



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

Сокеты.

Дисциплина «Операционные системы»

Студент: Зейналов З. Г.

Группа: ИУ7-61Б

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.

Листинг программы

Листинг 1 – Код программы сервера. server.c

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

int sock;

void signal_handler(int signum)
{
    printf("closing socket because of ctrl+C...\n");
    close(sock);
    unlink(SOCKET_NAME);
}

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

    sock = socket(AF_UNIX, SOCK_DGRAM, 0);
    if (sock < 0)
    {
        perror("socket error\n");
        return sock;
    }
    sock_adr.sa_family = AF_UNIX;
    strcpy(sock_adr.sa_data, SOCKET_NAME);
    if (bind(sock, &sock_adr, sizeof(sock_adr)) < 0)
    {
```

```

        printf("closing socket ...\n");
        close(sock);
        unlink(SOCKET_NAME);
        perror("binding error\n");
        return -1;
    }
    printf("server waits for clients...\n");
    signal(SIGINT, signal_handler);
    for (;;)
    {
        int rec = recv(sock, msg, sizeof(msg), 0);
        if (rec < 0)
        {
            close(sock);
            unlink(SOCKET_NAME);
            perror("message recieving error\n");
            return rec;
        }
        msg[recv] = 0;

        printf("Client message: %s", msg);
    }
    printf("closing socket ...\n");
    close(sock);
    unlink(SOCKET_NAME);
    return 0;
}

```

Листинг 2 – Код программы клиента. client.c

```

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

#include "info.h"

int main(void)
{
    int sockfd = socket(AF_UNIX, SOCK_DGRAM, 0);
    if (sockfd < 0)
    {
        printf("Error in socket();\n");
        return sockfd;
    }

    struct sockaddr server_addr;

```

```

server_addr.sa_family = AF_UNIX;
strcpy(server_addr.sa_data, SOCKET_NAME);

char msg[MSG_LEN];
sprintf(msg, "Hi, from !%d\n", getpid());
sendto(sockfd, msg, strlen(msg), 0, &server_addr, sizeof(server_addr));

close(sockfd);
return 0;
}

```

Листинг 3 – общая используемая часть кода.

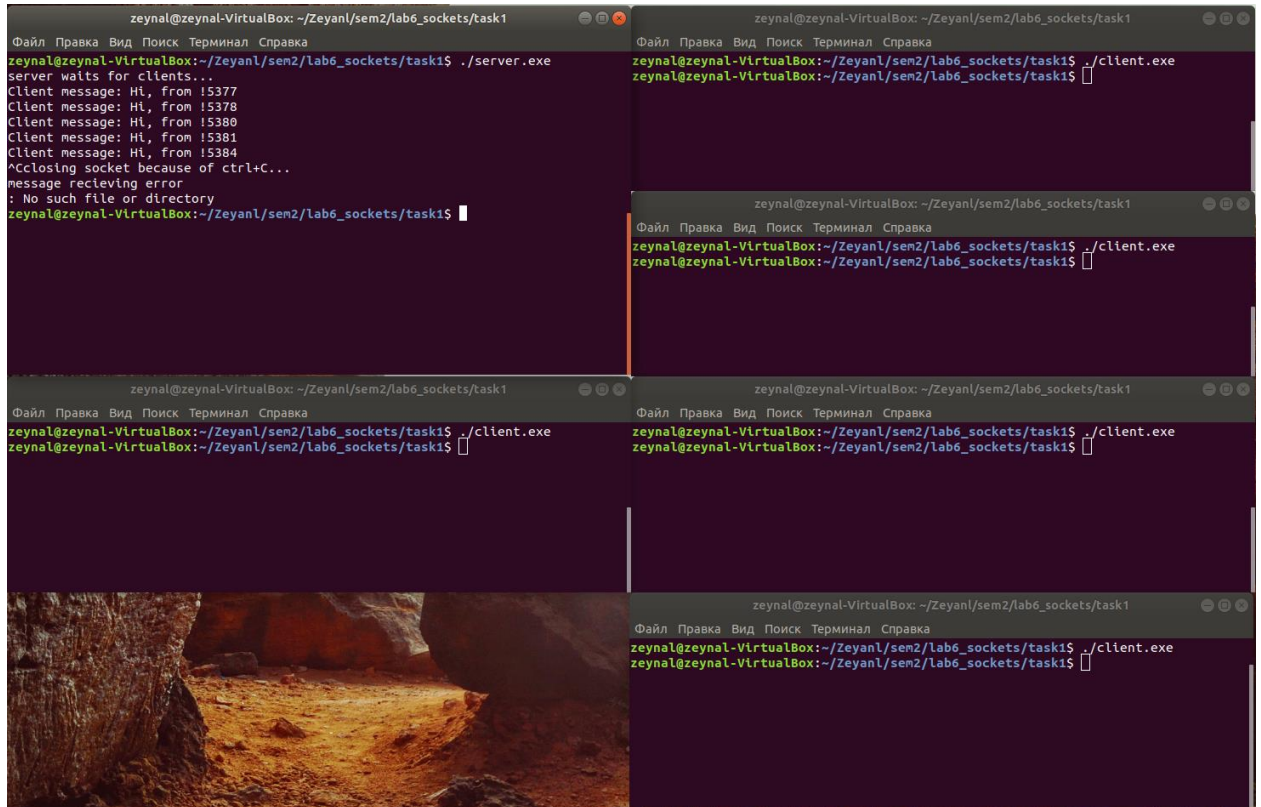
```

