



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

Тема: Сокеты

Студент Садулаева Т. Р.

Группа ИУ7-646

Преподаватель Рязанова Н.Ю.

Москва.
2021 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.

Код программ.

Листинг 1. Server.c

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

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

void sigint_handler(int signum)
{
    close(sock);
    unlink(SOCKET_NAME);
    printf("Socket was closed by ctrl+c!\n");
}

int main(void)
{
    struct sockaddr addr;

    sock = socket(AF_UNIX, SOCK_DGRAM, 0); if
(sock < 0)
    {
        perror("Can't open socket!");
        exit(1);
    }

    addr.sa_family = AF_UNIX;
    strcpy(addr.sa_data, SOCKET_NAME);
```

```

if (bind(sock, &addr, sizeof(addr)) < 0)
{
    printf("Can't bind name to socket!\n");
    close(sock);
    unlink(SOCKET_NAME);
    perror("Error in bind() ");
    exit(-1);
}

printf("\nServer is waiting\n");
signal(SIGINT, sigint_handler);

char msg[MSG_LEN];
while(1)
{
    int recievedSize = recv(sock, msg, sizeof(msg), 0);if
    (recievedSize < 0)
    {
        close(sock);
        unlink(SOCKET_NAME);
        perror("Error in recv(): ");
        return recievedSize;
    }

    msg[recievedSize] = 0;
    printf("Client send: %s\n", msg);
}

printf("Closing socket\n");
close(sock);
unlink(SOCKET_NAME);
return 0;
}

```

Листинг 2. Client.c

```

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

#include "info.h"

int main(void)
{
    int sockfd = socket(PF_LOCAL, SOCK_DGRAM, 0);if
    (sockfd < 0)
    {
        perror("Can't open socket!");
        exit(1);
    }
}

```

```

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

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

close(sockfd); return
0;
}

```

```

Файл Правка Вид Поиск Терминал Справка
~$ cd iu7/sem6/os/lab6/part1
~/iu7/sem6/os/lab6/part1$ ./client.o
~/iu7/sem6/os/lab6/part1$

Файл Правка Вид Поиск Терминал Справка
~$ cd iu7/sem6/os/lab6/part1
~/iu7/sem6/os/lab6/part1$ ./client.o
~/iu7/sem6/os/lab6/part1$

Файл Правка Вид Поиск Терминал Справка
~$ cd iu7/sem6/os/lab6/part1
~/iu7/sem6/os/lab6/part1$ ./client.o
~/iu7/sem6/os/lab6/part1$

Файл Правка Вид Поиск Терминал Справка
~$ cd iu7/sem6/os/lab6/part1
~/iu7/sem6/os/lab6/part1$ ./client.o
~/iu7/sem6/os/lab6/part1$

Файл Правка Вид Поиск Терминал Справка
~$ cd iu7/sem6/os/lab6/part1
~/iu7/sem6/os/lab6/part1$ ./client.o
~/iu7/sem6/os/lab6/part1$

Файл Правка Вид Поиск Терминал Справка
~$ cd iu7/sem6/os/lab6/part1
~/iu7/sem6/os/lab6/part1$ ./server.o
~/iu7/sem6/os/lab6/part1$

Server is waiting
Client send: Hello from client with pid 20692
Client send: Hello from client with pid 20907
Client send: Hello from client with pid 21121
Client send: Hello from client with pid 21336
Client send: Hello from client with pid 21551
^C Socket was closed by ctrl+c!
Error in recv(): No such file or directory
~/iu7/sem6/os/lab6/part1$

```

Рис 1. Пример работы программы.

