|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

**ОТЧЕТ**

*к лабораторной работе №6*

*По курсу: «Операционные системы»*

Студент ИУ7-66Б

Турсунов Ж.Р.

Преподаватель

Рязанова Н.Ю.

*Москва, 2021 г.*

1. СОКЕТ В ФАЙЛОВОМ ПРОСТРАНСТВЕ

Сокеты в файловом пространстве используют в качестве адреса имяфайла. В данной программе сервер создает сокет при помощи функции socket (с параметрами AF\_UNIX, что означает, что сокет будет в файловомпространстве, и SOCK\_DGRAM, что означает, что сокет будет датараммным) исвязывает его с файлом socket.soc при помощи функции bind. После этогоможно получать сообщения от клиентов при помощи функции recvfrom.

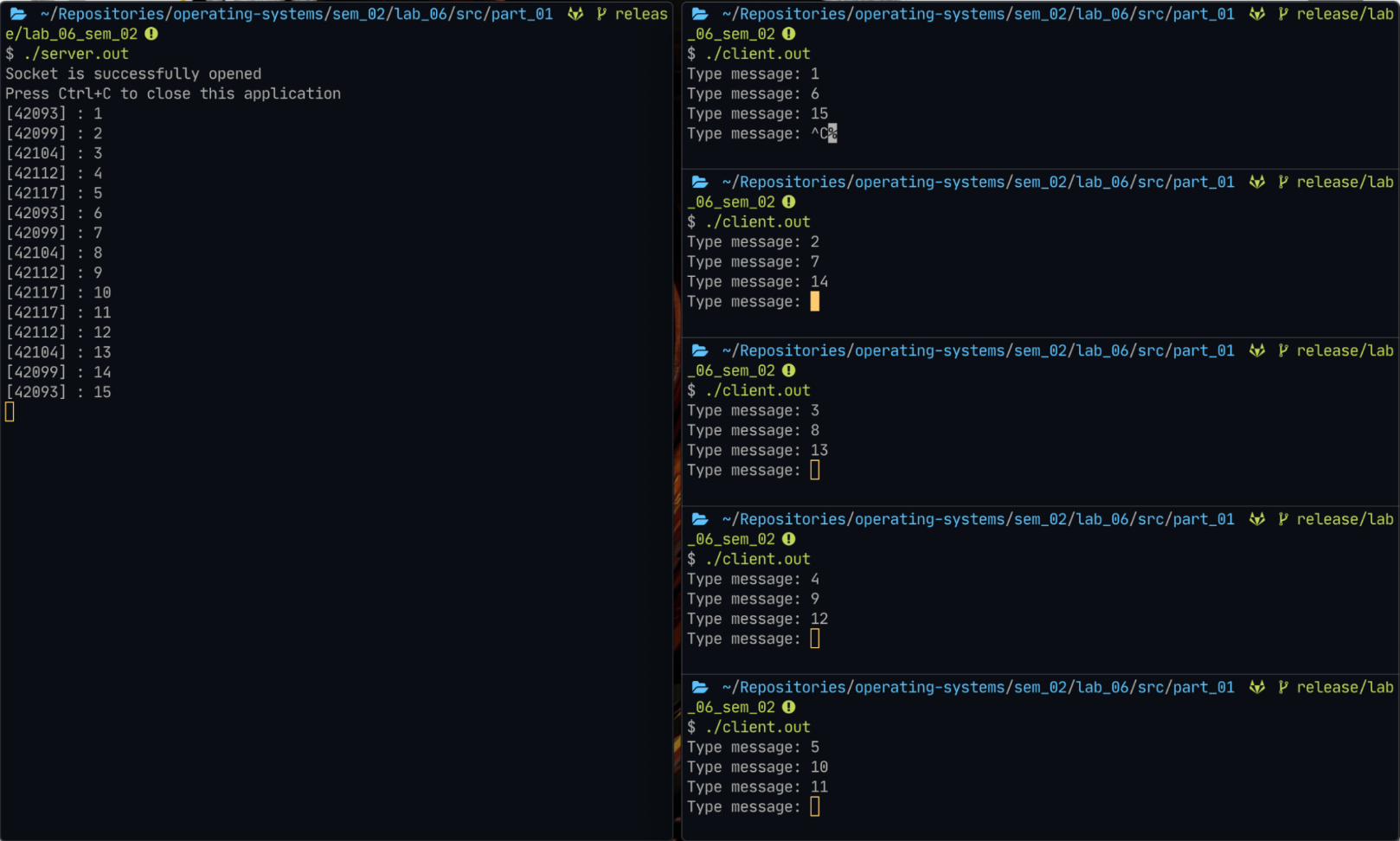
**Текст программы сервера:**

|  |
| --- |
| #include <string.h>  #include <sys/types.h>  #include <sys/socket.h>  #include <signal.h>  #include <stdlib.h>  #include <unistd.h>  #include "error.h"  #define SOCK\_NAME "socket.soc"  #define SIZE\_BUFFER 100  int sockfd**;**  void close\_app**()**  **{**  **if** **(**close**(**sockfd**)** **<** 0**)**  exit**(**error**());**  **if** **(**unlink**(**SOCK\_NAME**)** **<** 0**)**  exit**(**error**());**  printf**(**"\nSocket closed\n"**);**  exit**(**0**);**  **}**  int main**()**  **{**  struct sockaddr socket\_file**;**  struct sockaddr recv\_name**;**  socklen\_t recv\_len**;**  char buffer**[**SIZE\_BUFFER**];**  ssize\_t size**;**  signal**(**SIGINT**,** close\_app**);**  sockfd **=** socket**(**AF\_UNIX**,** SOCK\_DGRAM**,** 0**);**  **if** **(**sockfd **<** 0**)**  **return** error**();**  socket\_file**.**sa\_family **=** AF\_UNIX**;**  strcpy**(**socket\_file**.**sa\_data**,** SOCK\_NAME**);**  bind**(**sockfd**,** **&**socket\_file**,** strlen**(**socket\_file**.**sa\_data**)** **+** **sizeof(**socket\_file**.**sa\_family**)** **+** 1**);**  **if** **(**errno **!