#ifndef __INFO_H__
#define __INFO_H__
#define SOCKET_NAME "socket.soc"
#define MSG_LEN 128

#endif

```

Вывод программы



```
zeynal@zeynal-VirtualBox: ~/Zeyanl/sem2/lab6_sockets/task1
Файл Правка Вид Поиск Терминал Справка
zeynal@zeynal-VirtualBox:~/Zeyanl/sem2/lab6_sockets/task1$ ./server.exe
server waits for clients...
Client message: Hi, from !5377
Client message: Hi, from !5378
Client message: Hi, from !5380
Client message: Hi, from !5381
Client message: Hi, from !5384
^Cclosing socket because of ctrl+C...
message recieving error
: No such file or directory
zeynal@zeynal-VirtualBox:~/Zeyanl/sem2/lab6_sockets/task1$

zeynal@zeynal-VirtualBox:~/Zeyanl/sem2/lab6_sockets/task1$ ./client.exe
zeynal@zeynal-VirtualBox:~/Zeyanl/sem2/lab6_sockets/task1$

zeynal@zeynal-VirtualBox:~/Zeyanl/sem2/lab6_sockets/task1$ ./client.exe
zeynal@zeynal-VirtualBox:~/Zeyanl/sem2/lab6_sockets/task1$

zeynal@zeynal-VirtualBox:~/Zeyanl/sem2/lab6_sockets/task1$ ./client.exe
zeynal@zeynal-VirtualBox:~/Zeyanl/sem2/lab6_sockets/task1$
```

В процессе-сервере создается сокет с помощью системного вызова `socket()`. Сокет семейства `AF_UNIX` с типом `SOCK_DGRAM`. С помощью системного вызова `bind()` происходит связка сокета с локальным адресом. Сервер блокируется на функции `recv()` в ожидании сообщения от процессов-клиентов.

В процессе клиенте создается сокет семейства `AF_UNIX` с типом `SOCK_DGRAM` с помощью системного вызова `socket()`. Затем с помощью функции `sendto()` отправляется сообщение к процессу-серверу.

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.

Листинг программы

Листинг 4 – Код программы сервера с использованием сетевого сокета.

```
#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 };
int sock;
int connection_handling(int current_fd)
{
    struct sockaddr_in addr;
    int addr_len = sizeof(addr);
    int fd;
    if ((fd = accept(sock, (struct sockaddr *) &addr, (socklen_t *) &addr_len)) <
    0)
    {
        close(sock);
        perror("Error in access\n");
        exit(-1);
    };

    printf("server got new connection by:\n fd = %d \n ip %s : %d", fd, inet_ntoa
(addr.sin_addr), ntohs(addr.sin_port));
```

```

    for (int i = 0; i < MAX_CLIENTS; i++)
    {
        if (clients[i] == 0)
        {
            clients[i] = fd;
            return 0;
        }
    }
    return -1;
}

void client_handling(int fd, int i)
{
    struct sockaddr_in addr;
    int addrlen = sizeof(addr);
    char msg[MSG_LEN];

    int ms_len = recv(fd, &msg, sizeof(msg), 0);
    if (ms_len == 0)
    {
        getpeername(fd, (struct sockaddr*) &addr, (socklen_t*) &addrlen);
        printf("User %d disconnected %s:%d \n", i, inet_ntoa(addr.sin_addr), ntohs(addr.sin_port));
        close(fd);
        clients[i] = 0;
    }
    else
    {
        msg[ms_len] = 0;
        printf("Server got message from client: %d = %s", i + 1, msg);
    }
}

int main (void)
{
    sock = socket(AF_INET, SOCK_STREAM, 0);
    if (sock < 0)
    {
        perror("socket creating error\n");
        return sock;
    }
    struct sockaddr_in serv_sock;
    serv_sock.sin_family = AF_INET;
    serv_sock.sin_addr.s_addr = INADDR_ANY;
    serv_sock.sin_port = htons(SOCK_PORT);
    if (bind(sock, (struct sockaddr *) &serv_sock, sizeof(serv_sock)) < 0)
    {
        perror("binding error \n");
        return -1;
    }
}

```

