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»

Лабораторная работа № 6
По курсу «Операционные системы»

Тема Сокеты

Студент Неклепаева А. Н.

Группа ИУ7-63б

Преподаватель Рязанова Н. Ю.

Москва.
2020 г.

Задание

1. Организовать взаимодействие параллельных процессов на отдельном компьютере.

2. Организовать взаимодействие параллельных процессов в сети (ситуацию моделируем на одной машине).

Задание 1

• Написать приложение по модели клиент-сервер, демонстрирующее взаимодействие параллельных процессов на отдельном компьютере с использованием сокетов в файловом пространстве имен: семейство - AF_UNIX, тип - SOCK_DGRAM. При демонстрации работы программного комплекса необходимо запустить несколько клиентов (не меньше 5) и продемонстрировать, что сервер обрабатывает обращения каждого запущенного клиента.

Задание 2

• Написать приложение по модели клиент-сервер, осуществляющее взаимодействие параллельных процессов, которые выполняются на разных компьютерах. Для взаимодействия с клиентами сервер должен использовать мультиплексирование. Сервер должен обслуживать запросы параллельно запущенных клиентов. При демонстрации работы программного комплекса необходимо запустить несколько клиентов (не меньше 5) и продемонстрировать, что сервер обрабатывает обращения каждого запущенного клиента.

Задание №1

Листинг 1: Код сервера fsserver.c

```c
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <unistd.h>
#include <signal.h>

#define SOCK_NAME "socket.soc"
#define BUF_SIZE 256

int sock;

void catch_sigint(int signum)
{
    close(sock);
    unlink(SOCK_NAME);
    exit(1);
}

int main(int argc, char ** argv)
{
    struct sockaddr srvr_name, rcvr_name;
    char buf[BUF_SIZE];
    int namelen, bytes;

    sock = socket(AF_UNIX, SOCK_DGRAM, 0);
    if (sock < 0)
    {
        perror("socket() failed");
        return EXIT_FAILURE;
    }

    signal(SIGINT, catch_sigint);
    srvr_name.sa_family = AF_UNIX;
    strcpy(srvr_name.sa_data, SOCK_NAME);

    if (bind(sock, &srvr_name, strlen(srvr_name.sa_data) +
        sizeof(srvr_name.sa_family)) < 0)
    {
        close(sock);
        unlink(SOCK_NAME);
        perror("bind() failed");
        return EXIT_FAILURE;
    }

    printf("waiting...\n");
```

```
49
50      while (1)
51      {
52          bytes = recvfrom(sock, buf, sizeof(buf), 0, &rcvr_name, &namelen);
53          if (bytes < 0)
54          {
55              close(sock);
56              unlink(SOCK_NAME);
57              perror("recvfrom() failed");
58              return EXIT_FAILURE;
59          }
60          buf[bytes] = 0;
61          rcvr_name.sa_data[namelen] = 0;
62          printf("Client sent: %s\n", buf);
63      }
64
65      close(sock);
66      unlink(SOCK_NAME);
67
68      return EXIT_SUCCESS;
69  }
```

С помощью вызова socket() создается сокет семейства адресов файловых сокетов Unix AF_UNIX типа SOCK_DGRAM(датаграммный сокет). Сокеты в файловом пространстве имен используют в качестве адресов имена файлов специального типа. После получения дескриптора сокета с помощью системного вызова bind() сокет связывается с заданным адресом. После вызова bind() сервер становится доступным для соединения. Для чтения данных из датаграммного сокета используется функция recvfrom(), на этой функции сервер блокируется до тех пор, пока на вход не поступят новые данные от клиентов. По завершении работы сокет закрывается с помощью функции close(). Перед выходом из программы-сервера файл сокета, созданный в результате вызова socket(), удаляется с помощью функции unlink().

