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»

Дисциплина: «Операционные системы»

Лабораторная работа №6

Тема работы: «Сокеты»

Студент: Левушкин И. К.
Группа: ИУ7-62Б
Преподаватель: Рязанова Н. Ю.

Москва, 2020 г.

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.

Ниже приведен программный комплекс, реализующий поставленную задачу.

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

#include "info.h"

int sockfd;

void sigint_catcher(int signum)
{
    printf("Closing socket 'cause of Ctrl+C....\n");
    close(sockfd);
    unlink(SOCKET_NAME);

}

int main(void)
{
    char msg[MSG_LEN];
    struct sockaddr client_addr;

    if ((sockfd = socket(PF_LOCAL, SOCK_DGRAM, 0)) < 0)
    {
        perror("Error in socket(): ");
        return sockfd;
    }

    client_addr.sa_family = PF_LOCAL;
    strcpy(client_addr.sa_data, SOCKET_NAME);

    if (bind(sockfd, &client_addr, sizeof(client_addr)) < 0)
    {
        printf("Closing socket...\n");
        close(sockfd);
        unlink(SOCKET_NAME);
        perror("Error in bind(): ");
        return -1;
```

```
42      }
43
44      printf("\nServer is waiting for the message...\n");
45      signal(SIGINT, sigint_catcher);
46
47      int recievedSize;
48
49      while(1)
50      {
51          if ((recievedSize = recv(sockfd, msg, sizeof(msg), 0)) < 0)
52          {
53              close(sockfd);
54              unlink(SOCKET_NAME);
55              perror("Error in recv(): ");
56              return recievedSize;
57          }
58
59          msg[recievedSize] = 0;
60          printf("Client send this message: %s\n", msg);
61      }
62
63      printf("Closing socket...\n");
64      close(sockfd);
65      unlink(SOCKET_NAME);
66      return 0;
67  }
68
```

Рис. 1: server.c

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   #include <string.h>
4   #include <unistd.h>
5   #include <sys/types.h>
6   #include <sys/socket.h>
7
8   #include "info.h"
9
10  int main(void)
11  {
12      int sockfd = socket(PF_LOCAL, SOCK_DGRAM, 0); //socket descriptor
13      if (sockfd < 0)
14      {
15          printf("Error in socket();\n");
16          return sockfd;
17      }
18
19      struct sockaddr server_addr;
20      server_addr.sa_family = PF_LOCAL;
21      strcpy(server_addr.sa_data, SOCKET_NAME);
22
23      char msg[MSG_LEN];
24
25      sprintf(msg, "Hello, I am a client, pid = %d\n", getpid());
26      sendto(sockfd, msg, strlen(msg), 0, &server_addr, sizeof(server_addr));
27
28      close(sockfd);
29      return 0;
30  }
31
```

Рис. 2: client.c

```c
#ifndef INFO_H
#define INFO_H

#define MSG_LEN 256
#define SOCKET_NAME "socket.soc"

#endif // INFO_H
```

Рис. 3: info.h

Демонстрация работы программы

Ниже представлены результаты работы программного комплекса с 5-ю запущенными клиентами. Выход из программы осуществляется сигналом SIGINT.

```
ilalevuskin@ubuntu:~/Desktop/lab6/unix$ ./server

Server is waiting for the message...
Client send this message: Hello, I am a client, pid = 2495

Client send this message: Hello, I am a client, pid = 2508

Client send this message: Hello, I am a client, pid = 2517

Client send this message: Hello, I am a client, pid = 2520

Client send this message: Hello, I am a client, pid = 2543

^CClosing socket 'cause of Ctrl+C....
```

```
ilalevuskin@ubuntu:~/Desktop/lab6/unix$ ./client
ilalevuskin@ubuntu:~/Desktop/lab6/unix$ ./client
ilalevuskin@ubuntu:~/Desktop/lab6/unix$ ./client
ilalevuskin@ubuntu:~/Desktop/lab6/unix$ ./client
ilalevuskin@ubuntu:~/Desktop/lab6/unix$ ./client
```

Рис. 4: Демонстрация работы программного комплекса

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.