```

    if (listen(sock, 3) < 0)
    {
        perror("server can't listen");
        return -1;
    }
    printf("server active on ip %s, on port %d\n", inet_ntoa(serv_sock.sin_addr),
hton(serv_sock.sin_port));
    for (;;)
    {
        int max_fd = sock;
        fd_set set;
        FD_SET(sock, &set);

        for (int i = 0; i < MAX_CLIENTS; i++)
        {
            if (clients[i] > 0)
            {
                FD_SET(clients[i], &set);
            }

            if (clients[i] > max_fd)
                max_fd = clients[i];
        }
        int active_clients_count = select(max_fd + 1, &set, NULL, NULL, NULL);
        if (active_clients_count < 0)
        {
            perror("there is no active clients\n");
            return - 1;
        }
        if(FD_ISSET(sock, &set))
            if (connection_handling(sock) < 0)
            {
                perror("connection error[]\n");
                return -1;
            }
        for (int i = 0; i < MAX_CLIENTS; i++)
        {
            int client = clients[i];
            if (client > 0 && FD_ISSET(client, &set))
                client_handling(client, i);
        }
    }

    close(sock);
    return 0;
}

```


Листинг 5 – Код программа клиента с использованием сетевого сокета.

```
#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);    // /etc/hosts
    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 is here!\n", i);
        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;
}

```

Листинг 6 – общая используемая часть кода.