Листинг 2: Код клиента fsclient.c

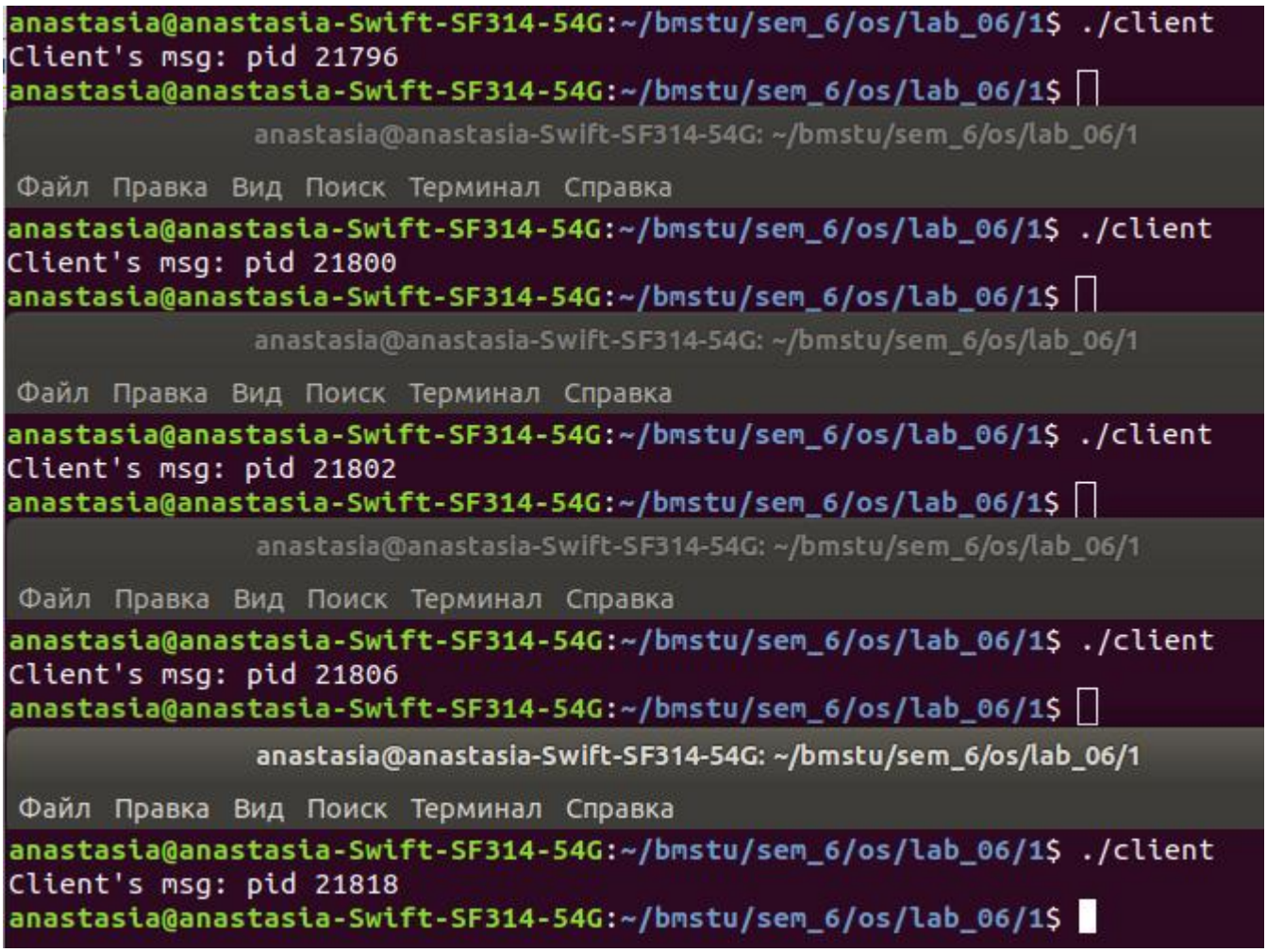
```
1  #include <stdlib.h>
2  #include <stdio.h>
3  #include <string.h>
4  #include <errno.h>
5  #include <unistd.h>
6  #include <sys/types.h>
7  #include <sys/socket.h>
8
9  #define SOCK_NAME "socket.soc"
10 #define BUF_SIZE 256
11
12 int main(int argc, char ** argv)
13 {
14     int sock;
15     char buf[BUF_SIZE];
```

```
16      struct sockaddr srvr_name;
17
18      sock = socket(AF_UNIX, SOCK_DGRAM, 0);
19
20      if (sock < 0)
21      {
22          perror("socket() failed");
23          return EXIT_FAILURE;
24      }
25
26      srvr_name.sa_family = AF_UNIX;
27      strcpy(srvr_name.sa_data, SOCK_NAME);
28      sprintf(buf, "pid %d", getpid());
29
30      printf("Client's msg: %s\n", buf);
31      sendto(sock, buf, strlen(buf), 0, &srvr_name,
32      strlen(srvr_name.sa_data) + sizeof(srvr_name.sa_family));
33
34      close(sock);
35      return EXIT_SUCCESS;
36  }
```

С помощью функции socket() открывается сокет семейства адресов файловых сокетов Unix AF_UNIX типа SOCK_DGRAM(датаграммный сокет). С помощью функции sendto() клиент передает данные серверу. После окончания передачи данных сокет закрывается при помощи close().