Ниже приведен программный комплекс, реализующий поставленную задачу.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/select.h>
#include <arpa/inet.h>
#include <netdb.h>

#include "info.h"

#define MAX_CLIENTS 10
int clients[MAX_CLIENTS] = { 0 };


void manageConnection(unsigned int fd)
{
    struct sockaddr_in client_addr;
    int addrSize = sizeof(client_addr);

    int incom = accept(fd, (struct sockaddr*) &client_addr, (socklen_t*) &addrSize);
    if (incom < 0)
    {
        perror("Error in accept(): ");
        exit(-1);
    }

    printf("\nNew connection: \nfd = %d \nip = %s:%d\n", incom,
                                inet_ntoa(client_addr.sin_addr), ntohs(client_addr.sin_port));

    for (int i = 0; i < MAX_CLIENTS; i++)
    {
        if (clients[i] == 0)
        {
            clients[i] = incom;
            printf("Managed as client #%d\n", i);
            break;
        }
    }

}
```

Рис. 5: server.c

```
44
45  void manageClient(unsigned int fd, unsigned int client_id)
46  {
47      char msg[MSG_LEN];
48      memset(msg, 0, MSG_LEN);
49
50      struct sockaddr_in client_addr;
51      int addrSize = sizeof(client_addr);
52
53      int recvSize = recv(fd, msg, MSG_LEN, 0);
54      if (recvSize == 0)
55      {
56          getpeername(fd, (struct sockaddr*) &client_addr, (socklen_t*) &addrSize);
57          printf("User %d disconnected %s:%d \n", client_id, inet_ntoa(client_addr.sin_addr),
58              ntohs(client_addr.sin_port));
59          close(fd);
60          clients[client_id] = 0;
61      }
62      else
63      {
64          msg[recvSize] = '\0';
65          printf("Message from %d client: %s\n", client_id, msg);
66      }
67  }
68
69
70  int main(void)
71  {
72      int listener = socket(PF_INET, SOCK_STREAM, 0);
73      if (listener < 0)
74      {
75          perror("Error in sock(): ");
76          return listener;
77      }
78
79      struct sockaddr_in client_addr;
80      client_addr.sin_family = PF_INET;
81      client_addr.sin_port = htons(SOCK_PORT);
82      client_addr.sin_addr.s_addr = INADDR_ANY;
83
84      if (bind(listener, (struct sockaddr*) &client_addr, sizeof(client_addr)) < 0)
85      {
86          perror("Error in bind():");
```

Рис. 6: server.c

```
87          return -1;
88      }
89      printf("Server is listening on the %d port...\n", SOCK_PORT);
90
91      if (listen(listener, 3) < 0)
92      {
93          perror("Error in listen(): ");
94          return -1;
95      }
96      printf("Waiting for the connections...\n");
97
98      fd_set readfds;
99      int max_fd;
100     int active_clients_count;
101
102     while(1)
103     {
104         FD_ZERO(&readfds);
105         FD_SET(listener, &readfds);
106         max_fd = listener;
107
108         for (int i = 0; i < MAX_CLIENTS; i++)
109         {
110             int fd = clients[i];
111
112             if (fd > 0)
113             {
114                 FD_SET(fd, &readfds);
115             }
116
117             max_fd = (fd > max_fd) ? (fd) : (max_fd);
118         }
119
120         active_clients_count = select(max_fd + 1, &readfds, NULL, NULL, NULL);
121
122         if (active_clients_count < 0 && (errno != EINTR))
123         {
124             perror("Error in select():");
125             return active_clients_count;
126         }
127
128         if (FD_ISSET(listener, &readfds))
129         {
```

Рис. 7: server.c

