Сети ЭВМ и телекоммуникации

Е. А. Никитин

29 декабря 2015 г.

Глава 1

Задание

Разработать приложение-клиент и приложение сервер, обеспечивающие функции обмена файлами.

1.1 Функциональные требования

Серверное приложение должно реализовывать следующие функции:
1) Прослушивание определенного порта
2) Обработка запросов на подключение по этому порту от клиентов
3) Поддержка одновременной работы нескольких клиентов через механизм нитей
4) Приём файла от клиента
5) Передача по запросу клиента списка файлов текущего каталога
6) Приём запросов на передачу файла и передача файла клиенту
7) Навигация по системе каталогов
8) Обработка запроса на отключение клиента
9) Принудительное отключение клиента
Клиентское приложение должно реализовывать следующие функции:
1) Установление соединения с сервером
2) Получение от сервера списка файлов каталога
3) Операции навигации по системе каталогов
4) Передача файла серверу
5) Приём файла от сервера
6) Разрыв соединения

1.2 Нефункциональные требования

Т.к. для выполнения данного задания требовалось изучить библиотеки, преднозначенные для работы с файловой системой как ОС Windiws, так ОС Linux, задача была упрощена - сервер написан для работы на Linux. Клиентское прложение создано для ОС Windows.

1.3 Накладываемые ограничения

Максимальный размер буфера - 1024, поэтому рабочая директория сервера, а также пути, которые будет указывать пользователь должны укладываться в данную длину.
Ограничение по одновременно подключенным пользователям - 10.
В некоторых функциях в качестве разделителя используется знак поэтому его использование в названиях и путях нежелательно, т.к. может привести к неправильной работе приложения.

Глава 2

Реализация для работы по протоколу TCP

2.1 Прикладной протокол

В данной реализации для некоторых команд был создан собственный протокол, отличный от реализации отправки в несколько посылок, однако реализация протокола труднее в описании.

Описание команд:
1) view - получить дерево каталогов всего рабочего пространства. Функция рекурсивно проходит все вложенные папки рабочей директории. Посылки отправляются в формате: "отступ"символ папки или файла"название". После использования данной команды текущая директори меняется на корень рабочей папки.
2)view_dir - команда получения списка фалов и папок, но внутри текущей директории. Используется протокол команды view.
3)get_dir - команда получения текущей директории. Сообщение приходит в виде пути в файловой системе.
4)ch_dir - изменение текущего каталога. Послыка формируется в виде пути относительно текущей папки.
5)mk_dir - создание папки. Папка будет создана в текущем каталоге. Послыка формируется в виде названия папки, ответ в виде успешного/-неуспешного создания.
6)rm_dir - команда удаления папки из текущего каталога.
7)help - получение списка и формата команд.
8)upload - функция загрузки файла на сервер. Работа в следующей последовательности: пользователь задает путь -> отправляется размер файла

-> отправляется название (как будет называться на сервере), на сервере формируется путь -> запоминается время начала отправки -> отправялется файл -> высчитывается время передачи.
9)download - функция загрузки с сервера. Работа в следующей последовательности: потправляется название файла -> принимается подтверждение открытия файла на сервере -> указывается путь, куда сохранять(формируется путь) -> приимается размер файла -> запоминается время начала отправки -> отправялется файл -> высчитывается время передачи.

2.2 Архитектура приложения

Архитектураприложения представлена на рис. 2.1:

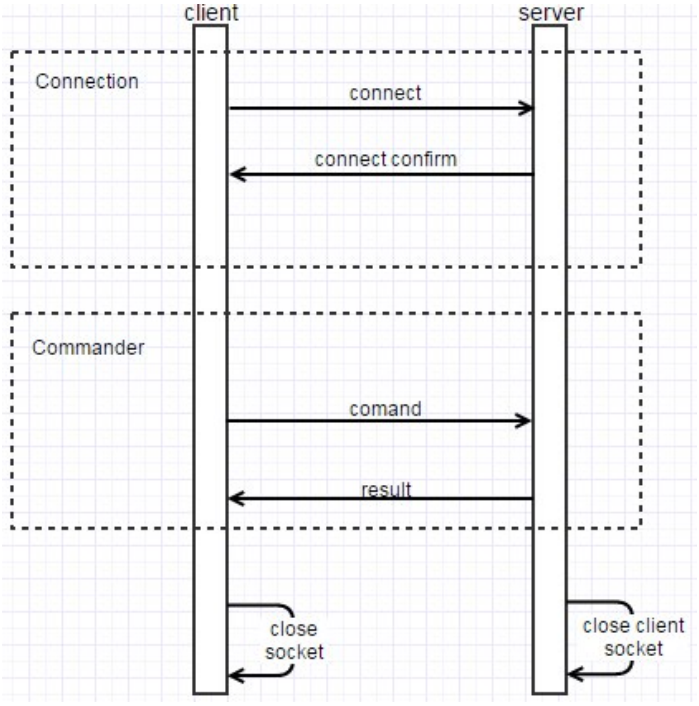

Рис. 2.1: Архитектура приложения

2.3 Тестирование

2.3.1 Описание тестового стенда и методики тестирования

2.3.2 Тестовый план и результаты тестирования

1) Тест1: ввод команд без подключения(рис. 2.2)
2) Тест2: обработка неверных команд(рис. 2.3)
3) Тест3: неправильный путь при загрузке файла на сервер(рис. 2.4)
4) Тест4: получение дерева каталогов(рис. 2.5)
5) Тест5: получения файла с сервера(рис. 2.6)
6) Тест6: отправка файла на сервер(рис. 2.7)

Результаты тестирования:

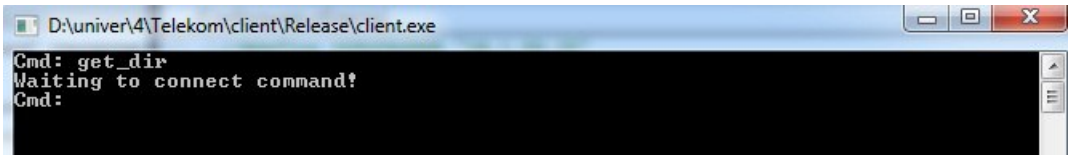

Рис. 2.2: Ввод команд без подключения

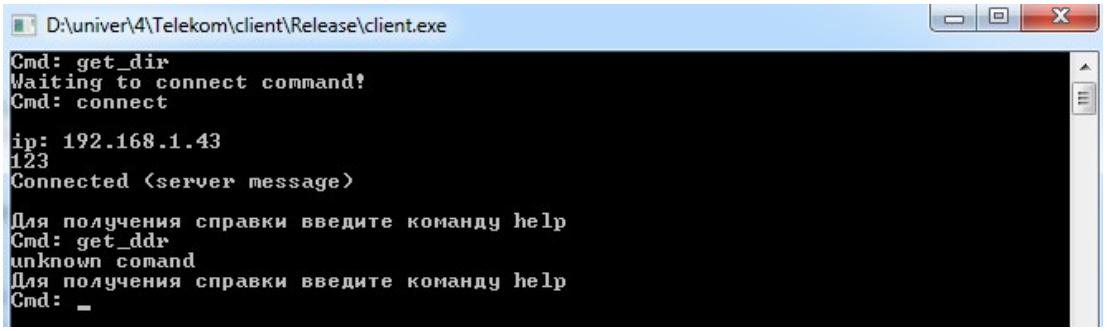

Рис. 2.3: Обработка неверных команд

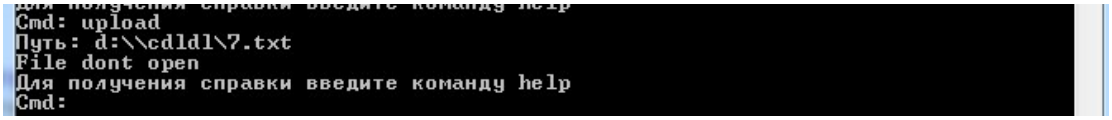

Рис. 2.4: Неправильный путь при загрузке файла на сервер

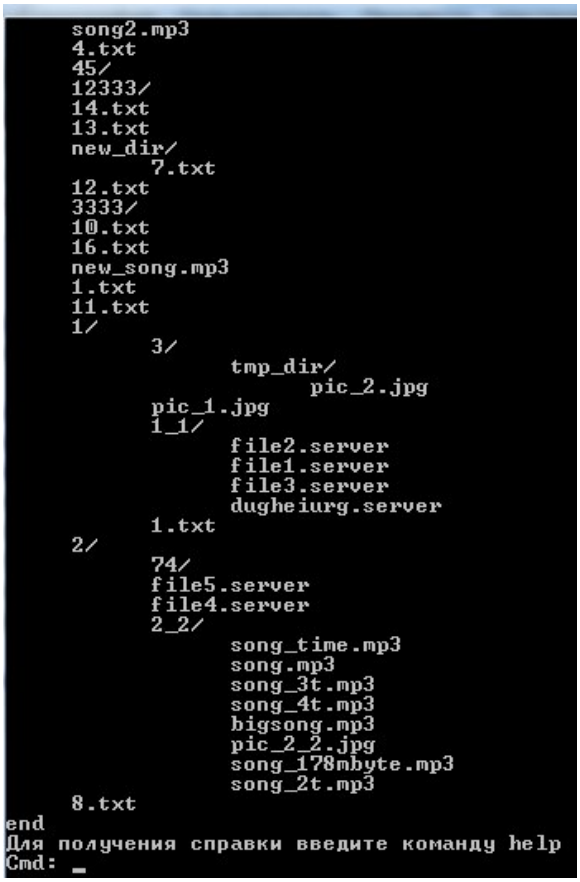

Рис. 2.5: Получение дерева каталогов

Рис. 2.6: Получения файла с сервера

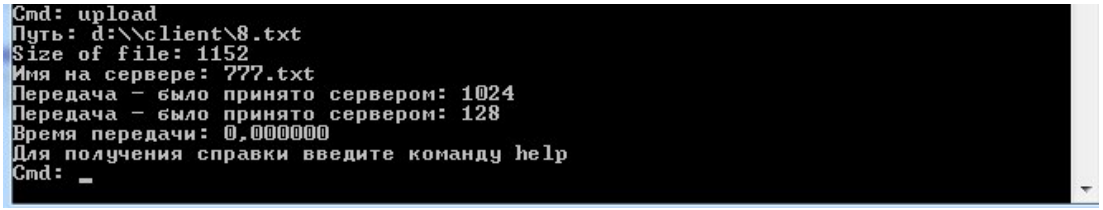

Рис. 2.7: Отправка файла на сервер

Глава 3

Реализация для работы по протоколу UDP

3.1 Прикладной протокол

Протоколы команд совпадют с протоколами в реализации TCP, кроме команд upload и download, в которых добавлена проверка на целостность полученного файла. После передачи дополнительная проверка и отправление результата в виде строки "error"/"ок".

3.2 Архитектура приложения

Архитектура приложения относительно реализации TCP немного изменилась. Обработчик команд был обособлен в функцию диспетчера. До диспетчера устанавливается соединение и создание новых сокетов для клиентов. Архитектура представлена на рисунке 3.1.

3.3 Тестирование

3.3.1 Описание тестового стенда и методики тестирования

3.3.2 Тестовый план и результаты тестирования

Тестирование проведено согласно плану тестирования TCP приложений.
Результаты совпали.
Тестирование с помощью команды tc.

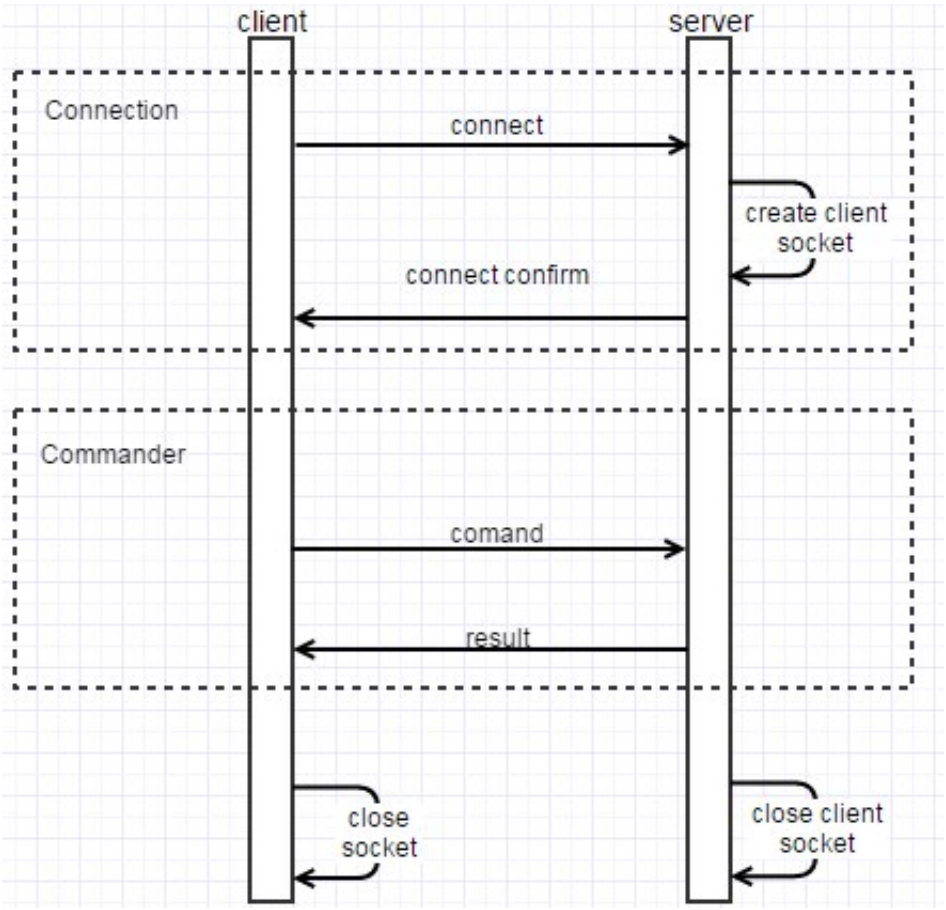

Рис. 3.1: Архитектура приложения

1) Введение задержек командой
tc qdisc change dev eth0 root netem delay 100ms 10ms
Результат представлен на рис. 3.2.
В результате приложение работает нормально, но посылки приходят медленнее. 2)Введение потери пакетов командой
tc qdisc change dev eth0 root netem loss 35Результат представлен на рис. 3.3.
В резульате приложение работает некорректно, т.к. ожидает посылку сервера.

