



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

ФАКУЛЬТЕТ «Робототехника и комплексная автоматизация»

КАФЕДРА «Системы автоматизированного проектирования (РК-6)»

ОТЧЕТ О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ

по дисциплине «Разработка программных систем»

Студент:	Кильдишев Петр Степанович
Группа:	РК6-66Б
Тип задания:	Лабораторная работа №3
Тема:	Сетевое программирование
Вариант:	1

Студент

подпись, дата

Кильдишев П.С.
Фамилия, И.О.

Преподаватель

подпись, дата

Козов А.В.
Фамилия, И.О.

Москва, 2023

Содержание

Задание	3
Описание структуры программы и реализованных способов взаимодействия клиента с сервером	4
Описание основных используемых структур данных	5
Блок-схемы	6
Примеры работы программы	8
Текст программы	11

Задание

Разработать клиент-серверное приложение "Телеграф" для общения двух абонентов на узлах сети Internet в полудуплексном режиме в текстовом формате. Абонент-сервер запускает серверную часть приложения в режиме ожидания запроса от клиента. Абонент-клиент, зная адрес абонента-сервера, инициирует связь с абонентом-сервером и передает ему сообщение (в общем случае, многострочное). Завершая сообщение, абонент-клиент передает абоненту-серверу код "Перехожу на прием". Получив этот код, абонент-сервер передает абоненту-клиенту свое сообщение, завершая его кодом "Перехожу на прием". Далее наступает очередь абонента-клиента передавать сообщение. Тексты сообщений с обеих сторон вводятся пользователями-людьми с устройства стандартного ввода. Тщательно проработать протокол взаимодействия, предусмотрев механизм прекращения диалога.

Описание структуры программы и реализованных способов взаимодействия клиента с сервером

Для выполнения поставленной задачи реализовано 2 программы: программы-клиент и программа-сервер.

Программа-сервер при запуске переходит в режим ожидания подключения к ней клиента. Подключение клиентом может быть выполнено по IP-адресу, введенному в параметрах запуска программы-клиента. При обратной последовательности запуска или не верно введенном IP-адресе сервера пользователь-клиент получит сообщение об ошибке.

При успешном подключении клиента к серверу, сервер переходит на режим приема и блокируется до получения ввода клиента, сообщая об этом пользователю сервера. Пользователь-клиент при этом может ввести текстовое сообщение, которое передастся на экран пользователю сервера. При таком действии клиентом его программа перейдет на режим приема и заблокируется до получения сообщения сервера. Сервер после этого получит сообщение и выведет его на экран пользователя сервером. После этого пользователь сервером может ввести свое сообщение, которое передастся клиенту аналогичным образом.

Обмен сообщениями будет происходить в бесконечном цикле до тех пор, пока один из пользователей не введет пустое сообщение комбинацией клавишь Ctrl+D или не завершит программу комбинацией клавишь Ctrl+C.

UML-диаграмма временной последовательности взаимодействия клиента и сервера представлена на рисунке 1.