```
130             manageConnection(listener);
131         }
132
133         for (int i = 0; i < MAX_CLIENTS; i++)
134         {
135             int fd = clients[i];
136             if ((fd > 0) && FD_ISSET(fd, &readfds))
137             {
138                 manageClient(fd, i);
139             }
140         }
141     }
142
143     return 0;
144 }
145
```

Рис. 8: server.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <unistd.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netdb.h>

#include "info.h"

int main(void)
{
    srand(time(NULL));

    int sock = socket(PF_INET, SOCK_STREAM, 0);
    if (sock < 0)
    {
        perror("Error in sock(): ");
        return sock;
    }

    struct hostent* host = gethostbyname(SOCK_ADDR);
    if (!host)
    {
        perror("Error in gethostbyname(): ");
        return -1;
    }

    struct sockaddr_in server_addr;
    server_addr.sin_family = PF_INET;
    server_addr.sin_port = htons(SOCK_PORT);
    server_addr.sin_addr = *((struct in_addr*) host->h_addr_list[0]);

    if (connect(sock, (struct sockaddr*) &server_addr, sizeof(server_addr)) < 0)
    {
        perror("Error in connect():");
        return -1;
    }

    char msg[MSG_LEN];
    for (int i = 0; i < 10; i++)
    {
        memset(msg, 0, MSG_LEN);
        sprintf(msg, "%d message was sended, pid = %d\n", i, getpid());
        printf("%s", msg);

        if (send(sock, msg, strlen(msg), 0) < 0)
        {
            perror("Error in send(): ");
            return -1;
        }

        printf("Sended %d message\n", i);

        int wait_time = 1 + rand() % 3;
        sleep(wait_time);
    }

    printf("Client app is over!\n");
    return 0;
}
```

Рис. 9: client.c

Демонстрация работы программы

Ниже представлены результаты работы программного комплекса с 5-ю параллельно запущенными клиентами.