```

total 64
drwxrwxr-x 2 v v 4096 июн 15 11:55 .
drwxrwxr-x 4 v v 4096 июн 9 13:28 ..
-rw-rw-r-- 1 v v 877 июн 15 11:47 client.c
-rwxrwxr-x 1 v v 17016 июн 15 11:55 client.out
-rw-rw-r-- 1 v v 128 июн 9 13:30 Makefile
srwxrwxr-x 1 v v 0 июн 15 11:55 my_socket.sock
-rw-rw-r-- 1 v v 1643 июн 15 11:49 server.c
-rwxrwxr-x 1 v v 17128 июн 15 11:55 server.out
-rw-rw-rw- 1 v v 222 июн 9 13:29 sockets.h

```

Рис. 2. Сокет в файловой системе.

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

В процессе-клиенте создается сокет семейства `AF_UNIX` с типом `SOCK_DGRAM` с помощью системного вызова `socket()`. 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.

Код программ.

Листинг 3. Server.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>

#define MSG_LEN 256
#define SOCK_ADDR "localhost"
#define SOCK_PORT 9999

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

void connectionHandler(unsigned int fd)
{
    struct sockaddr_in addr;
    int addrSize = sizeof(addr);

    int incom = accept(fd, (struct sockaddr*)&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(addr.sin_addr), ntohs(addr.sin_port));

    for (int i = 0; i < MAX_CLIENTS; i++)
    {
        if (clients[i] == 0)
        {
            clients[i] = incom;
            break;
        }
    }
}
```

```

void clientHandler(unsigned int fd, unsigned int client_id)
{
    char msg[MSG_LEN];
    memset(msg, 0, MSG_LEN);

    struct sockaddr_in addr;
    int addrSize = sizeof(addr);

    int recvSize = recv(fd, msg, MSG_LEN, 0); if
    (recvSize == 0)
    {
        getpeername(fd, (struct sockaddr*) &addr, (socklen_t*) &addrSize);
        printf("User %d disconnected %s:%d \n", client_id,
inet_ntoa(addr.sin_addr), ntohs(addr.sin_port));
        close(fd);
        clients[client_id] = 0;
    }
    else
    {
        msg[recvSize] = '\0';
        printf("Message from %d client: %s\n", client_id, msg);
    }
}

int main(void)
{
    int sock = socket(AF_INET, SOCK_STREAM, 0); if
    (sock < 0)
    {
        perror("Error in sock\n");
        return sock;
    }

    struct sockaddr_in addr;
    addr.sin_family = AF_INET;
    addr.sin_port = htons(SOCK_PORT);
    addr.sin_addr.s_addr = INADDR_ANY; //any address for binding

    if (bind(sock, (struct sockaddr*) &addr, sizeof(addr)) < 0)
    {
        perror("Error in bind\n");
        return -1;
    }
    printf("Server is listening on the %d port!\n", SOCK_PORT);

    if (listen(sock, 3) < 0)
    {
        perror("Error in listen(): ");
        return -1;
    }
    printf("Wait for the connections\n");

    while (1)
    {
        fd_set set;
        int max_fd = sock;

        FD_ZERO(&set);

```

```

        FD_SET(sock, &set);

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

        max_fd = (clients[i] > max_fd) ? (clients[i]) : (max_fd);
    }

    int active_clients_count = select(max_fd + 1, &set, NULL, NULL, NULL);

    if (active_clients_count < 0)
    {
        perror("No active clients");
        return active_clients_count;
    }

    if (FD_ISSET(sock, &set))
    {
        connectionHandler(sock);
    }

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

return 0;
}

```

Листинг 4. Client.c

```

#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>

#define MSG_LEN 256
#define SOCK_ADDR "localhost"
#define SOCK_PORT 9999

int main(void)
{

```

```

srand(time(NULL));

int sock = socket(AF_INET, SOCK_STREAM, 0); if
(sock < 0)
{
    perror("Error in sock\n");
    return sock;
}

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

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

if (connect(sock, (struct sockaddr*) &addr, sizeof(addr)) < 0)
{
    perror("Error in connect\n");
    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;
}

```

Рис 2. Демонстрация работы программы

The image shows five terminal windows arranged in a grid-like fashion, demonstrating the execution of a client program named `./client.o`. The windows show the following content:

- Top-left window:** Shows the initial connection and the first two messages being sent and received. It includes details like `fd = 5` and `ip = 127.0.0.1:45620`.
- Top-middle window:** Shows the third and fourth messages being sent and received, with `fd = 6` and `ip = 127.0.0.1:45624`.
- Top-right window:** Shows the fifth and sixth messages being sent and received, with `fd = 7` and `ip = 127.0.0.1:45626`.
- Bottom-left window:** Shows the seventh and eighth messages being sent and received, with `fd = 8` and `ip = 127.0.0.1:45628`. It also shows a user disconnecting at `127.0.0.1:45618`.
- Bottom-right window:** Shows the ninth and tenth messages being sent and received, with `fd = 9` and `ip = 127.0.0.1:45630`.

В процессе-клиенте создается сокет семейства AF_INET с типом SOCK_STREAM с помощью системного вызова `socket()`. С помощью функции `gethostbyname()` доменный адрес преобразуется в сетевой и с его помощью можно установить соединение, используя функцию `connect()`. Затем происходит отправка сообщений серверу.