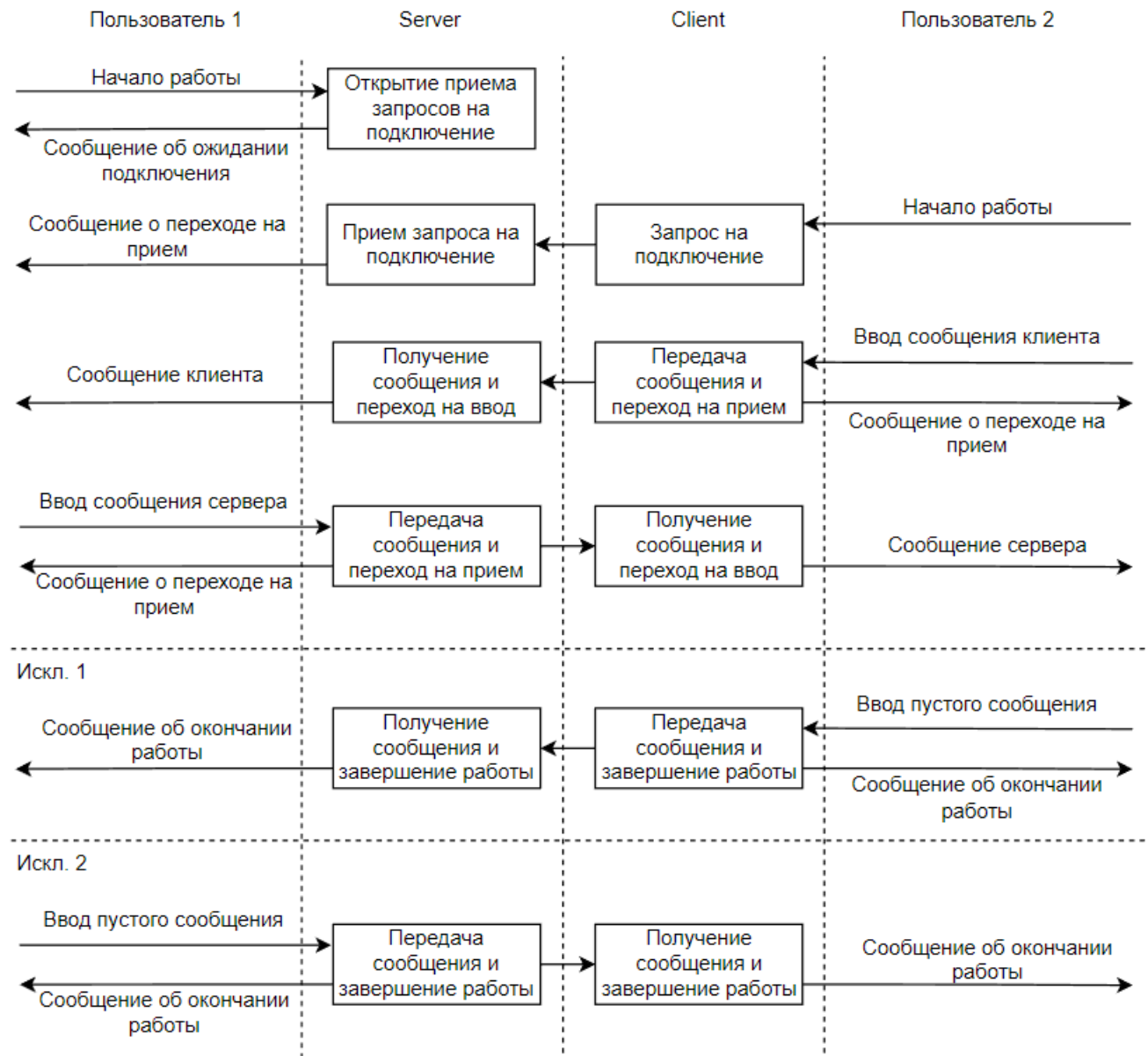


Рис. 1. UML-диаграмма временной последовательности взаимодействия клиента и сервера

Описание основных используемых структур данных

Первый аргумент запуска программы-клиента `argv[1]` должен содержать IP-адрес сервера для корректной работы программы.

Блок-схемы

На рисунке 2 представлена блок-схема программы-сервера.

