-2); } /* читаем данные из FIFO и выводим на экран */ while((n = read(readfd, buff, MAX_BUFF)) > 0) { if(write(1, buff, n) != n) { fprintf(stderr, “%s: Ошибка вывода (%s)”,* ***FILE****, strerror(errno)); exit(-3); } } close(readfd); /* закроем FIFO */ /* удалим FIFO из системы */* if(unlink(FIFO_NAME) < 0) { fprintf(stderr, “%s: Невозможно удалить FIFO (%s)”, **FILE**, strerror(errno)); exit(-4); } exit(0); }

client.c:

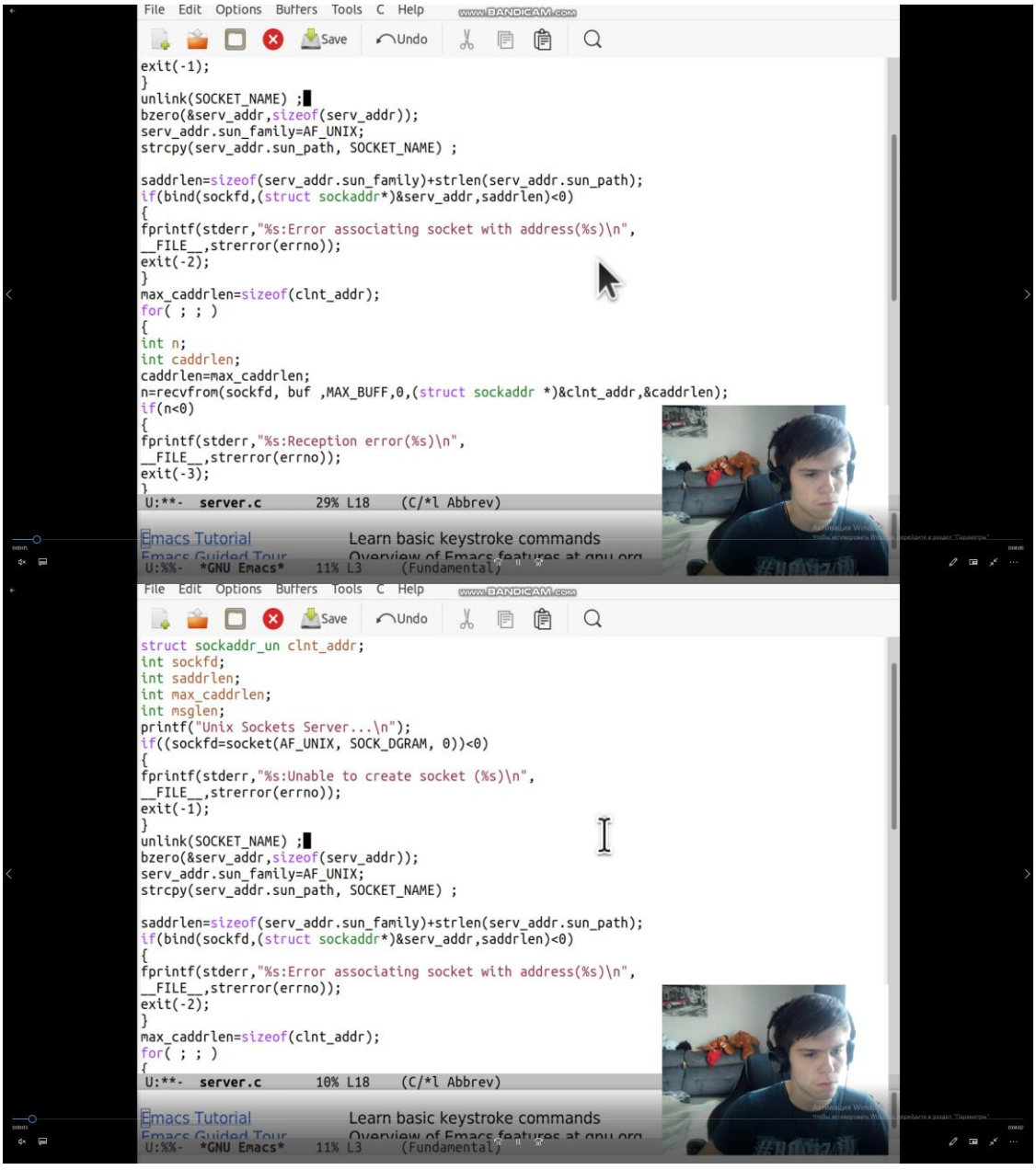
#include “common.h” #define MESSAGE “Hello Server!!!” int main() { int writefd; /* дескриптор для записи в FIFO */ int msglen; /* баннер */ printf(“FIFO Client...”); /* получим доступ к FIFO */ if((writefd = open(FIFO_NAME, O_WRONLY)) < 0) { fprintf(stderr, “%s: Невозможно открыть FIFO (%s)”, ***FILE***, strerror(errno)); exit(-1); } /* передадим сообщение серверу */ msglen = strlen(MESSAGE); if(write(writefd, MESSAGE, msglen) != msglen) { fprintf(stderr, “%s: Ошибка записи в FIFO (%s)”, ***FILE***, strerror(errno)); exit(-2); } /* закроем доступ к FIFO */ close(writefd); exit(0); }