=** 0**)**  **return** error**();**  printf**(**"Socket is successfully opened\n"**);**  printf**(**"Press Ctrl+C to close this application\n"**);**  **while** **(**1**)**  **{**  size **=** recvfrom**(**sockfd**,** buffer**,** **sizeof(**buffer**),** 0**,** **&**recv\_name**,** **&**recv\_len**);**  **if** **(**size **<** 0**)**  **return** error**();**  buffer**[**size**]** **=** '\0'**;**  printf**(**"%s"**,** buffer**);**  **}**  **return** 0**;**  **}** |

Клиент создает сокет с такими же параметрами, как и сервер, после чегозадает тип домена и имя файла в структуру sockaddr, с помощью которойи отправляе соощение серверу фукцией sendto.

**Текст программы клиента:**

|  |
| --- |
| #include <sys/types.h>  #include <sys/socket.h>  #include <unistd.h>  #include <signal.h>  #include <stdlib.h>  #include "error.h"  #define SIZE\_BUFFER 100  int sockfd**;**  void close\_app**()**  **{**  **if** **(**close**(**sockfd**)** **<** 0**)**  exit**(**error**());**  exit**(**0**);**  **}**  int main**()**  **{**  size\_t len**;**  struct sockaddr srvr\_name**;**  char **\***buffer**;**  char message**[**SIZE\_BUFFER**];**  signal**(**SIGINT**,** close\_app**);**  sockfd **=** socket**(**AF\_UNIX**,** SOCK\_DGRAM**,** 0**);**  **if** **(**sockfd **<** 0**)**  **return** error**();**  srvr\_name**.**sa\_family **=** AF\_UNIX**;**  strcpy**(**srvr\_name**.**sa\_data**,** "socket.soc"**);**  **while(**1**)**  **{**  printf**(**"Type message: "**);**  getline**(&**buffer**,** **&**len**,** stdin**);**  sprintf**(**message**,** "[%d] : %s"**,** getpid**(),** buffer**);**  sendto**(**sockfd**,** message**,** strlen**(**message**),** 0**,** **&**srvr\_name**,** strlen**(**srvr\_name**.**sa\_data**)** **+** **sizeof(**srvr\_name**.**sa\_family**)** **+** 1**);**  **}**  **return** 0**;**  **}** |