3)Введение дупликации пакетов командой
tc qdisc change dev eth0 root netem duplicate 35Результат представлен на рис. 3.4.
В резульате приложение работает некорректно, т.к. приходят повторя-

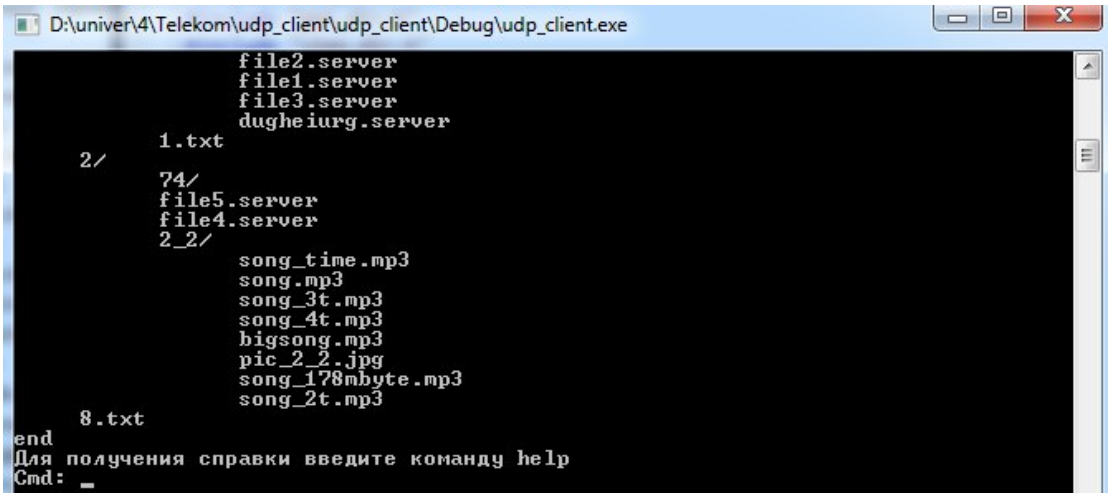

Рис. 3.2: Результаты при введении задержек

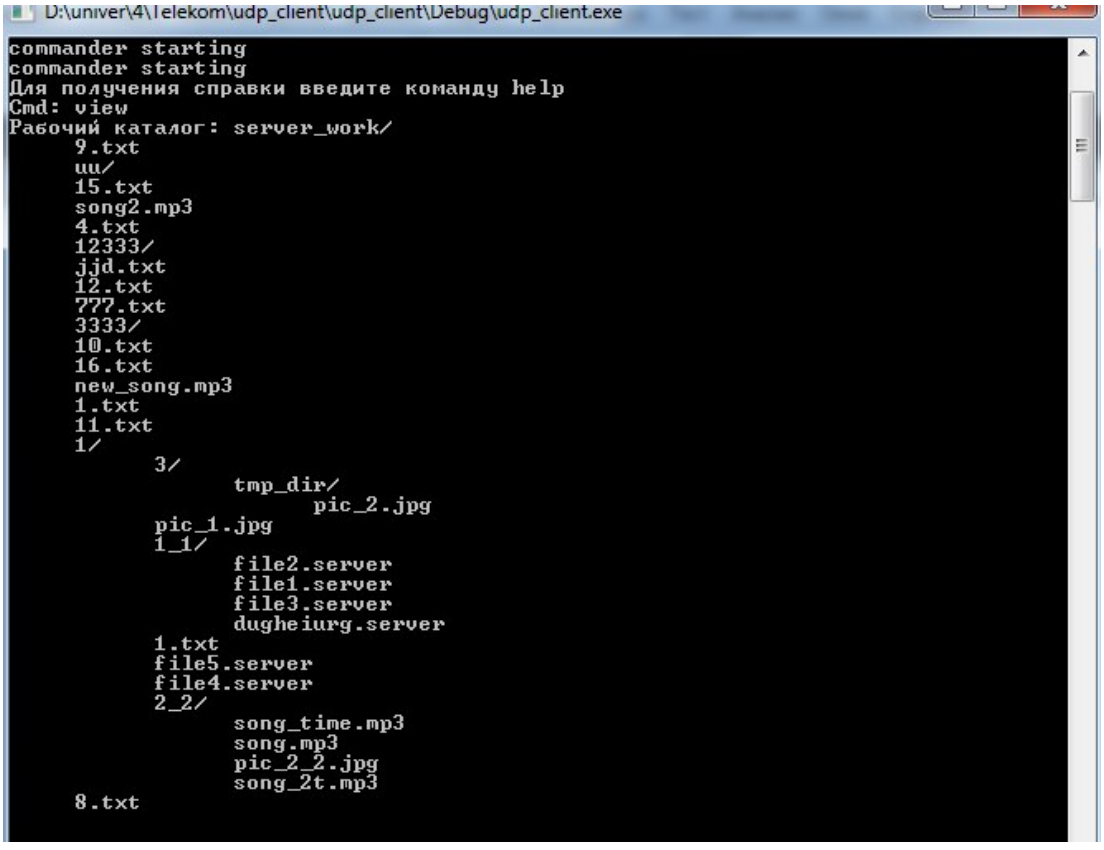

Рис. 3.3: Результаты при введении потерь

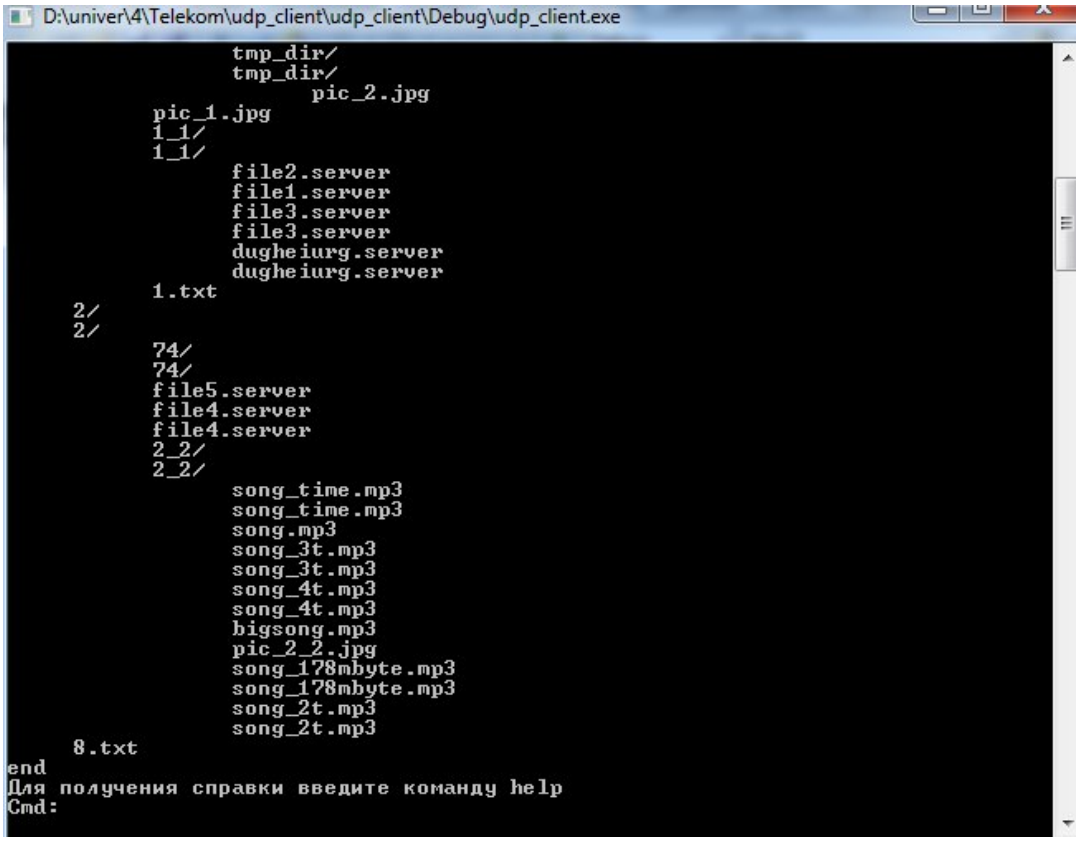

Рис. 3.4: Результаты при введении дупликации

ющиеся данные.

Глава 4

Тестирование с помощью утилиты valgrind

Тестирование производилось командой valgrind –leak-check=full ./tcp_server

```
Результаты:
==1441== Memcheck, a memory error detector
==1441== Copyright (C) 2002-2013, and GNU GPLd, by Julian Seward et al.
==1441== Using Valgrind-3.10.0 and LibVEX; rerun with -h for copyright info
==1441== Command: ./tcp_server
==1441==
Создание сокета
bind
listen
Waiting connections
Connection accepted
==1441== Invalid write of size 4
==1441==    at 0x804A182: main (main.c:63)
==1441==  Address 0x436e028 is 0 bytes inside a block of size 1 alloced
==1441==    at 0x40291CC: malloc (vg_replace_malloc.c:296)
==1441==    by 0x804A175: main (main.c:62)
==1441==
Handler assigned
==1441== Thread 2:
==1441== Invalid read of size 4
==1441==    at 0x8049A2C: connection_handler (connection_handler.h:16)
==1441==    by 0x41ABEFA: start_thread (pthread_create.c:309)
==1441==    by 0x42AA62D: clone (clone.S:129)
==1441==  Address 0x436e028 is 0 bytes inside a block of size
```

```
    1 alloced
==1441==    at 0x40291CC: malloc (vg_replace_malloc.c:296)
==1441==    by 0x804A175: main (main.c:62)
==1441==
^C==1441==
==1441== HEAP SUMMARY:
==1441==     in use at exit: 145 bytes in 2 blocks
==1441==   total heap usage: 15 allocs, 13 frees, 426,493
   bytes allocated
==1441==
==1441== Thread 1:
==1441== 144 bytes in 1 blocks are possibly lost in loss
   record 2 of 2
==1441==    at 0x402B0D5: calloc (vg_replace_malloc.c:623)
==1441==    by 0x4010C8E: allocate_dtv (dl-tls.c:296)
==1441==    by 0x40113EB: _dl_allocate_tls (dl-tls.c:460)
==1441==    by 0x41AC72C: allocate_stack (allocatestack.c
   :589)
==1441==    by 0x41AC72C: pthread_create@@GLIBC_2.1 (
   pthread_create.c:495)
==1441==    by 0x804A19B: main (main.c:65)
==1441==