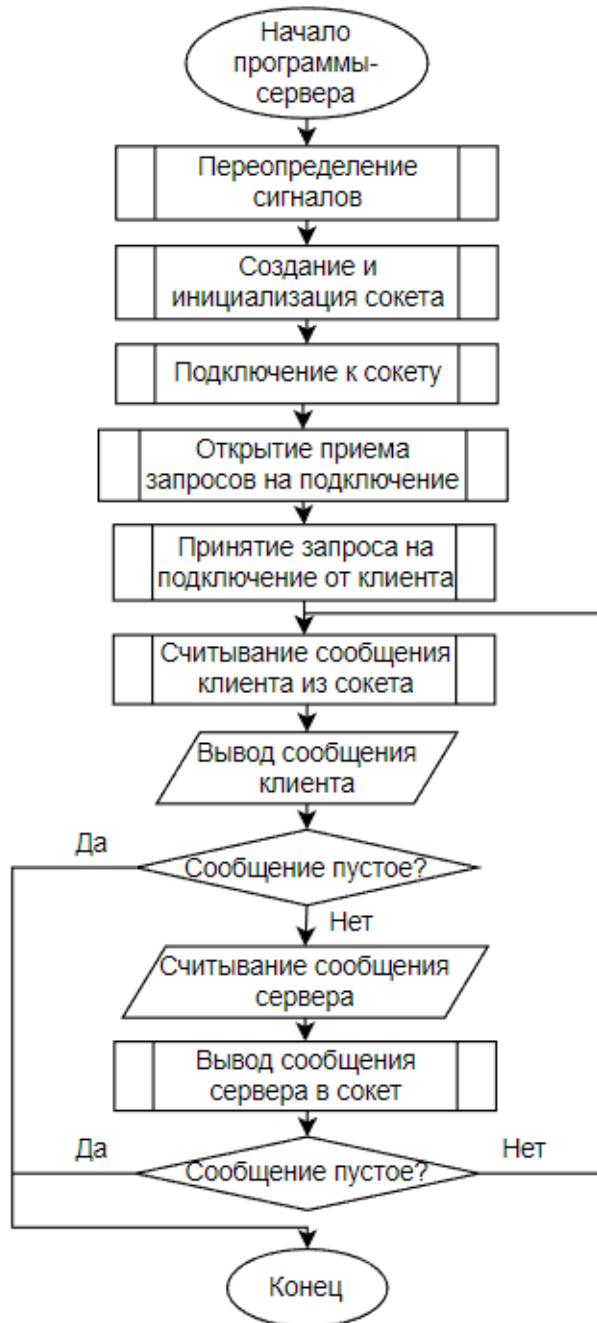


Рис. 2. Блок-схема программы-сервера

На рисунке 3 представлена блок-схема программы-клиента.

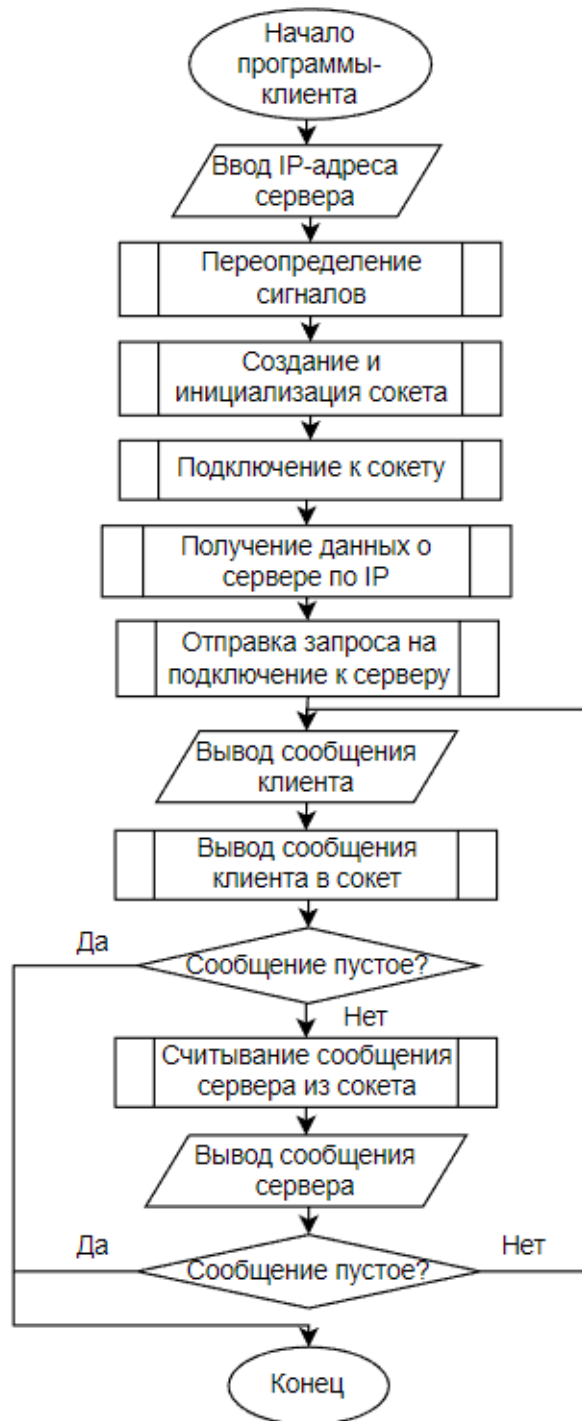


Рис. 3. Блок-схема программы-клиента

Примеры работы программы

На рисунке 4 приведен пример успешного запуска программы-сервера с переходом в ожидание подключения:

```
petr_kildishev@Balls:/mnt/d$ ./server
Ожидание подключения.
```

Рис. 4. Пример работы программы №1

На рисунке 5 приведен пример не успешного запуска программы-клиента по причине отсутствия в параметрах запуска IP-адреса сервера:

```
petr_kildishev@Balls:/mnt/d$ ./client
В параметрах запуска введите IP-адрес сервера.
```

Рис. 5. Пример работы программы №2

На рисунке 6 приведен пример успешного запуска программы-клиента с подключением к серверу и отправкой сообщения с переходом в режим приема:

```
petr_kildishev@Balls:/mnt/d$ ./client 127.0.0.1
aasdad
Перехожу на прием.
```