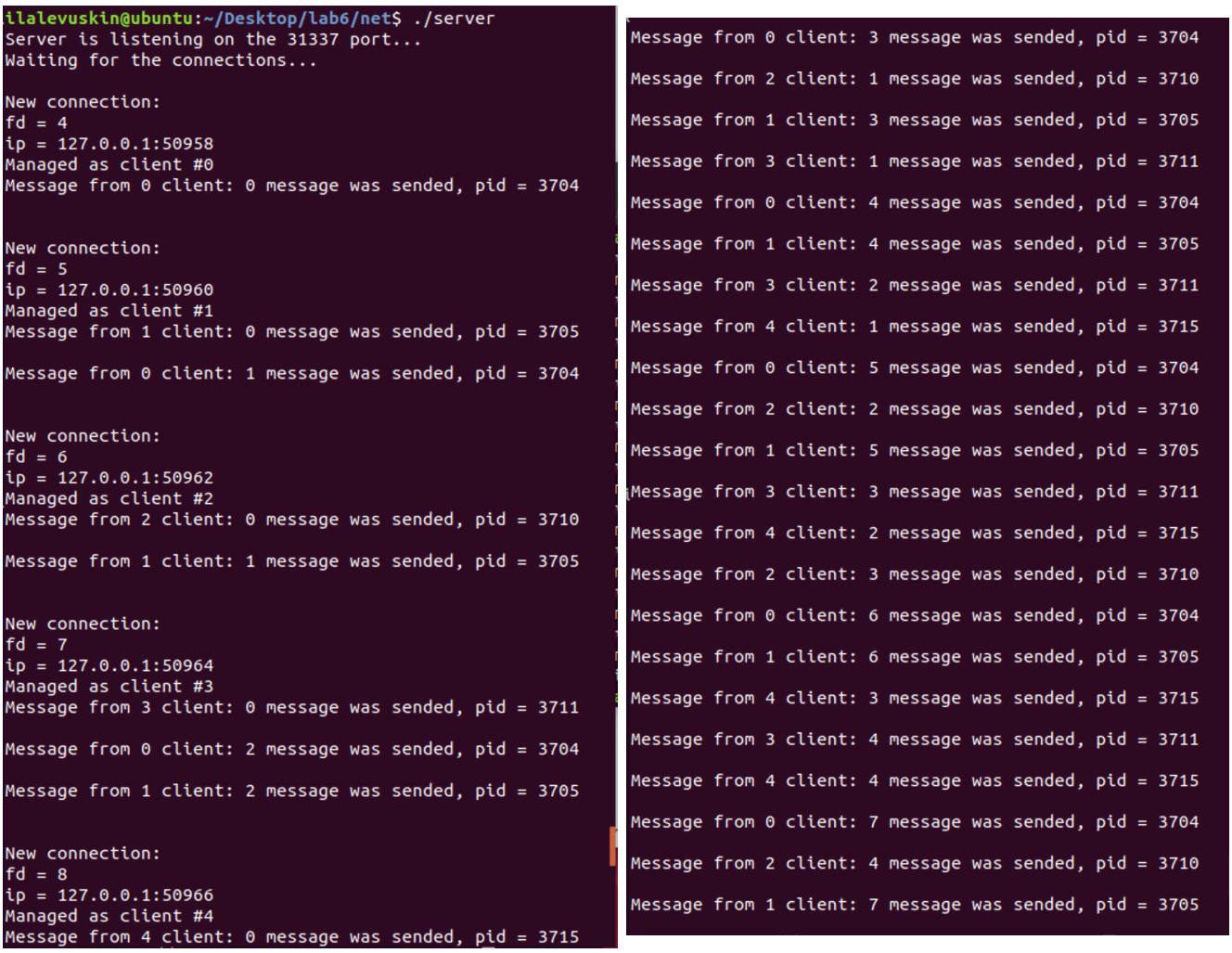

Рис. 10: Результат работы сервера

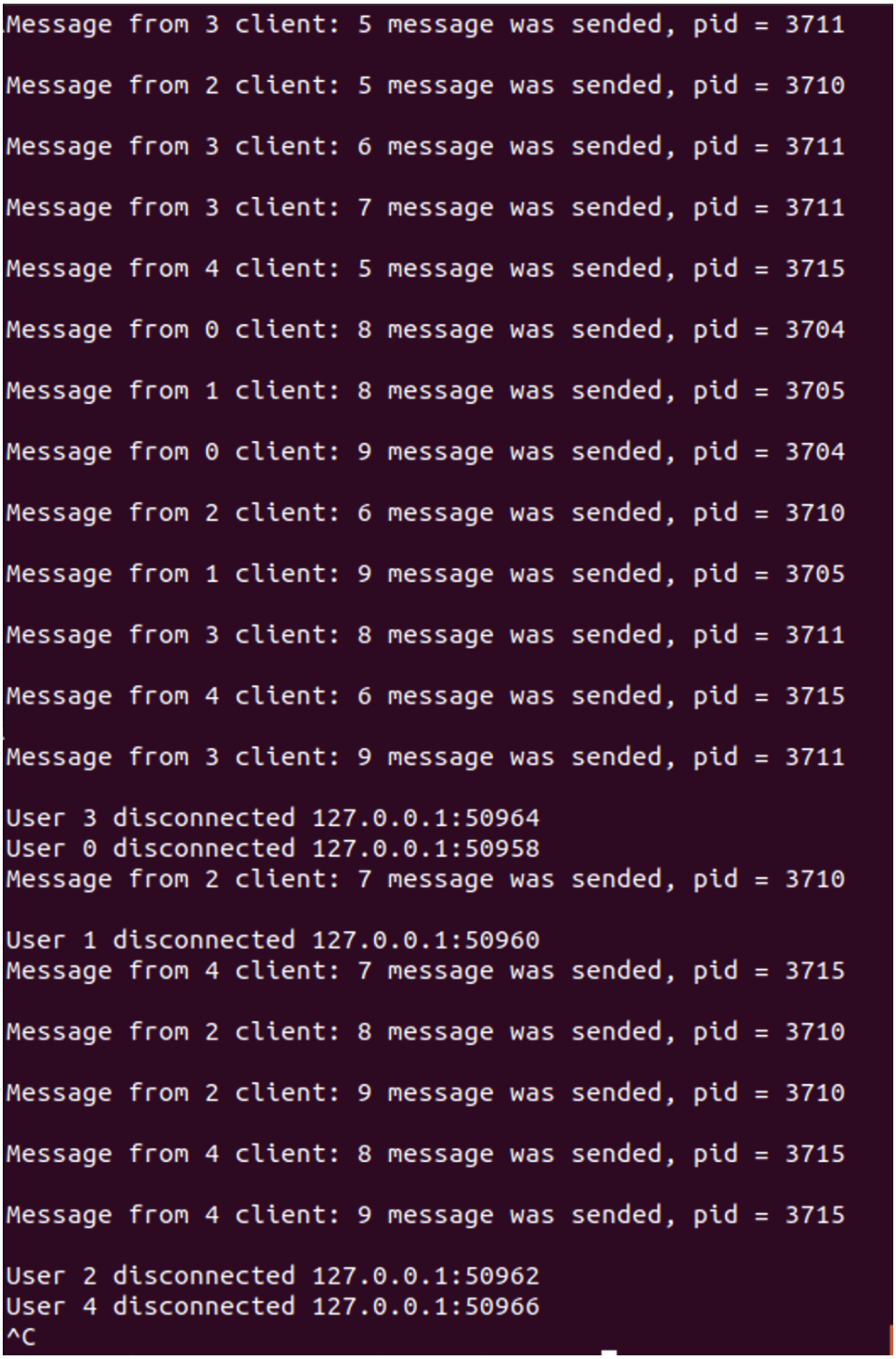