==1441== LEAK SUMMARY:
==1441==    definitely lost: 0 bytes in 0 blocks
==1441==    indirectly lost: 0 bytes in 0 blocks
==1441==      possibly lost: 144 bytes in 1 blocks
==1441==    still reachable: 1 bytes in 1 blocks
==1441==         suppressed: 0 bytes in 0 blocks
==1441== Reachable blocks (those to which a pointer was found
   ) are not shown.
==1441== To see them, rerun with: --leak-check=full --show-
   leak-kinds=all
==1441==
==1441== For counts of detected and suppressed errors, rerun
   with: -v
==1441== ERROR SUMMARY: 3 errors from 3 contexts (suppressed:
    0 from 0)
```

В итоге 3 ошибки. Результат после исправления:

```
==1502== Memcheck, a memory error detector
==1502== Copyright (C) 2002-2013, and GNU GPLd, by Julian
   Seward et al.
==1502== Using Valgrind-3.10.0 and LibVEX; rerun with -h for
   copyright info
==1502== Command: ./tcp_server
==1502==
Создание сокета
bind
listen
```

```
Waiting connections
Connection accepted
Handler assigned
^C==1502==
==1502== HEAP SUMMARY:
==1502==     in use at exit: 148 bytes in 2 blocks\\
==1502==   total heap usage: 15 allocs, 13 frees, 426,496
   bytes allocated\\
==1502==
==1502== 144 bytes in 1 blocks are possibly lost in loss
   record 2 of 2\\
==1502==    at 0x402B0D5: calloc (vg_replace_malloc.c:623)\\
==1502==    by 0x4010C8E: allocate_dtv (dl-tls.c:296)\\
==1502==    by 0x40113EB: _dl_allocate_tls (dl-tls.c:460)\\
==1502==    by 0x41AC72C: allocate_stack (allocatestack.c
   :589)\\
==1502==    by 0x41AC72C: pthread_create@@GLIBC_2.1 (
   pthread_create.c:495)
==1502==    by 0x804A19B: main (main.c:65)
==1502==
==1502== LEAK SUMMARY:
==1502==    definitely lost: 0 bytes in 0 blocks
==1502==    indirectly lost: 0 bytes in 0 blocks
==1502==      possibly lost: 144 bytes in 1 blocks
==1502==    still reachable: 4 bytes in 1 blocks
==1502==         suppressed: 0 bytes in 0 blocks
==1502== Reachable blocks (those to which a pointer was found
   ) are not shown.
==1502== To see them, rerun with: --leak-check=full --show-
   leak-kinds=all
==1502==
==1502== For counts of detected and suppressed errors, rerun
   with: -v
==1502== ERROR SUMMARY: 1 errors from 1 contexts (suppressed:
    0 from 0)
```

Осталась 1 ошибка не влияющая на работу приложения.