**Результат работы программы**

1. СЕТЕВОЙ СОКЕТ

Сервер создает сокет с параметрами AF\_INET(сетевое соединение) и SOCK\_STREAM (потоковый сокет). С помощью функции bind сокет связывается с адре-сом. После этого используется функция listen, которая переводит сервер в режим ожидания запроса на соединение. Затем создается массив дескрипторов для будущих подключаемых клиентов. Если дескриптор равен -1, то значит он свободен для новых клиентов. После проделанных действий можно ожидать подключения от клиентов. В функции connection, которая служит в этой программе для подключения новых клиентов, с помощью функции accept устанавливается соединение, затем находим свободный элемент в массиве дескрипторов и добавляем нового клиента, получив от него пер-вое сообщение, содержащее его pid. Функция message принимает сообщения от подключенных клиентов и проверяет на отключение от сервера. В этой функции производится проверка дескрипторов всех подключенных клиентов. Если появляется возможность чтения или записи из дескриптора, то происходит проверка на соединение с клиентом. Если соединения нет, то освобождается дескриптор и выводится сообщение об отключении клиента. В ином случае выводится полученное сообщение и отправляется ответное.

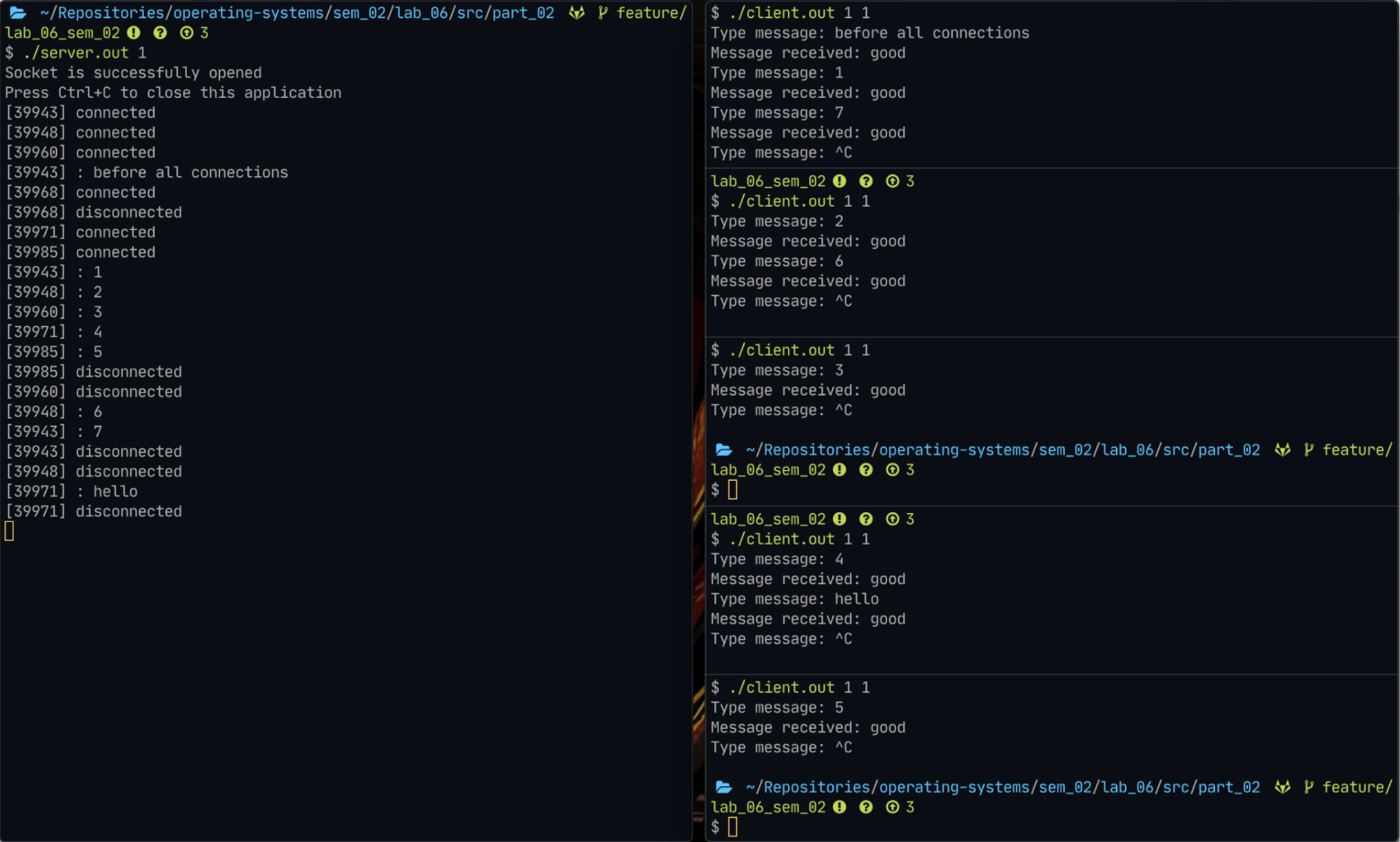
**Текст программы сервера:**

|  |
| --- |
| #include <sys/types.h>  #include <sys/socket.h>  #include <unistd.h>  #include <stdlib.h>  #include <strings.h>  #include <netinet/in.h>  #include <signal.h>  #include "error.h"  #define SIZE\_BUFFER 100  #define CLIENTS 100  int sockfd**;**  int maxi**,** maxfd**;**  int pid\_client**[**CLIENTS**];**  void close\_signal**()**  **{**  **if** **(**close**(**sockfd**)** **<** 0**)**  exit**(**error**());**  printf**(**"\nSocket closed\n"**);**  exit**(**0**);**  **}**  int connection**(**int client**[**FD\_SETSIZE**],** fd\_set **\***allset**,** fd\_set **\***rset**)**  **{**  int i **=** 0**;**  int connfd**;**  int message\_len**;**  char buffer**[**SIZE\_BUFFER**];**  **if** **(**FD\_ISSET**(**sockfd**,** rset**))**  **{**  connfd **=** accept**(**sockfd**,** **NULL,** **NULL);**  **if** **(**connfd **<** 0**)**  **return** errno**;**  **do**  **{**  **if** **(**client**[**i**]** **<** 0**)**  client**[**i**]** **=** connfd**;**  i**++;**  **}**  **while(**client**[**i**]** **>=** 0**);**  **if** **(**i **==** FD\_SETSIZE**)**  **return** errno**;**  FD\_SET**(**connfd**,** allset**);**  **if** **(**connfd **>** maxfd**)**  maxfd **=** connfd**;**  **if** **(**i **>** maxi**)**  maxi **=** i**;**  message\_len **=** read**(**connfd**,** buffer**,** SIZE\_BUFFER**);**  pid\_client**[**i**]** **=** atoi**(**buffer**);**  printf**(**"[%d] connected\n"**,** pid\_client**[**i**]);**  **}**  **return** 0**;**  **}**  int message**(**int client**[**FD\_SETSIZE**],** fd\_set **\***allset**,** fd\_set **\***rset**)**  **{**  int n**,** i**;**  int sockfd**;**  char buffer**[**SIZE\_BUFFER**];**  **for** **(**i **=** 0**;** i **<=** maxi**;** i**++)**  **{**  sockfd **=** client**[**i**];**  **if** **(**sockfd **>** 0**)**  **{**  **if** **(**FD\_ISSET**(**sockfd**,** rset**))**  **{**  n **=** read**(**sockfd**,** buffer**,** SIZE\_BUFFER**);**  **if** **(**n **==** 0**)**  **{**  close**(**sockfd**);**  FD\_CLR**(**sockfd**,** allset**);**  client**[**i**]** **=** **-**1**;**  printf**(**"[%d] disconnected\n"**,** pid\_client**[**i**]);**  **}**  **else**  **{**  write**(**sockfd**,** "good"**,** 4**);**  printf**(**"[%d] : %s"**,** pid\_client**[**i**],** buffer**);**  **}**  **}**  **}**  **}**  **return** 0**;**  **}**  int main**(**int argc**,** char **\*\***argv**)**  **{**  **if** **(**argc **!=** 2**)**  **{**  fprintf**(**stderr**,** "Usage: %s <port>\n"**,** argv**[**0**]);**  **return** **-**1**;**  **}**  int client**[**FD\_SETSIZE**];**  fd\_set rset**;**  fd\_set allset**;**  struct sockaddr\_in server**;**  signal**(**SIGINT**,** close\_signal**);**  sockfd **=** socket**(**AF\_INET**,** SOCK\_STREAM**,** 0**);**  **if** **(**sockfd **<** 0**)**  **return** errno**;**  server**.**sin\_family **=** AF\_INET**;**  server**.**sin\_addr**.**s\_addr **=** INADDR\_ANY**;**  server**.**sin\_port **=** htons**(**atoi**(**argv**[**1**]));**  bind**(**sockfd**,** **(**struct sockaddr **\*)** **&**server**,** **sizeof(**server**));**  **if** **(**errno **!=** 0**)**  **return** errno**;**  listen**(**sockfd**,** CLIENTS**);**  maxfd **=** sockfd**;**  maxi **=** **-**1**;**  **for** **(**int i **=** 0**;** i **<** FD\_SETSIZE**;** i**++)**  client**[**i**]** **=** **-**1**;**  FD\_ZERO**(&**allset**);**  FD\_SET**(**sockfd**,** **&**allset**);**  printf**(**"Socket is successfully opened\n"**);**  printf**(**"Press Ctrl+C to close this application\n"**);**  **while(**1**)**  **{**  rset **=** allset**;**  select**(**maxfd **+** 1**,** **&**rset**,** **NULL,** **NULL,** **NULL);**  connection**(**client**,** **&**allset**,** **&**rset**);**  **if** **(**errno **!=** 0**)**  **return** error**();**  message**(**client**,** **&**allset**,** **&**rset**);**  **if** **(**errno **!=** 0**)**  **return** error**();**  **}**  **return** 0**;**  **}** |

