

КАФЕДРА Системы автоматизированного проектирования (РК-6)

Тема лабораторной работы    Сетевое программирование

## Оценка

*Москва, 2022 г.*

## Оглавление

Текст задания .....	3
Описание используемого прикладного протокола сетевого взаимодействия .....	3
Описание структуры программы.....	3
Описание основных используемых структур данных.....	4
Блок-схема программы .....	4
Примеры результатов работы программы.....	7
Текст программы .....	7

### **Текст задания**

Разработать клиент-серверное приложение копирования файла (или поддерева файловой системы) с узла-сервера на узел-клиент в указанный каталог (аналог стандартной UNIX-команды `rcp`). Команда, выполняемая на стороне клиента, имеет следующий вид: `remcp host@path.to.src.file path.to.dst.dir`.

Замечание. Для передачи поддерева файловой системы (или одного файла) рекомендуется на стороне клиента использовать вызов (через `fork`) команды `tar` с перенаправлением ее вывода непосредственно в сокет. На стороне сервера также рекомендуется использовать `tar` с перенаправлением ее ввода непосредственно из сокета.

### **Описание используемого прикладного протокола сетевого взаимодействия**

В программе используется протокол TCP. Данный протокол работает на основе потока данных с установкой соединения и гарантированной передаче данных – при потере данных осуществляется повторный запрос.

Установку соединения обеспечивает механизм сокетов. После создания, сокет необходимо связать с коммуникационной средой. После создания и привязки сокета серверу необходимо прослушивать входящие соединения. Когда на сервер стучится клиент, серверу необходимо принять соединение клиента и установить с ним связь. В свою очередь, клиенту, после создания и привязки сокета необходимо отправить запрос серверу на возможность подключения.

В процессе взаимодействия клиент и сервер посылают друг другу данные. В конце подключения разрывается.

### **Описание структуры программы**

Программа-клиент отправляет название директории, которую необходимо копировать, на сервер. Далее она создает и открывает пустой файл, в который записывает данные, которые присылает сервер. Затем программа-клиент распаковывает полученный из сокета архив и удаляет его с помощью команды `tar`.

Программа-сервер получает от клиента название директории, которую необходимо передать. Программа архивирует файл с помощью команды tar, затем сервер отправляет на клиент данные созданного архива путём записи в сокет, который затем отправляется на клиента. После того, как весь архив отправлен, закрывает соединение с клиентом и ждёт следующего подключения.

Изображение взаимодействия клиента и сервера представлено на рис.1.

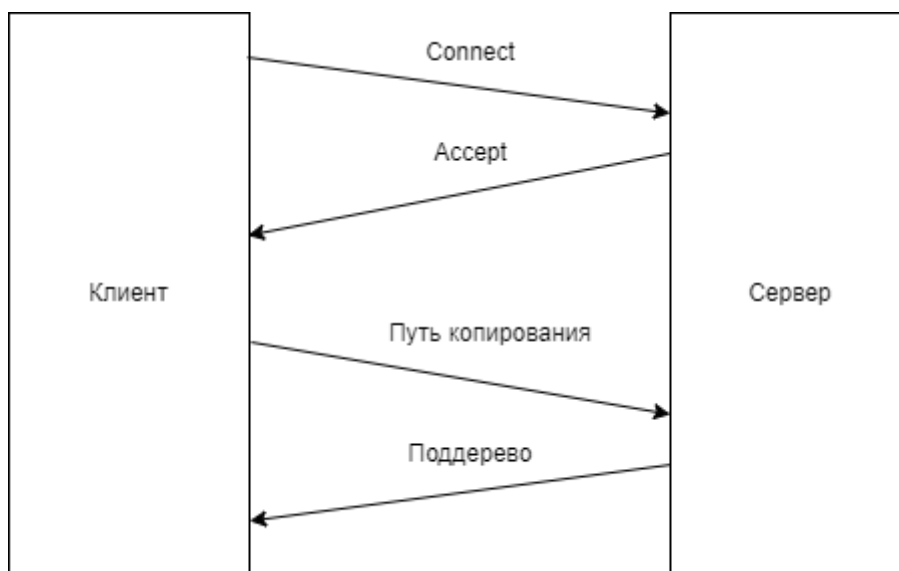


Рисунок 1. Схема взаимодействия клиента и сервера

### Описание основных используемых структур данных

С помощью define задаются порты клиента и сервера, размер буфера, адрес сервера и максимальная длина пути.