Демонстрация работы программного комплекса

```
anastasia@anastasia-Swift-SF314-54G:~/bmstu/sem_6/os/lab_06/1$ ./server
waiting...
Client sent: pid 21796
Client sent: pid 21800
Client sent: pid 21802
Client sent: pid 21806
Client sent: pid 21818
^Canastasia@anastasia-Swift-SF314-54G:~/bmstu/sem_6/os/lab_06/1$
```


anastasia@anastasia-Swift-SF314-54G:~/bmstu/sem_6/os/lab_06/1$ ./client
Client's msg: pid 21796
anastasia@anastasia-Swift-SF314-54G:~/bmstu/sem_6/os/lab_06/1$
anastasia@anastasia-Swift-SF314-54G: ~/bmstu/sem_6/os/lab_06/1
Файл Правка Вид Поиск Терминал Справка
anastasia@anastasia-Swift-SF314-54G:~/bmstu/sem_6/os/lab_06/1$ ./client
Client's msg: pid 21800
anastasia@anastasia-Swift-SF314-54G:~/bmstu/sem_6/os/lab_06/1$
anastasia@anastasia-Swift-SF314-54G: ~/bmstu/sem_6/os/lab_06/1
Файл Правка Вид Поиск Терминал Справка
anastasia@anastasia-Swift-SF314-54G:~/bmstu/sem_6/os/lab_06/1$ ./client
Client's msg: pid 21802
anastasia@anastasia-Swift-SF314-54G:~/bmstu/sem_6/os/lab_06/1$
anastasia@anastasia-Swift-SF314-54G: ~/bmstu/sem_6/os/lab_06/1
Файл Правка Вид Поиск Терминал Справка
anastasia@anastasia-Swift-SF314-54G:~/bmstu/sem_6/os/lab_06/1$ ./client
Client's msg: pid 21806
anastasia@anastasia-Swift-SF314-54G:~/bmstu/sem_6/os/lab_06/1$
anastasia@anastasia-Swift-SF314-54G: ~/bmstu/sem_6/os/lab_06/1
Файл Правка Вид Поиск Терминал Справка
anastasia@anastasia-Swift-SF314-54G:~/bmstu/sem_6/os/lab_06/1$ ./client
Client's msg: pid 21818
anastasia@anastasia-Swift-SF314-54G:~/bmstu/sem_6/os/lab_06/1$

Задание №2

Листинг 3: Код сервера netserver.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <strings.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>

#define BUF_SIZE 256
#define PORT 3425
#define NUMBER_OF_CLIENTS 5

void receive(int *clients, int n, fd_set *set)
{
    char buf[BUF_SIZE];
    int bytes;

    for (int i = 0; i < n; i++)
    {
        if (FD_ISSET(clients[i], set))
        {
            bytes = recv(clients[i], buf, BUF_SIZE, 0);

            if (bytes <= 0)
            {
                printf("Client[%d] disconnected\n", i);
                close(clients[i]);
                clients[i] = 0;
            }
            else
            {
                // send data back to client
                buf[bytes] = 0;
                printf("Client[%d] sent %s\n", i, buf);
                send(clients[i], buf, bytes, 0);
            }
        }
    }
}

int main(int argc, char ** argv)
{
     int sock;
     int new_sock;
     struct sockaddr_in serv_addr;
```