Клиент создает сокет с теми же параметрами, как и сервер. После этого, клиенту необходимо подключиться по адресу. Это делается с помощью структуры sockaddr\_in, поля которой необходимо заполнить информацией о сервере, к которому мы хотим подключиться. Функцией connect производится подключение. После успешного подключения, клиент отправляет свой pid сообщением и начинает отправлять сообщения, введенные пользователем. После каждого отправленного сообщения клиент ждет ответного сообщения от сервера. Отправка сообщения производится функцией write, получение –read.

**Текст программы клиента:**

|  |
| --- |
| #include <stdint.h>  #include <stdio.h>  #include <sys/types.h>  #include <sys/socket.h>  #include <unistd.h>  #include <netinet/in.h>  #include <netdb.h>  #include <stdlib.h>  #include <signal.h>  #include "error.h"  #define SIZE\_BUFFER 100  int sockfd**;**  void close\_signal**()**  **{**  **if** **(**close**(**sockfd**)** **<** 0**)**  exit**(**error**());**  printf**(**"\nSocket closed\n"**);**  exit**(**0**);**  **}**  int main**(**int argc**,** char **\***argv**[])**  **{**  **if** **(**argc **!=** 3**)**  **{**  printf**(**"Usage: %s <servername> <port>"**,** argv**[**0**]);**  **return** **-**1**;**  **}**  signal**(**SIGINT**,** close\_signal**);**  struct hostent **\***server**;**  struct sockaddr\_in serv\_addr**;**  char **\***buffer **=** **NULL;**  size\_t len**;**  char buffer\_server**[**SIZE\_BUFFER**];**  sockfd **=** socket**(**AF\_INET**,** SOCK\_STREAM**,** 0**);**  **if** **(**sockfd **<** 0**)**  **return** error**();**  server **=** gethostbyname**(**argv**[**1**]);**  **if** **(**server **==** **NULL)**  **return** error**();**  serv\_addr**.**sin\_family **=** AF\_INET**;**  strncpy**((**char **\*)&**serv\_addr**.**sin\_addr**.**s\_addr**,** **(**char **\*)**server**->**h\_addr**,** server**->**h\_length**);**  serv\_addr**.**sin\_port **=** htons**(**atoi**(**argv**[**2**]));**  connect**(**sockfd**,** **(**struct sockaddr **\*)&**serv\_addr**,** **sizeof(**serv\_addr**));**  **if** **(**errno **!=** 0**)**  **return** error**();**  buffer **=** calloc**(**10**,** 1**);**  sprintf**(**buffer**,** "%d"**,** getpid**());**  write**(**sockfd**,** buffer**,** strlen**(**buffer**));**  free**(**buffer**);**  buffer **=** **NULL;**  printf**(**"Type message: "**);**  getline**(&**buffer**,** **&**len**,** stdin**);**  buffer**[**len**]** **=** '\0'**;**  **while** **(**strcmp**(**buffer**,** "exit\n"**))**  **{**  write**(**sockfd**,** buffer**,** len**);**  memset**(**buffer\_server**,** 0**,** SIZE\_BUFFER**);**  read**(**sockfd**,** buffer\_server**,** SIZE\_BUFFER**);**  printf**(**"Message received: %s\n"**,** buffer\_server**);**  printf**(**"Type message: "**);**  getline**(&**buffer**,** **&**len**,** stdin**);**  buffer**[**len**]** **=** '\0'**;**  **}**  **return** 0**;**  **}** |

**Результат работы программы**