Рис. 11: Результат работы сервера

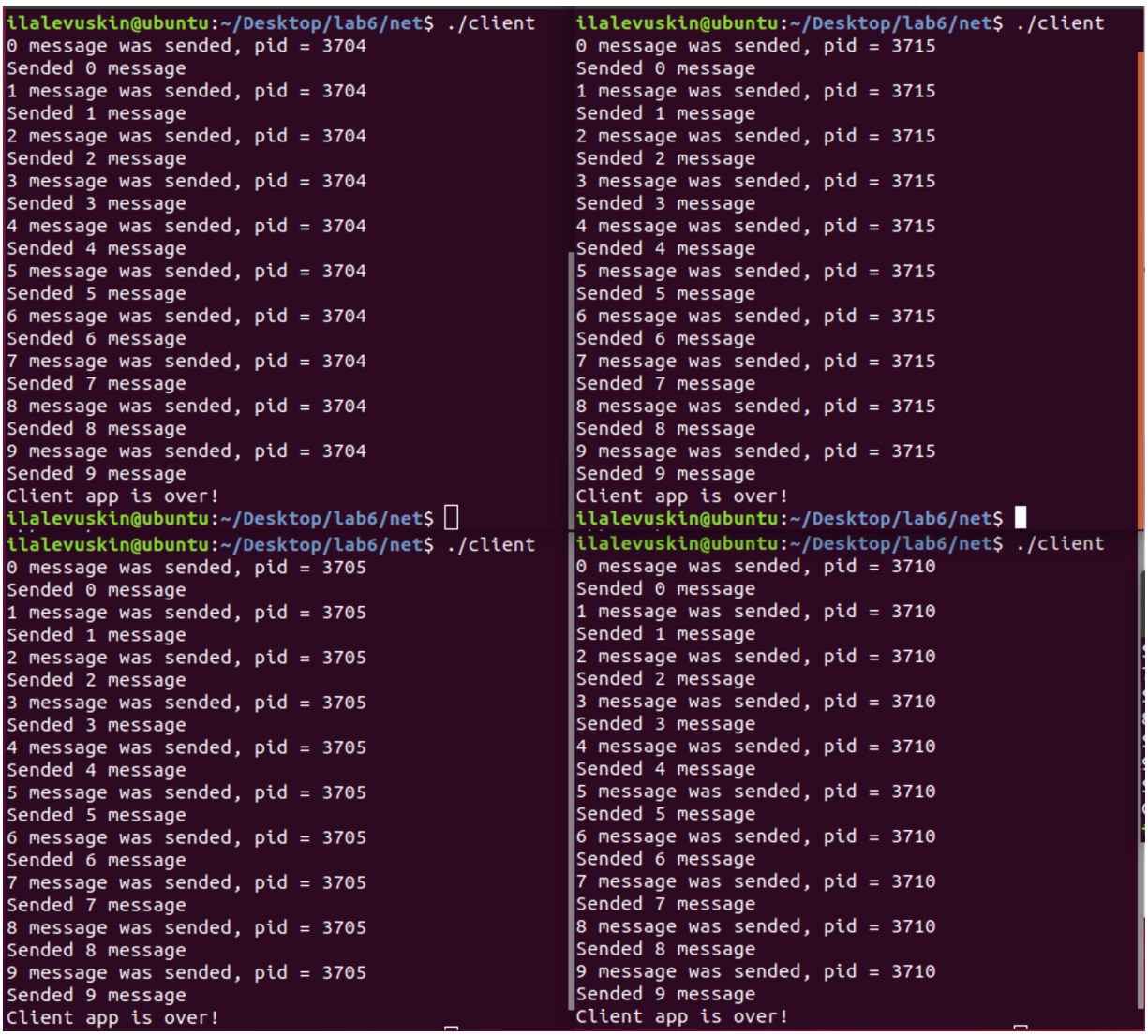

Рис. 12: Демонстрация