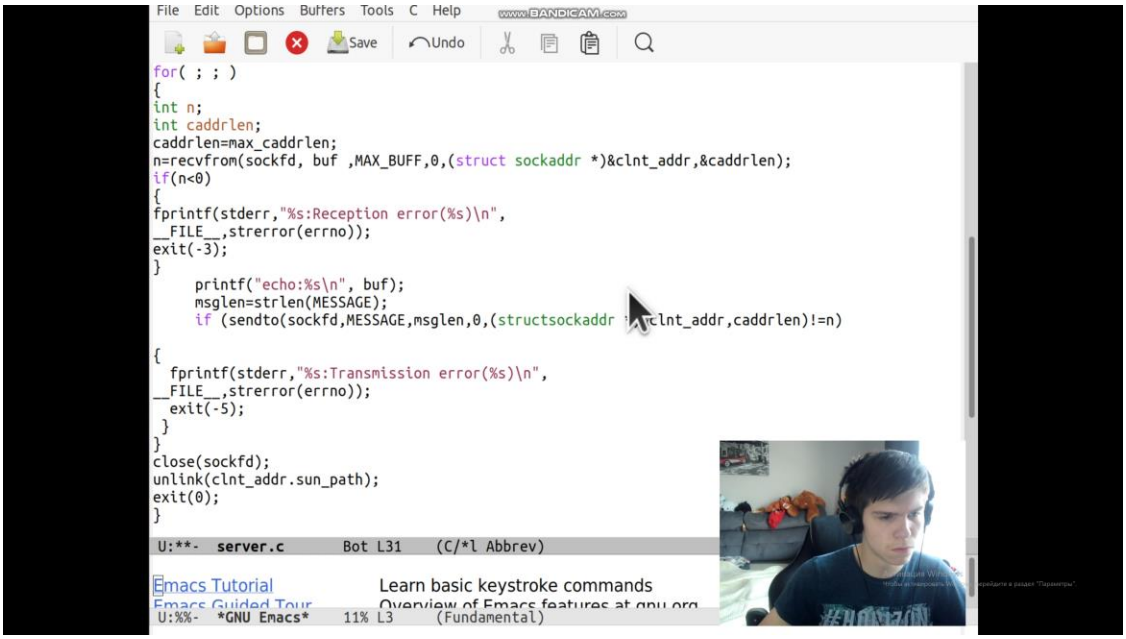
makefile:

all: server client server: server.c common.h gcc server.c -o server client: client.c common.h gcc client.c -o client clean: -rm server client *.o

2.Взяв данные примеры за образец, попробовал написать аналогичные программы, внеся следующие изменения:

server.c:


File Edit Options Buffers Tools C Help
www.BANDICAM.com
Save
Undo
exit(-1);
}
unlink(SOCKET_NAME) ;
bzero(&serv_addr,sizeof(serv_addr));
serv_addr.sun_family=AF_UNIX;
strcpy(serv_addr.sun_path, SOCKET_NAME) ;
saddrlen=sizeof(serv_addr.sun_family)+strlen(serv_addr.sun_path);
if(bind(sockfd,(struct sockaddr*)&serv_addr,saddrlen)<0)
{
fprintf(stderr,"%s:Error associating socket with address(%s)\n",
__FILE__,strerror(errno));
exit(-2);
}
max_caddrlen=sizeof(clnt_addr);
for(; ;)
{
int n;
int caddrlen;
caddrlen=max_caddrlen;
n=recvfrom(sockfd, buf ,MAX_BUFF,0,(struct sockaddr *)&clnt_addr,&caddrlen);
if(n<0)
{
fprintf(stderr,"%s:Reception error(%s)\n",
__FILE__,strerror(errno));
exit(-3);
}
U:**- server.c 29% L18 (C/*l Abbrev)
Emacs Tutorial
Learn basic keystroke commands
Emacs Guided Tour
Overview of Emacs features at gnu.org
U:%%- *GNU Emacs* 11% L3 (Fundamental)
File Edit Options Buffers Tools C Help
www.BANDICAM.com
Save
Undo
struct sockaddr_un clnt_addr;
int sockfd;
int saddrlen;
int max_caddrlen;
int msglen;
printf("Unix Sockets Server...\n");
if((sockfd=socket(AF_UNIX, SOCK_DGRAM, 0))<0)
{
fprintf(stderr,"%s:Unable to create socket (%s)\n",
__FILE__,strerror(errno));
exit(-1);
}
unlink(SOCKET_NAME) ;
bzero(&serv_addr,sizeof(serv_addr));
serv_addr.sun_family=AF_UNIX;
strcpy(serv_addr.sun_path, SOCKET_NAME) ;
saddrlen=sizeof(serv_addr.sun_family)+strlen(serv_addr.sun_path);
if(bind(sockfd,(struct sockaddr*)&serv_addr,saddrlen)<0)
{
fprintf(stderr,"%s:Error associating socket with address(%s)\n",
__FILE__,strerror(errno));
exit(-2);
}
max_caddrlen=sizeof(clnt_addr);
for(; ;)
{
U:**- server.c 10% L18 (C/*l Abbrev)
Emacs Tutorial
Learn basic keystroke commands
Emacs Guided Tour
Overview of Emacs features at gnu.org
U:%%- *GNU Emacs* 11% L3 (Fundamental)


File Edit Options Buffers Tools C Help
www.BANDICAM.com
Save
Undo
for(; ;)
{
int n;
int caddrlen;
caddrlen=max_caddrlen;
n=recvfrom(sockfd, buf ,MAX_BUFF,0,(struct sockaddr *)&clnt_addr,&caddrlen);
if(n<0)
{
fprintf(stderr,"%s:Reception error(%s)\n",
__FILE__,strerror(errno));
exit(-3);
}
printf("echo:%s\n", buf);
msglen=strlen(MESSAGE);
if (sendto(sockfd,MESSAGE,msglen,0,(structsockaddr *)&clnt_addr,caddrlen)!=n)
{
fprintf(stderr,"%s:Transmission error(%s)\n",
__FILE__,strerror(errno));
exit(-5);
}
}
close(sockfd);
unlink(clnt_addr.sun_path);
exit(0);
}
U:**- server.c Bot L31 (C/*l Abbrev)
Emacs Tutorial
Learn basic keystroke commands
Emacs Guided Tour
Overview of Emacs features at gnu.org
U:%%- *GNU Emacs* 11% L3 (Fundamental)

client.c:

File Edit Options Buffers Tools C Help

Save Undo

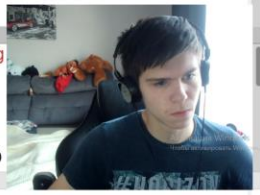
```
#include "common.h"
#define MESSAGE "Hello Server!!!"
int main ()
{
char buf[MAX_BUFF];
struct sockaddr_un serv_addr;
struct sockaddr_un clnt_addr;
int sockfd;
int saddrlen;
int caddrlen;
int msglen;
int n;
bzero(&serv_addr, sizeof(serv_addr));
serv_addr.sun_family=AF_UNIX;
strcpy(serv_addr.sun_path,SOCKET_NAME) ;
saddrlen=sizeof(serv_addr.sun_family)+strlen(serv_addr.sun_path);

bzero(&clnt_addr, sizeof(clnt_addr));
clnt_addr.sun_family=AF_UNIX;
strcpy(clnt_addr.sun_path,"/tmp/clnt.XXXXXX");
mktemp(clnt_addr.sun_path);
caddrlen=sizeof(clnt_addr.sun_family)+strlen(clnt_addr.sun_path);

if ((sockfd=socket(AF_UNIX,SOCK_DGRAM,0))<0)
```