```
49      fd_set set;
50      int clients[NUMBER_OF_CLIENTS] = {0};
51      int mx;
52      int flag = 1;
53
54      sock = socket(AF_INET, SOCK_STREAM, 0);
55      if (socket < 0)
56      {
57        printf("socket() failed: %d\n", errno);
58        return EXIT_FAILURE;
59      }
60
61      fcntl(sock, F_SETFL, O_NONBLOCK);
62
63      serv_addr.sin_family = AF_INET;
64      serv_addr.sin_addr.s_addr = INADDR_ANY;
65      serv_addr.sin_port = htons(PORT);
66
67      if (bind(sock, (struct sockaddr *) &serv_addr, sizeof(serv_addr)) < 0)
68      {
69        printf("bind() failed: %d\n", errno);
70        return EXIT_FAILURE;
71      }
72
73      if (listen(sock, 6) < 0)
74      {
75          printf("listen() failed: %d\n", errno);
76          return EXIT_FAILURE;
77      }
78
79      printf("waiting...\n");
80
81      while(1)
82      {
83          FD_ZERO(&set);
84          FD_SET(sock, &set);
85          mx = sock;
86
87          for (int i = 0; i < NUMBER_OF_CLIENTS; i++)
88          {
89              if (clients[i])
90              {
91                  FD_SET(clients[i], &set);
92              }
93              mx = (mx > clients[i]) ? mx : clients[i];
94          }
95
96          if (select(mx + 1, &set, NULL, NULL, NULL) <= 0)
97          {
98              perror("select");
99              exit(1);
100         }
```