запуска четырех клиентов

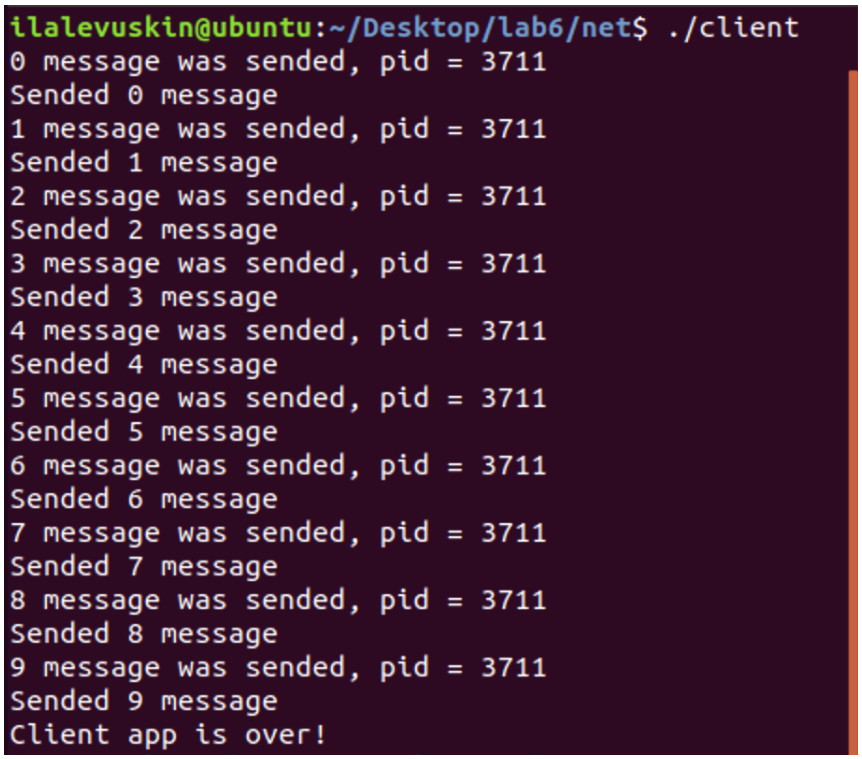

Рис. 13: Демонстрация запуска 5-ого клиента