U:**- client.c Top L13 (C/*l Abbrev)

Welcome to GNU Emacs, one component of the GNU/Linux operating

Emacs Tutorial	Learn basic keystroke commands
Emacs Guided Tour	Overview of Emacs features at gnu.org
View Emacs Manual	View the Emacs manual using Info
Absence of Warranty	GNU Emacs comes with ABSOLUTELY NO

U:%%- *GNU Emacs* Top L3 (Fundamental)



File Edit Options Buffers Tools C Help

Save Undo

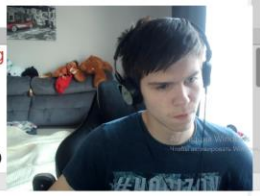
```
__FILE__,strerror(errno));
exit(-1);
}
if(bind(sockfd,(struct sockaddr *)&clnt_addr,caddrlen)<0)
{
fprintf(stderr,"%s:Error associating socket with address(%s)\n",
__FILE__,strerror(errno));
exit(-2);
}
msglen=str lLen(MESSAGE) ;
if(sendto(sockfd,MESSAGE,msglen,0,(struct sockaddr *)&serv_addr,saddrlen)!=msglen)
{
fprintf(stderr,"%s: Transmission error(%s)\n",
__FILE__,strerror(errno));
exit(-3);
}
memset (buf ,0,MAX_BUFF);
if((n=recvfrom(sockfd, buf ,MAX_BUFF ,0,NULL,0))<0)
{
fprintf(stderr,"%s:Reception error(%s)\n",
__FILE__,strerror(errno));
exit(-4);
}
```

U:**- client.c 52% L26 (C/*l Abbrev)

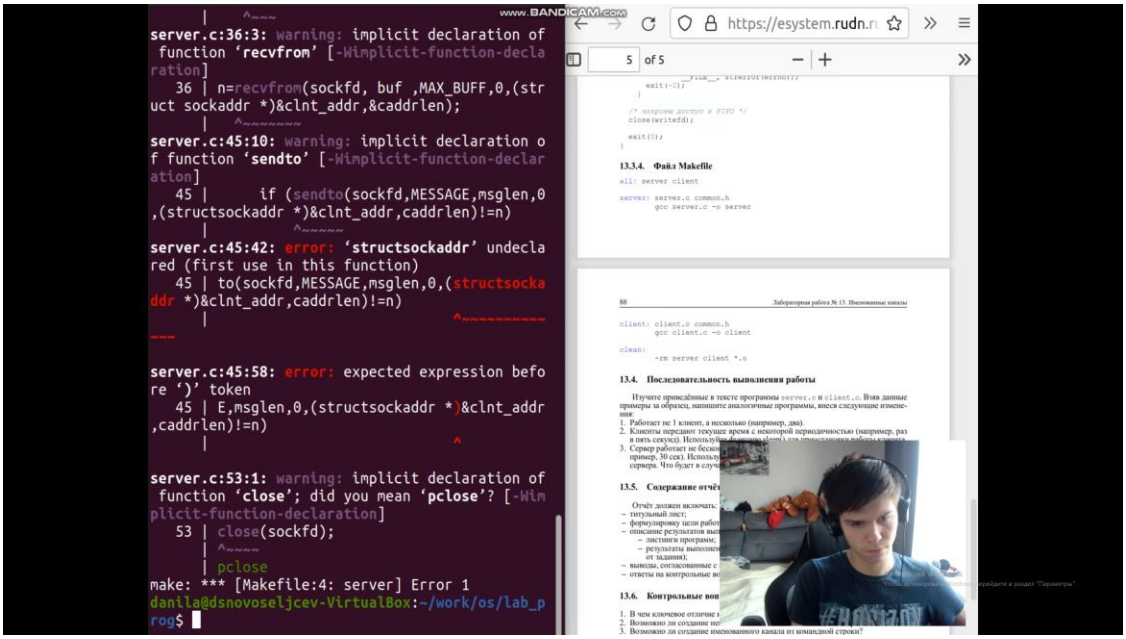
Welcome to GNU Emacs, one component of the GNU/Linux operating

Emacs Tutorial	Learn basic keystroke commands
Emacs Guided Tour	Overview of Emacs features at gnu.org
View Emacs Manual	View the Emacs manual using Info
Absence of Warranty	GNU Emacs comes with ABSOLUTELY NO

U:%%- *GNU Emacs* Top L3 (Fundamental)



Итог:



Вывод: не смог приобрести практические навыки работы с сокетами.