```
101
102         if (FD_ISSET(sock, &set))
103         {
104             new_sock = accept(sock, NULL, NULL);
105
106             if (new_sock < 0)
107             {
108                 perror("accept");
109                 exit(1);
110             }
111
112             fcntl(new_sock, F_SETFL, O_NONBLOCK);
113
114             flag = 1;
115             for (int i = 0; i < NUMBER_OF_CLIENTS && flag; i++)
116             {
117                 if (!clients[i])
118                 {
119                     clients[i] = new_sock;
120                     printf("Added as client №%d\n", i);
121                     flag = 0;
122                 }
123             }
124         }
125
126         receive(clients, NUMBER_OF_CLIENTS, &set);
127     }
128     return EXIT_SUCCESS;
129 }
```

С помощью вызова socket() создается сокет семейства AF_INET(что указывает, что сокет должен быть сетевым) типа SOCK_STREAM(потоковый сокет). Затем вызывается функция bind(), которая связывает сокет с адресом, указанным в SOCKET_ADDRESS. Функция listen() сообщает сокету, что должны приниматься новые соединения. Каждый раз, когда очередной клиент пытается соединиться с сервером, его запрос ставится в очередь, так как сервер может быть занят обработкой других запросов. С помощью макроса FD_ZERO() на каждой итерации цикла набор дескрипторов set очищается, затем с помощью макроса FD_SET() набор декрипторов set заполняется дескрипторами сокетов сервера и клиентов. Затем функция select() проверяет состояние нескольких дескрипторов сокетов сразу. Сама функция select() - блокирующая, она возвращает управление, если хотя бы один из проверяемых сокетов готов к выполнению соответствующей операции. Чтобы сделать сокет неблокирующим, используется функция fcntl() с флагом O_NONBLOCK, то есть вызов любой функции с таким сокетом будет возвращать управление немедленно. Затем проверяется наличие нового запроса на соединение. Вызывается функция accept(), которая устанавливает соединение в ответ на запрос клиента и создает копию сокета для того, чтобы исходный сокет мог продолжать прослушивание. Новый сокет объявляется как неблокирующий и

добавляется в массив дескрипторов сокетов клиентов. Затем осуществляется обход массива дескрипторов сокетов клиентов, если дескриптор сокета i-го клиента есть в наборе set, то с помощью функции recv() читаются данные i-го клиента, если не было прочитано положительное количество байт, то соединение разорвано, сокет удаляется из массива, иначе вывод сообщения клиента и отправка ответного сообщения клиенту.

