|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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\_\_

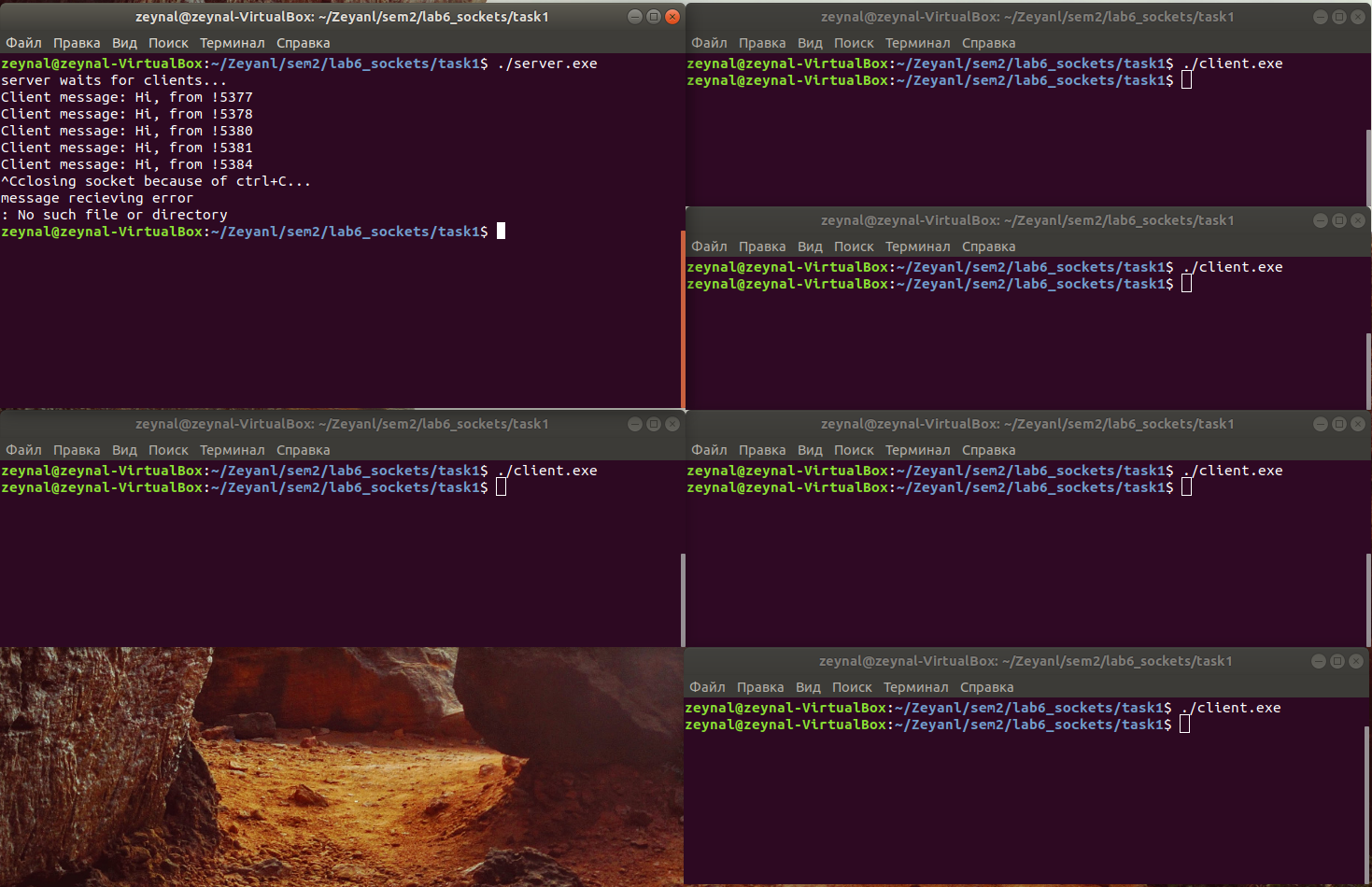
#define \_\_INFO\_H\_\_

#define SOCKET\_NAME "socket.soc"

#define MSG\_LEN 128

#endif

**Вывод программы**



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 семейста 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), htons(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