Глава 5

Выводы

В результате работы были созданы клиентское и серверное приложения для собственного протокола взаимодействия. Было создано две реализации приложения для протоколов TCP и UDP. Клиентское и серверное приложения были реализованы для двух разных платформ: ОС Windows и Linux. При реализации использова- лись стандартные сокеты. Реализации сокетов для использованных ОС идентичны, портирование программ с одной платформы на другую вы- полняется достаточно просто. В результате работы была создана клиент-серверная система загрузки и приема файлов с сервера. Файлы храянятся в общей папке, клиенты могут создавать, удалять папки, пермещаться по каталогу.

5.1 Реализация для TCP

Протокол TCP удобен для реализации пользовательских приложений, так как обеспечивает установление соединения и надежную доставку пакетов. Протокол обечпечивает стабильное надежное соединение, поэтому при реализации своего протокола не требуется волноваться об этом. Однако, эти дополнительные средства синхронизации требуют больше времени на доставку, т.е. скорость передачи данных ниже чем в UDP.

5.2 Реализация для UDP

Протокол UDP удобен для реализации приложений, не требующих точной доставки пакетов. Он позволяет передавать данные с большей скоростью, однако вероятность потери пакета при этом выше, чем в TCP. Поэтому, использовать данный протокол для реализации поставленной задачи не очень удобно. Требуется использовать дополнительные инстру-

менты для подтверждения корректной доставки, т.е. каким-то образом «симулировать» TCP. Это неудобно и неэффективно.

Приложения

Описание среды разработки

Серерное приложение реализовано на ОС Linux debian 3.16.0 (среда разработки QT Creator 3.5.0), клиентское на ОС Windows 7 (среда разработки Visual Stidio 2010). Срервер запускался на виртуальной машине, соединение через сетвеой мост.

Листинги

TCP SERVER

Основной файл программы main.c

```
1  #include <stdio.h>
2  #include <stdbool.h>
3  #include <sys/socket.h>
4  #include <resolv.h>
5  #include <arpa/inet.h>
6  #include <errno.h>
7  #include <string.h>
8  #include <unistd.h> // для ch_dir
9  #include<pthread.h>
10 #include "connection_handler.h"
11
12 #define PORT        1234
13
14 int main(){
15     int sockfd;
16     struct sockaddr_in addr, client;
17     char dir[MAXBUF] = "/home/user/server/server_work";
18     // инициализация рабочей директории
19     chdir(dir);
20
21     //Создание сокета TCP
```

```
22      //AF_INET - стек протоколов TCP/IP, SOXK_STREAM - создани
            е надежного сокета
23      //0 - протокол определяется типом сокета
24
25      printf("Создание сокета\n");
26      if ( (sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0 )
27      {
28          perror("Ошибка создания сокета");
29          exit(errno);
30      }
31
32      //Задание адреса и порта серверу
33      bzero(&addr, sizeof(addr));
34      addr.sin_family = AF_INET;
35      addr.sin_port = htons(PORT);
36      addr.sin_addr.s_addr = INADDR_ANY;
37
38      //Связь сокета с портом
39      printf("bind\n");
40      if ( bind(sockfd, (struct sockaddr*)&addr, sizeof(addr))
            != 0 ){
41          perror("ошибка связи сокета");
42          exit(errno);
43      }
44
45
46      //включение прослушивания
47      // дескриптор сокета, максимальное число очереди
48      printf("listen\n");
49      if ( listen(sockfd, 1) != 0 ) {
50          perror("ошибка включения прослушивания");
51          exit(errno);
52      }
53
54      puts("Waiting connections");
55      int    c = sizeof(struct sockaddr_in);
56      int client_sock, *new_sock;
57      while( (client_sock = accept(sockfd, (struct sockaddr *)&
            client, (socklen_t*)&c)) )
58      {
59          puts("Connection accepted");
60
61          pthread_t sniffer_thread;
62          new_sock = malloc(4);
63          *new_sock = client_sock;
64
65          if( pthread_create( &sniffer_thread , NULL ,
                connection_handler , (void*) new_sock) < 0)
66          {
```

```
67              perror("could not create thread");
68              return 1;
69          }
70          puts("Handler assigned");
71      }
72
73      if (client_sock < 0)
74      {
75          perror("accept failed");
76          return 1;
77      }
78      return 0;
79  }
```

headers

chdir.h

```
1     #ifndef CH_DIR
2  #define CH_DIR
3  #include <unistd.h>
4  #include <string.h>
5
6  int ch_dir(char *buffer, int size, int sock){
7      bzero(buffer, size);
8      recv(sock, buffer, size, 0);
9      int i;
10     char dir[256] = "/home/user/server/server_work";
11     strcat(dir, buffer);
12     i = chdir(dir);
13     if (i == -1){
14         bzero(buffer, size);
15         strcpy(buffer, "Недействительный путь!");
16         send(sock, buffer, size, 0);
17
18         bzero(buffer, size);
19         getcwd(buffer, size);
20         send(sock, buffer, size, 0);
21     }
22     else{
23         bzero(buffer, size);
24         strcpy(buffer, "succes");
25         send(sock, buffer, size, 0);
26
27         bzero(buffer, size);
28         getcwd(buffer, size);
29         send(sock, buffer, size, 0);
30     }
```

```
31      printf("%s\n", dir);
32      return 0;
33  }
34
35  #endif // CH_DIR
```

connectionhandler.h

```
1   #ifndef CONNECTION_HANDLER
2   #define CONNECTION_HANDLER
3   #include<pthread.h>
4   #include "view.h"
5   #include "get_dir.h"
6   #include "view_dir.h"
7   #include "ch_dir.h"
8   #include "download.h"
9   #include "mk_dir.h"
10  #include "rm_dir.h"
11  #include "upload.h"
12
13  void *connection_handler(void *socket_desc)
14  {
15      //socket
16      int sock = *(int*)socket_desc;
17      char buffer[MAXBUF];
18      char dir[MAXBUF] = "/home/user/server/server_work"; //
            for "view"
19
20      bzero(buffer, MAXBUF);
21      strcpy(buffer, "Connected (server message)\n");
22      send(sock, buffer, MAXBUF, 0);
23
24      while(1){
25           bzero(buffer, MAXBUF);
26           recv(sock, buffer, MAXBUF, 0);
27
28          // Проверка команд
29           if (strcmp(buffer, "view\r\n") == 0){
30               // send(clientfd, "WORK DIRECTORY/\n", 16, 0);
31              view(dir, 5, sock);
32
33              bzero( buffer, MAXBUF);
34              sprintf( buffer, "%d", 0);
35              strcat(buffer, "|");
36              strcat(buffer, "d");
37               strcat(buffer, "|");
38              strcat(buffer, "end");
39              send(sock, buffer, MAXBUF, 0);
40           }
41          if (strcmp(buffer, "view_dir\r\n") == 0){
```

```
42              chdir("/home/user/server/server_work");
43              view_dir(buffer, MAXBUF, sock);
44
45              bzero( buffer, MAXBUF);
46              sprintf( buffer, "%d", 0);
47              strcat(buffer, "|");
48              strcat(buffer, "d");
49              strcat(buffer, "|");
50               strcat(buffer, "end");
51              send(sock, buffer, MAXBUF, 0);
52              chdir("/home/user/server/server_work");
53          }
54          else if (strcmp(buffer,"disconnect")==0){
55              //завершение соединения
56              close(sock);
57              break;
58          }
59          else if (strcmp(buffer,"get_dir\r\n")==0){
60              get_dir(buffer, MAXBUF, sock);
61          }
62          else if (strcmp(buffer,"ch_dir\r\n")==0){
63              ch_dir(buffer, MAXBUF, sock);
64          }
65          else if (strcmp(buffer,"upload\r\n")==0){
66              download(buffer, MAXBUF, sock);
67          }
68          else if (strcmp(buffer,"download\r\n")==0){
69              upload(buffer, MAXBUF, sock);
70          }
71          else if (strcmp(buffer,"mk_dir\r\n")==0){
72              mk_dir(buffer, MAXBUF, sock, S_IRWXU);
73          }
74          else if (strcmp(buffer,"rm_dir\r\n")==0){
75              rm_dir(buffer, MAXBUF, sock);
76          }
77          else{
78                  printf("%s", buffer);
79                  send(sock, "void␣\n", 5, 0);
80           }
81      }
82
83      //Free the socket pointer
84      free(socket_desc);
85
86      return 0;
87  }
88  #endif // THREAD_HANDLER
```