int s – дескриптор сокета;

int s\_new – дескриптор сокета для связи с клиентом;

char path[] – путь до директории (передаваемый в качестве аргумента командной строки);

struct sockaddr\_in clnt\_sin, sin – структура, содержащая сведения об адресе клиента/сервера (протокол, IP-адрес, порт);

char data[] – массив, в который считывается информация из сокета;

### Блок-схема программы

Блок-схема реализованных клиента и сервера представлены на рисунке 2 и 3.



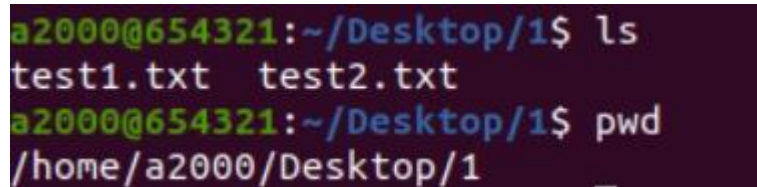
Рисунок 2. Блок-схема клиента



Рисунок 3. Блок-схема сервера

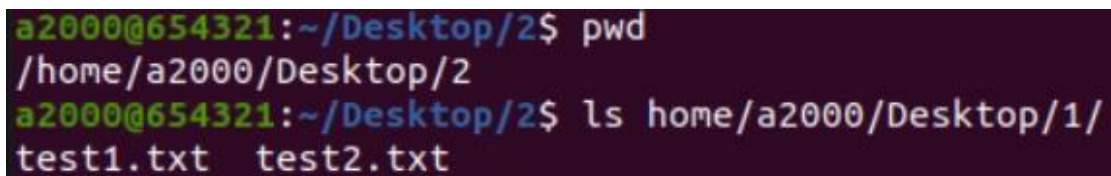
## Примеры результатов работы программы

Результатом работы разработанной программы с пустой папкой 2 при аргументах 127.0.0.1@/home/.../Desktop/1 /home/.../Desktop/2 представлены на рисунках 4, 5.



```
a2000@654321:~/Desktop/1$ ls
test1.txt test2.txt
a2000@654321:~/Desktop/1$ pwd
/home/a2000/Desktop/1
```

Рисунок 4. Начальное расположение файлов



```
a2000@654321:~/Desktop/2$ pwd
/home/a2000/Desktop/2
a2000@654321:~/Desktop/2$ ls home/a2000/Desktop/1/
test1.txt test2.txt
```

Рисунок 5. Результат работы программы

## Текст программы

Код клиента:

```
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>

#define SRV_PORT 1234 // порт сервера
#define CLNT_PORT 1235 // порт клиента
#define PATH_SIZE 64 // размер пути
#define TAR_C 256 // размер команды архиватора
#define TAR "tar -x <&" // утилита архиватор prev

int main (int argc, char** argv) {
```

```

int s;
char path[PATH_SIZE]; // путь
char *path_p;
char tar[TAR_C];
char host[TAR_C]; // адрес сервера
char data[20];
struct sockaddr_in clnt_sin, srv_sin;
if (argc < 3) {
    write(1, "Need more arguments\n", 23);
    exit(1);
}

s = socket(AF_INET, SOCK_STREAM, 0);
memset((char *) &clnt_sin, '\0', sizeof(clnt_sin));

clnt_sin.sin_family = AF_INET;
clnt_sin.sin_addr.s_addr = INADDR_ANY;
clnt_sin.sin_port = CLNT_PORT;
bind(s, (struct sockaddr *) &clnt_sin, sizeof(clnt_sin));

memcpy(host, argv[1], strlen(argv[1]));
*strchr(host, '@') = '\0';
memset((char *) &srv_sin, '\0', sizeof(srv_sin));

if ((gethostbyname(host)) == NULL) { // получаем host
    write(1, "Can't find ", 11);
    write(1, host, strlen(host));
    write(1, "\n", 1);
    return 1;
}