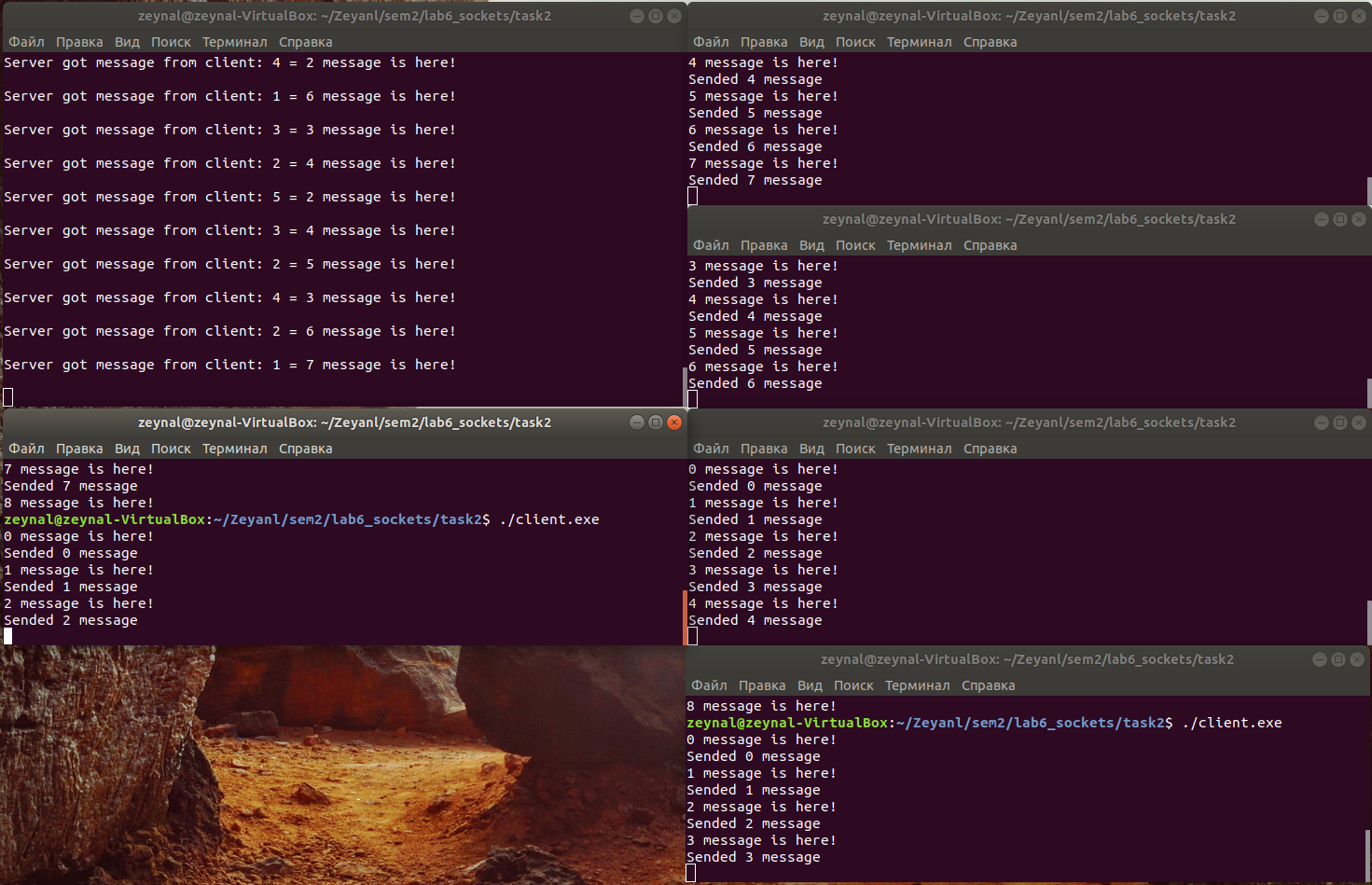
#define MSG\_LEN 256

#define SOCK\_ADDR "localhost"

#define SOCK\_PORT 8088

#endif // INFO\_H

**Вывод программы**



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\_handing(), внутри которой с помощью вызова accept() принимается новое соединение и создается сокет, который записывается в массив дескрипторов. После осуществляется обход по массиву дескрипторов и если дескриптор находится в наборе дескрипторов, то с помощью функции client\_handling() осуществляется считывание c 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.