download.h

```c
#ifndef DOWNLOAD
#define DOWNLOAD
#include <string.h>
#include <stdio.h>
#include <unistd.h>

int download(char *buffer, size_t size, int sock){
    int r;
    int p;
    int i = 0;
    int j = 0;
    int k = 0;
    long int fileSize = 0;
    char send_buf[size], name[size];

    //прием и формирование размера файла
    bzero(buffer, size);
    recv(sock, buffer, size, 0);
    while(buffer[i] != '\0'){
          i++;
          k++;
     }
     for(j=0; j<k; j++){
         p = pow(10,i-1);
         fileSize+=p*((int)buffer[j]-48);
         i--;
     }
    printf("SIze:%d\n", fileSize);


    // прием имени файла
    bzero(buffer, size);
    recv(sock, buffer, size, 0);
    // формирование пути
    bzero(name, size);
    getcwd(name, size);
    strcat(name, "/");
    strcat(name, buffer);
    printf("%s\n", name);

    //формирование пути с именем файла

   FILE* f;
   f = fopen(name, "wb");
   if(f == NULL){
       printf("File dont open\n");
       return -1;
   }
```

```
49
50      while(fileSize > 0){
51          if(fileSize >=size){
52              bzero(buffer, size);
53              r = recv(sock, buffer, size, 0);
54              fwrite(buffer, 1, size, f); // записываем весь буф
                    ер
55
56              // отправка клиенту
57              bzero(send_buf, size);
58              sprintf(send_buf, "%d",r );
59              send(sock, send_buf, size, 0);
60          }
61          else{
62              bzero(buffer, size);
63              r = recv(sock, buffer, size, 0);
64              fwrite(buffer, 1, fileSize, f); //записываем тольк
                    о сколько надо
65
66              // отправка клиенту
67              bzero(send_buf, size);
68              sprintf(send_buf, "%d",r );
69              send(sock, send_buf, size, 0);
70          }
71          fileSize -=size;
72      }
73
74  /*while(fileSize > 0){
75              if(fileSize >=fbuf_size){
76                  bzero(f_buf, fbuf_size);
77                  r = recv(sock, f_buf, fbuf_size, 0);
78                  fwrite(f_buf, 1, size, f); // записываем весь б
                        уфер
79
80                  // отправка клиенту
81                  bzero(send_buf, size);
82                  sprintf(send_buf, "%d",r );
83                  send(sock, send_buf, size, 0);
84              }
85              else{
86                  bzero(f_buf, fbuf_size);
87                  r = recv(sock, f_buf, fbuf_size, 0);
88                  fwrite(f_buf, 1, fileSize, f); //записываем тол
                        ько сколько надо
89
90                  // отправка клиенту
91                  bzero(send_buf, size);
92                  sprintf(send_buf, "%d",r );
93                  send(sock, send_buf, size, 0);
```

```
94              }
95              fileSize -=fbuf_size;
96              printf("now fileSize: %d\n", fileSize);
97              printf("now r: %d\n", r);
98              sleep(2);
99          }*/
100     printf("succes!\n");
101     fclose(f);
102     //free(f_buf);
103     return 0;
104 }
105
106 #endif // DOWNLOAD
```

getdir.h

```
1  #ifndef GET_DIR
2  #define GET_DIR
3  #include <unistd.h>
4
5  int get_dir(char *buffer, size_t size, int clientfd){
6      bzero( buffer, size);
7      getcwd(buffer, size);
8      // отправка клиенту
9      send(clientfd, buffer, size, 0);
10     return 0;
11 }
12
13 #endif // GET_DIR
```

mkdir.h

```
1  #ifndef MK_DIR
2  #define MK_DIR
3  #include <sys/types.h>
4  #include <sys/stat.h>
5  #include <string.h>
6
7  int mk_dir(char* buffer, int size, int sock, mode_t mode){
8      int s;
9      bzero(buffer, size);
10     recv(sock, buffer, size, 0);
11     s = mkdir(buffer, mode);
12     if(s == 0){
13         bzero(buffer, size);
14         strcpy(buffer,"Directory␣created\n");
15         send(sock, buffer, size, 0);
16     }
17     else{
18         bzero(buffer, size);
```

```
19          strcpy(buffer,"Directory␣don't␣created\n");
20          send(sock, buffer, size, 0);
21      }
22
23      return 0;
24  }
25
26  #endif // MK_DIR
```

rmdir.h

```
1   #ifndef RM_DIR
2   #define RM_DIR
3   #include <unistd.h>
4
5   int rm_dir(char* buffer, int size, int sock){
6       int s;
7       bzero(buffer, size);
8       recv(sock, buffer, size, 0);
9       s = rmdir(buffer);
10
11      if(s == 0){
12          bzero(buffer, size);
13          strcpy(buffer,"Directory␣removed\n");
14          send(sock, buffer, size, 0);
15      }
16      else{
17          bzero(buffer, size);
18          strcpy(buffer,"Directory␣don't␣removed\n");
19          send(sock, buffer, size, 0);
20      }
21
22      return 0;
23  }
24  #endif // RM_DIR
```