Рис. 6. Пример работы программы №3

На рисунке 7 приведен пример последующего получения сервером сообщения клиента и перехода в режим ввода сообщения:

```
petr_kildishev@Balls:/mnt/d$ ./server
Ожидание подключения.
Перехожу на прием.
aasdad
```

Рис. 7. Пример работы программы №4

На рисунке 8 приведен пример последующего ввода сообщения в программе-сервере и переходе её на режим приема:

```
petr_kildishev@Balls:/mnt/d$ ./server
Ожидание подключения.
Перехожу на прием.
aasdad
hello
Перехожу на прием.
```

Рис. 8. Пример работы программы №5

На рисунке 9 приведен пример последующего получения клиентом сообщения от сервера и перехода клиента на режим ввода:

```
petr_kildishev@Balls:/mnt/c$ ./client 127.0.0.1
aasdad
Перехожу на прием.
hello
```

Рис. 9. Пример работы программы №6

На рисунках 10 и 11 приведен результат ввода клиентом пустого сообщения, что приводит к завершению работы программы-сервера и программы-клиента:

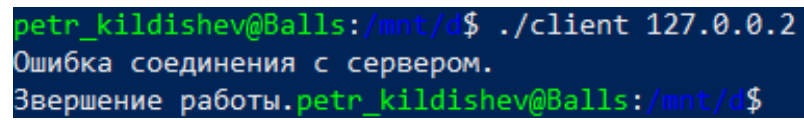
```
petr_kildishev@Balls:/mnt/d$ ./client 127.0.0.1
aasdad
Перехожу на прием.
hello
Звершение работы.petr_kildishev@Balls:/mnt/d$
```

Рис. 10. Пример работы программы №7

```
petr_kildishev@Balls:/mnt/d$ ./server
Ожидание подключения.
Перехожу на прием.
aasdad
hello
Перехожу на прием.
Звершение работы.
petr_kildishev@Balls:/mnt/d$
```

Рис. 11. Пример работы программы №8

На рисунке 12 приведен пример неудачного запуска программы-клиента, не сумевшей подключиться к серверу по заданному IP-адресу:

A screenshot of a terminal window with a dark blue background and light green text. The prompt is 'petr_kildishev@Balls:/mnt/d\$'. The user enters './client 127.0.0.2'. The output shows 'Ошибка соединения с сервером.' followed by 'Звершение работы.' and the prompt again.

```
petr_kildishev@Balls:/mnt/d$ ./client 127.0.0.2
Ошибка соединения с сервером.
Звершение работы.petr_kildishev@Balls:/mnt/d$
```

Рис. 12. Пример работы программы №9

Текст программы

Ниже в листинге 1 представлен текст программы-сервера.

Листинг 1. Листинг программы-сервера

```
1 #include <sys/types.h>
2 #include <sys/socket.h>
3 #include <netinet/in.h>
4 #include <netdb.h>
5 #include <memory.h>
6 #include <stdio.h>
7 #include <string.h>
8 #include <arpa/inet.h>
9 #include <unistd.h>
10 #include <signal.h>
11 #include <stdlib.h>
12
13 #define SRV_PORT 1234
14 #define BUF_SIZE 128
15 #define GET "Перехожу наприем.\n"
16 #define CONNECTION_WAIT "Ожидание подключения.\n"
17
18 int s, s_new;
19
20 void finish(int a = NULL) // Функция завершения работы
21 {
22     printf("Звершение работы.\n");
23     close(s);
24     close(s_new);
25     exit(0);
26 }
27
28 int main()
29 {
30     socklen_t from_len;
31     char buf[BUF_SIZE];
32     struct sockaddr_in sin, from_sin;
33
34     signal(SIGINT, finish); // Переопределение сигналов
35     signal(SIGKILL, finish);
36     signal(SIGTERM, finish);
37
38     if ((s = socket(AF_INET, SOCK_STREAM, 0)) < 0) // Создание сокета
39     {
40         printf("Ошибка создания сокета.\n");
```

```

41     exit(-1);
42 }
43 memset((char*)&sin, 0, sizeof(sin)); // Выделение памяти для сокета и его
    инициализация
44 sin.sin_family = AF_INET;
45 sin.sin_addr.s_addr = INADDR_ANY;
46 sin.sin_port = SRV_PORT;
47
48 if ((bind(s, (struct sockaddr*)&sin, sizeof(sin))) < 0) // Подключение к сокету
49 {
50     printf("Ошибка соединения адресом.\n");
51     exit(-1);
52 }
53
54 write(1, CONNECTION_WAIT, strlen(CONNECTION_WAIT));
55
56 if (listen(s, 1) < 0) // Ожидание соединения
57 {
58     printf("Ошибка перехода в режим приема.\n");
59     close(s);
60     exit(-1);
61 }
62
63 from_len = sizeof(from_sin);
64 if ((s_new = accept(s, (struct sockaddr*)&from_sin, &from_len)) < 0) // Принятие
    запроса на соединение
65 {
66     printf("Ошибка соединения клиентом.\n");
67     finish();
68 }
69 while (1)
70 {
71     write(1, GET, strlen(GET));
72     from_len = read(s_new, buf, BUF_SIZE); // Чтение сообщения из сокета
73     write(1, buf, from_len); // Вывод сообщения
74     if (from_len == 0) // Если пришло сообщение нулевой длины - конец
75         finish();
76
77     from_len = read(0, buf, BUF_SIZE); // Считывание нового сообщения
78     write(s_new, buf, from_len); // Запись сообщения в сокет
79     if (from_len == 0) // Если сообщение имело нулевую длину - конец
80         finish();
81 }
82 close(s_new); // Очистка памяти сокета
83 }