Листинг 4: Код клиента netclient.c

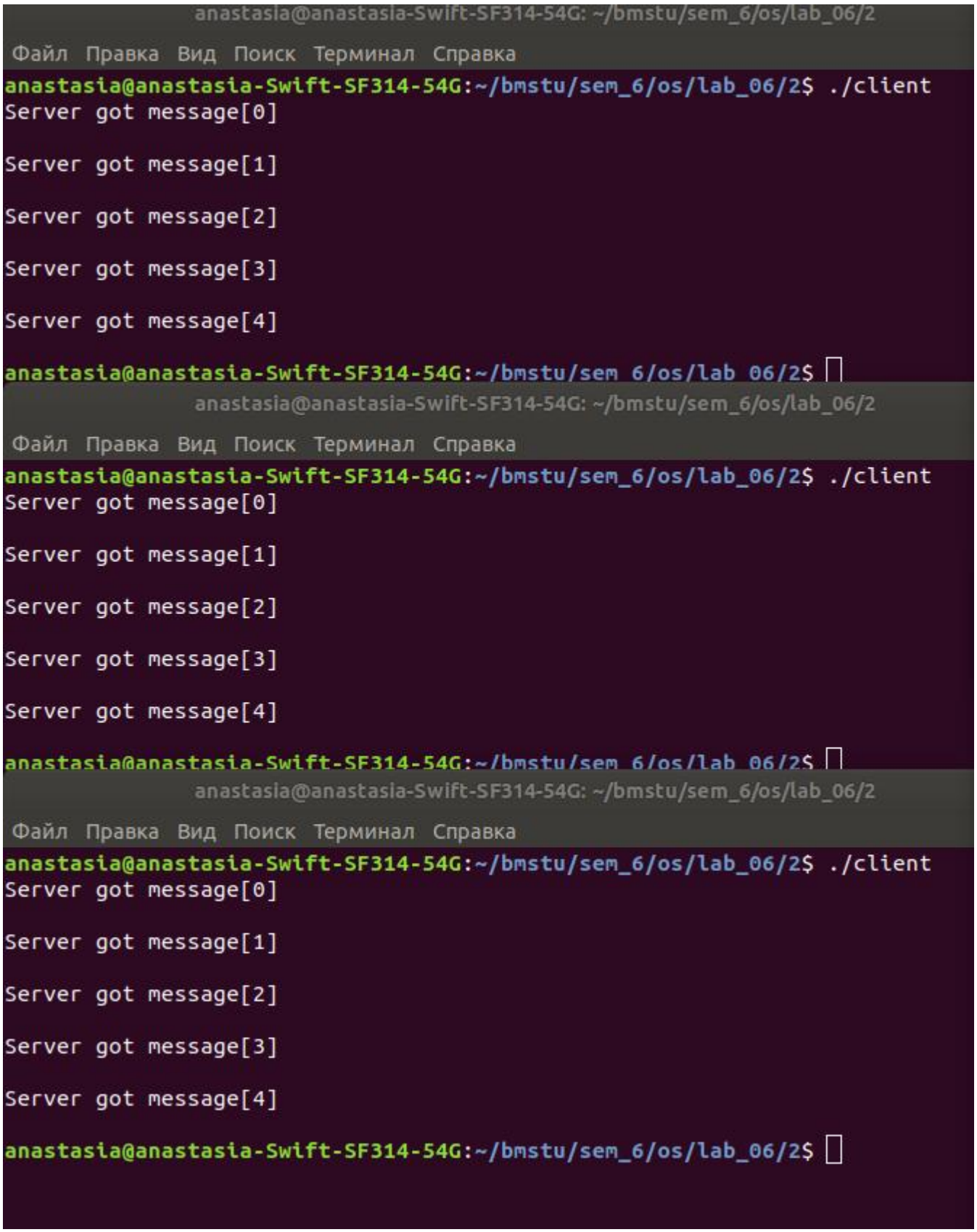
```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <errno.h>
4  #include <strings.h>
5  #include <sys/types.h>
6  #include <sys/socket.h>
7  #include <netinet/in.h>
8  #include <netdb.h>
9  #include <unistd.h>
10 #include <time.h>
11 #include <string.h>
12
13 #define PORT 3425
14 #define BUF_SIZE 256
15 #define COUNT 5
16 #define SOCK_ADDR "localhost"
17
18 int main(int argc, char ** argv)
19 {
20     int sock;
21     struct sockaddr_in serv_addr;
22     struct hostent *host;
23     char buf[BUF_SIZE];
24     char message[BUF_SIZE];
25
26     sock = socket(AF_INET, SOCK_STREAM, 0);
27     if (sock < 0)
28     {
29       printf("socket() failed: %d", errno);
30       return EXIT_FAILURE;
31     }
32
33     host = gethostbyname(SOCK_ADDR);
34     if (!host)
35     {
36         perror("gethostbyname() failed: ");
37         return EXIT_FAILURE;
38     }
39
40     serv_addr.sin_family = AF_INET;
41     serv_addr.sin_port = htons(PORT);
42     serv_addr.sin_addr = *((struct in_addr*) host->h_addr_list[0]);
43
```

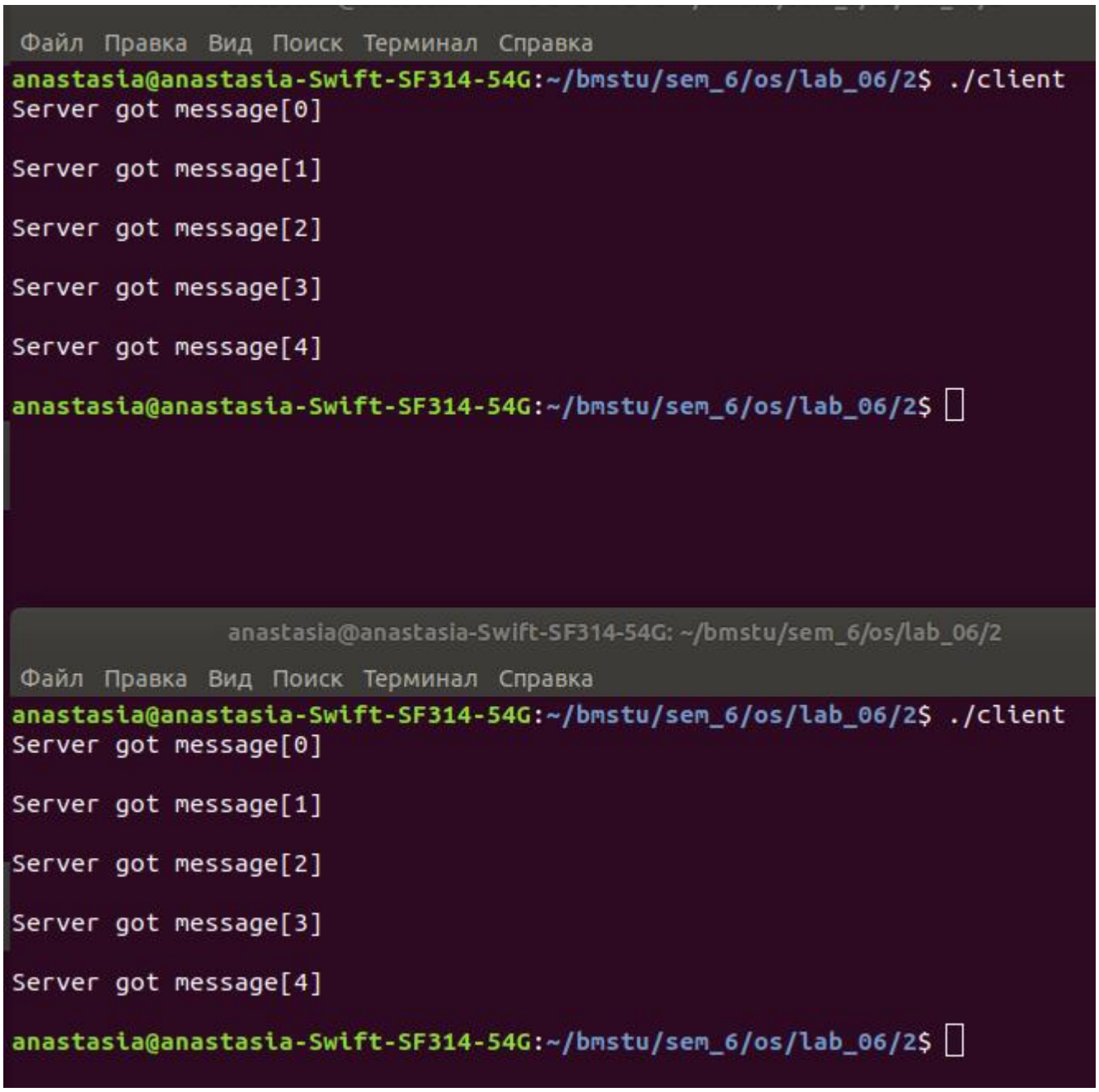
```
44      if (connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0)
45      {
46        printf("connect() failed: %d", errno);
47        return EXIT_FAILURE;
48      }
49
50      srand(time(NULL));
51
52      for (int i = 0; i < COUNT; i++)
53      {
54          memset(message, 0, BUF_SIZE);
55          sprintf(message, "message[%d]\n", i);
56
57          if (send(sock, message, sizeof(message), 0) < 0)
58          {
59              perror("send() failed:");
60              return EXIT_FAILURE;
61          }
62
63          recv(sock, buf, sizeof(message), 0);
64
65          printf("Server got %s\n", buf);
66
67          sleep(1 + rand() % 5);
68      }
69
70      close(sock);
71
72      return EXIT_SUCCESS;
73  }
```

С помощью функции socket() открывается сокет семейства AF_INET (то есть открываемый сокет должен быть сетевым) типа SOCK_STREAM(потоковый сокет). Затем для получения сетевого адреса по доменному имени используется функция gethostbyname(). Вызывается функция connect() для установки соединения. С помощью функции send() клиент передает серверу данные, затем клиент блокируется на функции recv() до тех пор, пока на вход не поступит ответное сообщение от сервера. После выхода из блокировки выводится полученный ответ от сервера. После окончания передачи данных сокет закрывается при помощи close().

Демонстрация работы программного комплекса