```



```

srv_sin.sin_family = AF_INET;
memcpy((char *) &srv_sin.sin_addr, hp->h_addr, hp->h_length);
srv_sin.sin_port = SRV_PORT;

```

```

if (connect(s, (struct sockaddr *) &srv_sin, sizeof(srv_sin)) == -1) { //

```

ПОДКЛЮЧЕНИЕ

```

    write(1, "Can't connect to ", 17);
    write(1, host, strlen(host));
    write(1, "\n", 1);
    return 1;
}

```

```

path_p = strstr(argv[1], "@");
path_p++;
memcpy(path, path_p, strlen(path_p));
path[strlen(path_p)] = '\0';

```

```

write(s, path, strlen(path));
if (chdir(argv[2]) < 0) {
    return (-4);
}

```

```

sprintf(data, "%d", s);
memcpy(tar, TAR, strlen(TAR));
memcpy(tar + strlen(TAR), data, strlen(data));
tar[strlen(TAR) + strlen(data)] = '\0';
system(tar);

```

```

exit(0);

```

```
}
```

Код сервера:

```
#include <unistd.h>
```

```
#include <sys/socket.h>
```

```
#include <netinet/in.h>
```

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
#include <string.h>
```

```
#include <signal.h>
```

```
#define SRV_PORT 1234 // порт подключения
```

```
#define BUF_SIZE 4096 // размер буфера
```

```
#define PATH_SIZE 64 // размер пути
```

```
#define TAR_C 128 // длина команды tar
```

```
#define TAR "tar -c -O " // команда архивации
```

```
#define REDIRECT " 1>&"
```

```
int s, s_new = 0;
```

```
int from_len;
```

```
char path[PATH_SIZE];
```

```
char tar[TAR_C];
```

```
void siginhandler() { // обработчик Ctrl+C, грамотное отключение сервера
```

```
    write(1, "\nServer is shutting down\n", 25);
```

```
    shutdown(s_new, 0);
```

```
    shutdown(s, 2);
```

```
    if(s_new) close(s_new);
```

```
    exit(0);
```

```
}
```

```

int main () {
    int i;
    char data[20];
    struct sockaddr_in sin;

    signal(SIGINT, &siginthandler); // установка обработчика SIGINT

    s = socket(AF_INET, SOCK_STREAM, 0); // установка сокета
    memset((char *)&sin, '\0', sizeof(sin)); // обнуление структуры
    sin.sin_family = AF_INET;
    sin.sin_addr.s_addr = INADDR_ANY;
    sin.sin_port = SRV_PORT;
    bind(s, (struct sockaddr *)&sin, sizeof(sin));
    listen(s, 3);

    while(1) {
        struct sockaddr_in from_sin;
        int path_1;

        for(i = 0; i < PATH_SIZE; i++) {
            path[i] = '\0';
        }

        from_len = sizeof(from_sin);
        s_new = accept(s, (struct sockaddr *) &from_sin, &from_len);

        path_1 = read(s_new, path, PATH_SIZE);
        if (path_1 == 0) {
            continue;
        }
    }
}

```

```

chdir(path);

sprintf(data,"%d", s_new);

memcpy(tar, TAR, strlen(TAR));
memcpy(tar + strlen(TAR), path, strlen(path));
memcpy(tar + strlen(TAR) + strlen(path), REDIRECT,
strlen(REDIRECT));
memcpy(tar + strlen(TAR) + strlen(path) + strlen(REDIRECT), data,
strlen(data));
tar[strlen(TAR) + strlen(path) + strlen(REDIRECT) + strlen(data)] = '\0';

system(tar);
write(1, path, strlen(path));

shutdown(s_new, 2); // отключение подключения

close(s_new);
s_new = 0;
}
}

```