upload.h

```
1   #ifndef UPLOAD
2   #define UPLOAD
3   #include <string.h>
4   #include <stdio.h>
5   #include <unistd.h>
6
7   int upload(char* buffer, int size, int sock){
8       char name[2048], trace[2048];
9       int fileSize;
10      int message_length;
11      FILE* f;
12
```

```
13      //прием имени файла
14      bzero(name, size);
15      recv(sock, name, size, 0);
16
17      //формирование пути открытия файла
18      bzero(trace, size);
19      getcwd(trace, size);
20      strcat(trace, "/");
21      strcat(trace, name);
22
23      f = fopen(name, "rb");
24      if(f == NULL){
25          printf("File dont open\n");
26          bzero(buffer, size);
27          strcpy(buffer, "false");
28          send(sock, buffer, size, 0);
29          sleep(0.01);
30          return -1;
31      }
32      else{
33          bzero(buffer, size);
34          strcpy(buffer, "true");
35          send(sock, buffer, size, 0);
36          sleep(0.01);
37      }
38
39
40      //отправка размера
41      fseek(f, 0, SEEK_END);
42      fileSize = ftell(f);
43      fseek(f, 0, SEEK_SET);
44      printf("Size of file: %d\n", fileSize);
45      bzero(buffer, size);
46      //strcpy(buffer, (char)fileSize);
47      sprintf(buffer, "%d", fileSize);
48      send(sock, buffer, size, 0);
49      sleep(0.01);
50
51      while(fileSize > 0){
52          bzero(buffer, size);
53          if(fileSize >=1024){
54              message_length = fread(buffer, size, 1, f); // ко
                    личество
55
56              if(message_length != 0)
57                      send(sock, buffer, size, 0);
58                      sleep(0.01);
59          }
60          else{
```

```
61              message_length = fread(buffer, fileSize, 1, f);
62
63              if(message_length != 0)
64                      send(sock, buffer, fileSize, 0);
65                      sleep(0.01);
66          }
67          fileSize -= size;
68      }
69      fclose(f);
70
71      return 0;
72  }
73
74  #endif // UPLOAD
```

view.h

```
1  #ifndef VIEW_H
2  #define VIEW_H
3
4  #include <string.h>
5  #include <unistd.h>
6  #include <dirent.h>
7  #include <sys/stat.h>
8  #include <stdlib.h>
9
10 #define MAXBUF 1024
11 void view(char *dir, int depth, int clientfd) {
12     //chdir("/home/user/server/server_work");
13     // буфер для послыки
14     char buffer[MAXBUF];
15     DIR *dp;
16     struct dirent *entry;
17      struct stat statbuf;
18      if ((dp = opendir(dir)) == NULL) {
19              fprintf(stderr, "cannot open directory: %s\n",
                    dir);
20              return;
21     }
22
23     chdir(dir);
24     while((entry = readdir(dp)) != NULL) {
25          lstat(entry->d_name, &statbuf);
26          if (S_ISDIR(statbuf.st_mode)) {
27              //Находит каталог, но игнорирует . и ..
28                 if (strcmp(".", entry->d_name) == 0 || strcmp
                       ("..", entry->d_name) == 0)
29                         continue;
30
31                  bzero( buffer, MAXBUF);
```

```
32                  sprintf( buffer, "%d", depth);
33                  strcat(buffer, "|");
34                  strcat(buffer, "d");
35                  strcat(buffer, "|");
36                  strcat(buffer, entry->d_name);
37                  send(clientfd, buffer, MAXBUF, 0);
38
39                 int new_depth = depth + 6;
40                  // Рекурсиеный вызое с новый отступом
41                  view(entry->d_name, new_depth, clientfd);
42          }
43           else {
44              bzero( buffer, MAXBUF);
45              sprintf( buffer, "%d", depth);
46              strcat(buffer, "|");
47              strcat(buffer, "f");
48              strcat(buffer, "|");
49              strcat(buffer, entry->d_name);
50              send(clientfd, buffer, MAXBUF, 0);
51           }
52   }
53  chdir("..");
54  closedir(dp);
55  //chdir("/home/user/server/server_work");
56 }
57 #endif // VIEW_H
```

viewdir.h

```
1  #ifndef VIEW_DIR
2  #define VIEW_DIR
3  #include "view.h"
4
5  int view_dir(char *buffer, int buf_size, int clientfd){
6      bzero(buffer, buf_size);
7      getcwd(buffer, buf_size);
8      view(buffer, 5, clientfd);
9      return 0;
10 }
11 #endif // VIEW_DIR
```

TCP client

headers

chdir.h

```c
#ifndef CHDIR
#define CHDIR
#include <string.h>

int ch_dir(char *buffer, int sock, int size){
   memset(buffer, 0, size);
    strcpy(buffer, "ch_dir\r\n");
    send(sock, buffer, size, 0);

    printf("путь: ");
    memset(buffer, 0, size);
    scanf("%s", buffer);
    send(sock, buffer, size, 0);

    //прием ошибки либо сообщения об успехе
    memset(buffer, 0, size);
    recv(sock, buffer, size, 0);
    if(strcmp(buffer, "succes") != 0)
       printf("Неверный каталог!");

    memset(buffer, 0, size);
    recv(sock, buffer, size, 0);
    printf("Текущий каталог: %s\n", buffer);
    return 0;
}

#endif // CHDIR
```

download.h

```c
#ifndef DOWNLOAD
#define DOWNLOAD
#include <string.h>
#include <stdio.h>
#include <math.h>
#include <time.h>
#define MAXBUF  1024

int download(char *buffer, size_t size, int sock){
    int r;
    int p;
    int i = 0;
    int j = 0;
    int k = 0;
    int fileSize = 0;
   time_t start_time, finish_time;
    char send_buf[MAXBUF], name[MAXBUF], trace[MAXBUF];

   //отправка команды
```

```
20      memset(buffer, 0, size);
21       strcpy(buffer, "download\r\n");
22       send(sock, buffer, size, 0);
23
24      // отправка на сервер имени файла
25       printf("Имя на сервере: ");
26       memset(name, 0, size);
27       scanf("%s",name);
28       send(sock, name, size, 0);
29
30      //ожидание подтверждения открытия файла
31      memset(buffer, 0, size);
32      recv(sock, buffer, size, 0);
33      if(strcmp(buffer, "false") == 0){
34         printf("Incorrect name - file dont open!\n");
35         return -1;
36      }
37
38      // где сохранять
39      printf("Путь (куда сохранить): ");
40       memset(trace, 0, size);
41       scanf("%s",trace);
42      printf("\n");
43      //формирование пути с именем файла
44      strcat(trace, "\\");
45      strcat(trace, name);
46
47      FILE* f;
48      f = fopen(trace, "wb");
49      if(f == NULL){
50          printf("File dont create\n");
51          return -1;
52      }
53
54       //прием и формирование размера файла
55       memset(buffer, 0, size);
56       recv(sock, buffer, size, 0);
57       while(buffer[i] != '\0'){
58             i++;
59             k++;
60        }
61        for(j=0; j<k; j++){
62            p = pow(10.0,i-1);
63            fileSize+=p*((int)buffer[j]-48);
64            i--;
65        }
66       //printf("size string: %s\n", buffer);
67       printf("SIze:%d\n", fileSize);
68
```

```
69
70      // запомнить время начала
71      start_time = time(NULL);
72      while(fileSize > 0){
73          if(fileSize >=1024){
74              memset(buffer, 0, size);
75              r = recv(sock, buffer, size, 0);
76              fwrite(buffer, 1, size, f); // записываем весь буф
                    ер
77
78              // печать
79              memset(send_buf, 0, size);
80              printf("Принято␣%d\n",r );
81          }
82          else{
83              memset(buffer,0, size);
84              r = recv(sock, buffer, size, 0);
85              fwrite(buffer, 1, fileSize, f); //записываем тольк
                    о сколько надо
86
87              // отправка клиенту
88              memset(send_buf,0, size);
89              printf("Принято␣%d\n",r );
90          }
91          fileSize -=1024;
92      }
93      // запомнить время завершения передачи
94      finish_time = time(NULL);
95      fclose(f);
96      printf("Время␣передачи:␣%f\n", difftime(finish_time,
            start_time));
97      return 0;
98  }
99
100 #endif // DOWNLOAD
```