```

Ниже в листинге 2 представлен текст программы-клиента.

Листинг 2. Листинг программы-клиента

```
1 #include <sys/types.h>
2 #include <sys/socket.h>
3 #include <netinet/in.h>
4 #include <netdb.h>
5 #include <memory.h>
6 #include <stdio.h>
7 #include <string.h>
8 #include <arpa/inet.h>
9 #include <unistd.h>
10 #include <signal.h>
11 #include <stdlib.h>
12
13 #define SRV_PORT 1234
14 #define CLNT_PORT 1235
15 #define BUF_SIZE 128
16 #define GET "Перехожу наприем.\n"
17
18 int s;
19
20 void finish(int a = NULL) // Функция завершения работы
21 {
22     printf("Звершение работы.");
23     close(s);
24     exit(0);
25 }
26
27 int main(int argc, char* argv[])
28 {
29     int from_len;
30     char buf[BUF_SIZE];
31     struct hostent* hp;
32     struct sockaddr_in clnt_sin, srv_sin;
33
34     signal(SIGINT, finish); // Переопределение сигналов
35     signal(SIGKILL, finish);
36     signal(SIGTERM, finish);
37
38     if (argc < 2)
39     {
40         printf("В параметрах запуска введите IP-адресс — сервера.\n");
41         exit(-1);
42     }
43 }
```

```

44  if ((s = socket(AF_INET, SOCK_STREAM, 0)) < 0) // Создание сокета
45  {
46      printf("Ошибка создания сокета.\n");
47      exit(-1);
48  }
49  memset((char*)&clnt_sin, 0, sizeof(clnt_sin)); // Выделение памяти для сокета и его
        инициализация
50  clnt_sin.sin_family = AF_INET;
51  clnt_sin.sin_addr.s_addr = INADDR_ANY;
52  clnt_sin.sin_port = CLNT_PORT;
53  if ((bind(s, (struct sockaddr*)&clnt_sin, sizeof(clnt_sin))) < 0) // Подключение к
        сокету
54  {
55      printf("Ошибка соединения адресом.\n");
56      finish();
57  }
58
59  memset((char*)&srv_sin, 0, sizeof(srv_sin));
60  hp = gethostbyname(argv[1]); // Получение информации о сервере по ip
61  srv_sin.sin_family = AF_INET;
62  memcpy((char*)&srv_sin.sin_addr, hp->h_addr, hp->h_length);
63  srv_sin.sin_port = SRV_PORT;
64  if (connect(s, (struct sockaddr*)&srv_sin, sizeof(srv_sin)) < 0) // Соединение с
        сервером
65  {
66      printf("Ошибка соединения сервером.\n");
67      finish();
68  }
69  while (1)
70  {
71      // Цикл работает также как в сервере, но последовательность ввода и вывода в
        сокет изменена
72      // Смотри текст программы-сервера
73      from_len = read(0, buf, BUF_SIZE);
74      write(s, buf, from_len);
75      if (from_len == 0)
76          finish();
77      write(1, GET, sizeof(GET));
78      from_len = read(s, buf, BUF_SIZE);
79      write(1, buf, from_len);
80      if (from_len == 0)
81          finish();
82  }
83  close(s); // Очистка памяти сокета
84  exit(0);
85 }

```