```

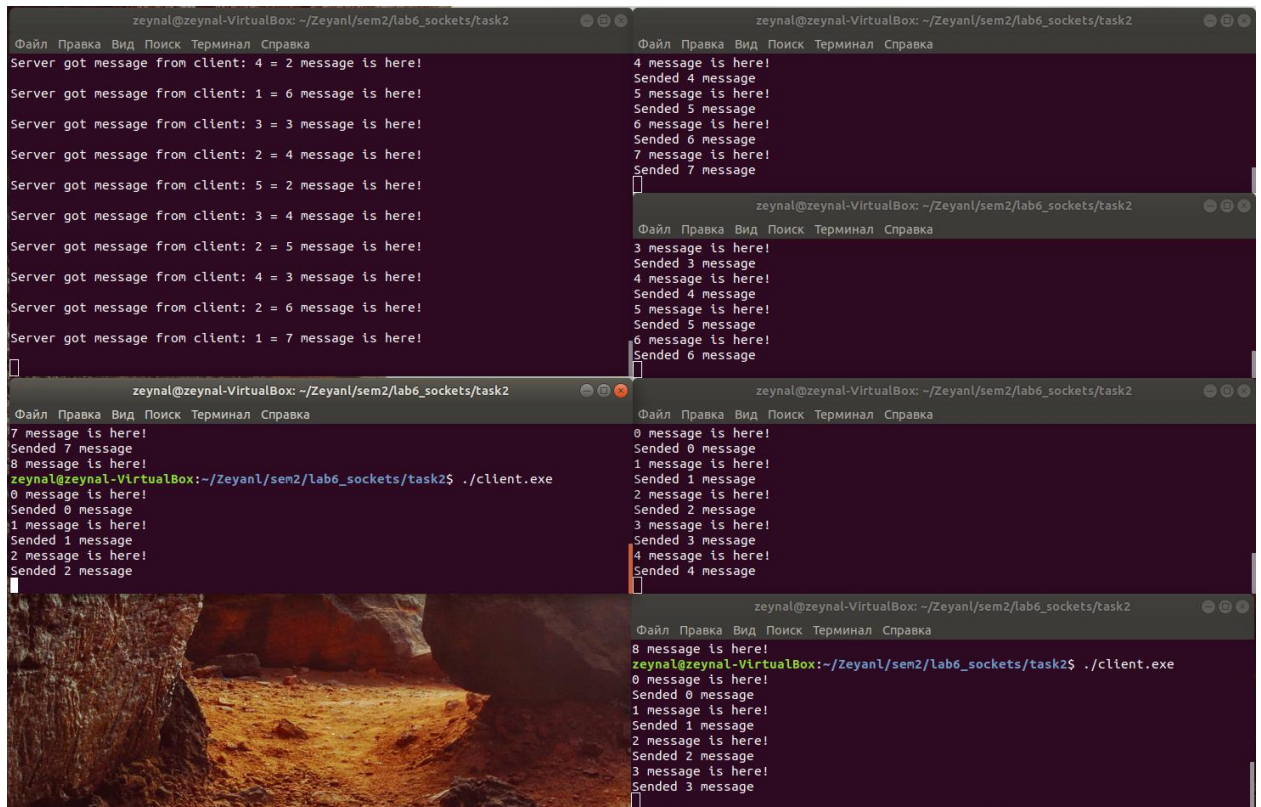
#ifndef INFO_H
#define INFO_H

#define MSG_LEN 256
#define SOCK_ADDR "localhost"
#define SOCK_PORT 8088

#endif // INFO_H

```

Вывод программы



```
zeynal@zeynal-VirtualBox: ~/ZeyanI/sem2/lab6_sockets/task2
Файл Правка Вид Поиск Терминал Справка
Server got message from client: 4 = 2 message is here!
Server got message from client: 1 = 6 message is here!
Server got message from client: 3 = 3 message is here!
Server got message from client: 2 = 4 message is here!
Server got message from client: 5 = 2 message is here!
Server got message from client: 3 = 4 message is here!
Server got message from client: 2 = 5 message is here!
Server got message from client: 4 = 3 message is here!
Server got message from client: 2 = 6 message is here!
Server got message from client: 1 = 7 message is here!
[]

zeynal@zeynal-VirtualBox: ~/ZeyanI/sem2/lab6_sockets/task2
Файл Правка Вид Поиск Терминал Справка
4 message is here!
Sended 4 message
5 message is here!
Sended 5 message
6 message is here!
Sended 6 message
7 message is here!
Sended 7 message
[]

zeynal@zeynal-VirtualBox: ~/ZeyanI/sem2/lab6_sockets/task2
Файл Правка Вид Поиск Терминал Справка
3 message is here!
Sended 3 message
4 message is here!
Sended 4 message
5 message is here!
Sended 5 message
6 message is here!
Sended 6 message
[]

zeynal@zeynal-VirtualBox: ~/ZeyanI/sem2/lab6_sockets/task2
Файл Правка Вид Поиск Терминал Справка
7 message is here!
Sended 7 message
8 message is here!
zeynal@zeynal-VirtualBox:~/ZeyanI/sem2/lab6_sockets/task2$ ./client.exe
0 message is here!
Sended 0 message
1 message is here!
Sended 1 message
2 message is here!
Sended 2 message
[]

zeynal@zeynal-VirtualBox: ~/ZeyanI/sem2/lab6_sockets/task2
Файл Правка Вид Поиск Терминал Справка
0 message is here!
Sended 0 message
1 message is here!
Sended 1 message
2 message is here!
Sended 2 message
3 message is here!
Sended 3 message
4 message is here!
Sended 4 message
[]

zeynal@zeynal-VirtualBox: ~/ZeyanI/sem2/lab6_sockets/task2
Файл Правка Вид Поиск Терминал Справка
8 message is here!
zeynal@zeynal-VirtualBox:~/ZeyanI/sem2/lab6_sockets/task2$ ./client.exe
0 message is here!
Sended 0 message
1 message is here!
Sended 1 message
2 message is here!
Sended 2 message
3 message is here!
Sended 3 message
[]
```

В процессе-сервере создается сетевой сокет вызовом `socket()` семейства `AF_INET` с типом `SOCK_STREAM`. Далее, с помощью вызова `bind()` сокет связывается с адресом, прописанным в `SOCKET_ADDRESS`. После этого сервер переводится в режим ожидания на системном вызове `listen()`, ожидая запроса на соединение. На каждом шаге цикла создается новый набор дескрипторов, в него заносятся сокет сервера с помощью макроса `FD_SET`. Затем сервер блокируется на вызове функции `select()`, которая возвращает управление в функцию при получении хотя бы одного запроса от клиента. При выходе из блокировки проверяется наличие новых соединений и при их наличии вызывается функция `connection_handling()`, внутри которой с помощью вызова `accept()` принимается новое соединение и создается сокет, который записывается в массив файловых дескрипторов. После осуществляется обход по массиву дескрипторов и если дескриптор находится в наборе дескрипторов, то с помощью функции `client_handling()` осуществляется считывание с помощью функции `recv()` и вывод сообщения от клиента на сервере. Если `recv()` вернула нулевое значение, значит соединение было сброшено, выводится сообщение о закрытии соединения и закрытие сокета.

В процессе-клиенте создается сетевой сокет семейства AF_INET с типом SOCK_STREAM. Функция gethostbyname() преобразует доменный адрес в сетевой, благодаря которому можно установить соединение с помощью функции connect(). Затем в цикле происходит отправка сообщений серверу.