```
anastasia@anastasia-Swift-SF314-54G:~/bmstu/sem_6/os/lab_06/2$ ./server
waiting...
Added as client №0
Client[0] sent message[0]

Added as client №1
Client[1] sent message[0]

Added as client №2
Client[2] sent message[0]

Client[2] sent message[1]

Added as client №3
Client[3] sent message[0]

Client[1] sent message[1]

Client[0] sent message[1]

Client[2] sent message[2]

Added as client №4
Client[4] sent message[0]

Client[3] sent message[1]

Client[1] sent message[2]

Client[2] sent message[3]

Client[0] sent message[2]

Client[4] sent message[1]

Client[2] sent message[4]

Client[1] sent message[3]

Client[0] sent message[3]

Client[2] disconnected
Client[3] sent message[2]

Client[4] sent message[2]

Client[0] sent message[4]

Client[3] sent message[3]

Client[1] sent message[4]

Client[3] sent message[4]

Client[3] disconnected
Client[4] sent message[3]

Client[1] disconnected
Client[0] disconnected
Client[4] sent message[4]

Client[4] disconnected
^C
```


anastasia@anastasia-Swift-SF314-54G: ~/bmstu/sem_6/os/lab_06/2
Файл Правка Вид Поиск Терминал Справка
anastasia@anastasia-Swift-SF314-54G:~/bmstu/sem_6/os/lab_06/2$ ./client
Server got message[0]
Server got message[1]
Server got message[2]
Server got message[3]
Server got message[4]
anastasia@anastasia-Swift-SF314-54G:~/bmstu/sem_6/os/lab_06/2$
anastasia@anastasia-Swift-SF314-54G: ~/bmstu/sem_6/os/lab_06/2
Файл Правка Вид Поиск Терминал Справка
anastasia@anastasia-Swift-SF314-54G:~/bmstu/sem_6/os/lab_06/2$ ./client
Server got message[0]
Server got message[1]
Server got message[2]
Server got message[3]
Server got message[4]
anastasia@anastasia-Swift-SF314-54G:~/bmstu/sem_6/os/lab_06/2$
anastasia@anastasia-Swift-SF314-54G: ~/bmstu/sem_6/os/lab_06/2
Файл Правка Вид Поиск Терминал Справка
anastasia@anastasia-Swift-SF314-54G:~/bmstu/sem_6/os/lab_06/2$ ./client
Server got message[0]
Server got message[1]
Server got message[2]
Server got message[3]
Server got message[4]
anastasia@anastasia-Swift-SF314-54G:~/bmstu/sem_6/os/lab_06/2$


Файл Правка Вид Поиск Терминал Справка
anastasia@anastasia-Swift-SF314-54G:~/bmstu/sem_6/os/lab_06/2$ ./client
Server got message[0]
Server got message[1]
Server got message[2]
Server got message[3]
Server got message[4]
anastasia@anastasia-Swift-SF314-54G:~/bmstu/sem_6/os/lab_06/2$
anastasia@anastasia-Swift-SF314-54G: ~/bmstu/sem_6/os/lab_06/2
Файл Правка Вид Поиск Терминал Справка
anastasia@anastasia-Swift-SF314-54G:~/bmstu/sem_6/os/lab_06/2$ ./client
Server got message[0]
Server got message[1]
Server got message[2]
Server got message[3]
Server got message[4]
anastasia@anastasia-Swift-SF314-54G:~/bmstu/sem_6/os/lab_06/2$