Ответы на контрольные вопросы:

1. BSD является сокращением от ‘Berkeley Software Distribution’, названия, которое было выбрано Berkeley CSRG (Computer Systems Research Group) для их дистрибутива Unix.

2. Сокет (socket) - это конечная точка сетевых коммуникаций. Он является чем-то вроде “портала”, через которое можно отправлять байты во внешний мир. Приложение просто пишет данные в сокет. Программирование сокетов в Linux , их дальнейшая буферизация, отправка и транспортировка осуществляется используемым стеком протоколов и сетевой аппаратурой. Чтение данных из сокета происходит аналогичным образом. В программе сокет идентифицируется дескриптором - это просто переменная типа int. Программа получает дескриптор от операционной системы при создании сокета, а затем передаёт его сервисам socket API для указания сокета, над которым необходимо выполнить то или иное действие

3.Именованные каналы, описанные в главе 11, очень похожи на сокеты, но в способах их использования имеются значительные различия.

· Именованные каналы могут быть ориентированными на работу с сообщениями, что значительно упрощает программы.

· Именованные каналы требуют использования функций ReadFile и WriteFile, в то время как сокеты могут обращаться также к функциям send и recv.

· В отличие от именованных каналов сокеты настолько гибки, что предоставляют пользователям возможность выбрать протокол для использования с сокетом, например, TCP или UDP. Кроме того, пользователь имеет возможность выбирать протокол на основании характера предоставляемой услуги или иных факторов.

· Сокеты основаны на промышленном стандарте, что обеспечивает их совместимость с системами, отличными от Windows.

Имеются также различия в моделях программирования сервера и клиента.

4. Коммуникационный домен определяет форматы адресов и правила их интерпретации. Внутри них существуют сокеты.

5. Виды сокетов:

· Сокеты в файловом пространстве имён (file namespace, сокеты Unix) используют в

качестве адресов имена файлов специального типа.

· Сокеты в файловом пространстве имён похожи на именованные каналы тем, что для идентификации сокетов используются файлы специального типа. В мире сокетов есть и аналог неименованных каналов — парные сокеты.

· Сетевой сокет – сокет, в котором формат адреса имеет вид ip(7). Поскольку адрес транспортного уровня состоит из пары ip-адрес: порт, то и в структуре

под адрес отводится два поля.

6. Когда поддержка BSD сокетов были добавлена в ядро Linux, разработчики решили добавить их единовременно все 17 (на сегодня 20) сокетных вызовов, и добавили для этих вызовов один дополнительный уровень косвенности. Для всей группы этих вызовов введен один новый, редко упоминаемый, системный вызов:

int socketcall(int call, unsigned long *args),

где:

— call — численный номер сетевого вызова (SYS_CONNECT, SYS_ACCEPT…);

— args — указатель 6-ти элементного массива (блок параметров), в который последовательно упакованы все параметры любого из системных вызовов этой группы (сетевой), без различения их типа (приведенные к unsigned long)

7. Базовая эталонная модель взаимодействия открытых систем (БЭМВОС) – это концептуальная основа, определяющая характеристики и средства открытых систем. Она обеспечивает работу в одной сети систем, выпускаемых различными производителями. Разработана ISO (международной организацией стандартов) и широко используется во всём мире как основа концепций информационных сетей и их ассоциаций. На базе этой модели описываются правила и процедуры передачи данных между открытыми

системами. Она также описывает структуру открытой системы и комплекс стандартов, которым она должна удовлетворять. Основными элементами модели являются: уровни, объекты, соединения, физические средства соединений.

Модель информационной системы состоит из трёх основных составляющих:

- прикладные процессы (осуществляют обработку данных);
- область взаимодействия (размещаемые в ней блоки прокладывают в сети логические каналы (пунктирная линия на рисунке) между портами прикладных процессов и обеспечивает их взаимодействие);
- физические средства соединений (обеспечивают физическую связь систем).