getdir.h

```
1  #ifndef GET_DIR
2  #define GET_DIR
3  #include <string.h>
4
5  int get_dir(char *buffer, int sock, int size){
6       memset(buffer, 0, size);
7       strcpy(buffer, "get_dir\r\n");
8       send(sock, buffer, size, 0);
9
10      // прием
11      memset(buffer, 0, size);
12      recv(sock, buffer, size,0);
```

```
13      printf("Текущий каталог: %s\n", buffer);
14
15      return 0;
16  }
17  #endif // GET_DIR
```

mkdir.h

```
1   #ifndef MK_DIR
2   #define MK_DIR
3   #include <string.h>
4
5   int mk_dir(char* buffer, int size, int sock){
6       memset(buffer, 0, size);
7       strcpy(buffer, "mk_dir\r\n");
8       send(sock, buffer, size, 0);
9
10      memset(buffer, 0, size);
11      printf("Имя папки: ");
12      scanf("%s", buffer);
13      send(sock, buffer, size, 0);
14
15      memset(buffer, 0, size);
16      recv(sock, buffer, size, 0);
17      printf("%s", buffer);
18      return 0;
19  }
20
21  #endif // MK_DIR
```

rmdir.h

```
1   #ifndef RM_DIR
2   #define RM_DIR
3   #include <string.h>
4
5   int rm_dir(char* buffer, int size, int sock){
6       memset(buffer, 0, size);
7       strcpy(buffer, "rm_dir\r\n");
8       send(sock, buffer, size, 0);
9
10      memset(buffer, 0, size);
11      printf("Имя папки: ");
12      scanf("%s", buffer);
13      printf("\n");
14      send(sock, buffer, size, 0);
15
16      memset(buffer, 0, size);
17      recv(sock, buffer, size, 0);
18      printf("%s", buffer);
```

```
19
20      return 0;
21  }
22
23  #endif // RM_DIR
```

upload.h

```
1   #ifndef UPLOAD
2   #define UPLOAD
3   #include <string.h>
4   #include <stdio.h>
5   #include <time.h>
6   #include <stdlib.h>
7
8   int upload(char* buffer, int size, int sock){
9       char name[1024];
10     //int size = ssize;
11     //char *f_buf;//[1024];
12      long int fileSize;
13      long int message_length;
14
15     time_t start_time, finish_time;
16      FILE* f;
17
18     //f_buf = (char*) malloc(fbuf_size);
19
20      printf("Путь: ");
21      memset(name, 0, 1024);
22      scanf("%s", name);
23
24      f = fopen(name, "rb");
25      if(f == NULL){
26          printf("File dont open\n");
27          return -1;
28      }
29
30      memset(buffer, 0, size);
31      strcpy(buffer, "upload\r\n");
32      send(sock, buffer, size, 0);
33     //Sleep(1000);
34
35
36      //отправка размера
37      fseek(f, 0, SEEK_END);
38      fileSize = ftell(f);
39      fseek(f, 0, SEEK_SET);
40      printf("Size of file: %d\n", fileSize);
41      memset(buffer, 0, size);
42      //strcpy(buffer, (char)fileSize);
```

```
43      sprintf(buffer, "%d", fileSize);
44      send(sock, buffer, size, 0);
45     //Sleep(1000);
46
47
48     // отправка на сервер имени файла
49      printf("Имя на сервере: ");
50      memset(buffer, 0, size);
51      scanf("%s",buffer);
52      send(sock, buffer, size, 0);
53     //Sleep(1000);
54
55     // запомнить время начала
56     start_time = time(NULL);
57
58     while(fileSize > 0){
59          memset(buffer, 0, size);
60          if(fileSize >= size){
61              message_length = fread(buffer, size, 1, f); // ко
                    личество
62
63              if(message_length != 0){
64                      send(sock, buffer, size, 0);
65                 Sleep(1);
66           }
67              memset(buffer, 0, size);
68              recv(sock, buffer, size, 0);
69              printf("Передача - было принято сервером: %s\n",
                    buffer);
70          }
71          else{
72              message_length = fread(buffer, fileSize, 1, f);
73
74              if(message_length != 0){
75                      send(sock, buffer, fileSize, 0);
76                 Sleep(1);
77           }
78              memset(buffer, 0, size);
79              recv(sock, buffer, size, 0);
80              printf("Передача - было принято сервером: %s\n",
                    buffer);
81          }
82          fileSize -= size;
83      }
84
85
86     finish_time = time(NULL);
87      fclose(f);
```

```
88      printf("Время передачи: %f\n", difftime(finish_time,
            start_time));
89
90      return 0;
91  }
92
93  #endif // UPLOAD
```

view.h

```
1   #ifndef VIEW_H
2   #define VIEW_H
3   #include <string.h>
4   #include <math.h>
5
6   int view(char* buffer, int sock, int buf_size){
7       printf("Рабочий каталог: server_work/\n");
8       memset(buffer, 0, buf_size);
9       strcpy(buffer, "view\r\n");
10      send(sock, buffer, buf_size, 0);
11
12      while(1){
13          int depth;
14          int i;
15          int j;
16          int k;
17          memset(buffer, 0, buf_size);
18          recv(sock, buffer, buf_size,0);
19          if(strcmp(buffer, "0|d|end")==0){
20              printf("end\n");
21              break;
22          }
23          else{
24              i = 0;
25              j = 0;
26              k = 0;
27              int p=0;
28             depth = 0;
29             // подсчет количества знаков числа
30             while(buffer[i] != '|'){
31                    i++;
32                    k++;
33              }
34              for(j=0; j<k; j++){
35                  p = pow(10.0,i-1);
36                  depth+=p*((int)buffer[j]-48);
37                  i--;
38              }
39              //
                ------------------------------------------------
```

```
40              k++; // знак '|'
41              if(buffer[k] == 'f')
42                  p = 0;
43              else
44                  p = 1;
45              //
                    ----------------------------------------------------
46              k++; // знак '|'
47              k++;
48              // печать пробелов
49              for(i=0; i<depth; i++)
50                  printf("␣");
51              // печать названия
52              while(buffer[k] != '\0'){
53                  printf("%c", buffer[k]);
54                  k++;
55              }
56              // печать символа папки и перевод строки
57              if(p == 1)
58                  printf("/\n");
59              else
60                  printf("\n");
61          }
62  }
63      return 0;
64 }
65
66 #endif // VIEW_H
```

viewdir.h

```
1  #ifndef view_dir_DIR_H
2  #define view_dir_DIR_H
3  #include <string.h>
4  #include <math.h>
5
6  int view_dir(char* buffer, int sock, int size){
7      memset(buffer, 0, size);
8      strcpy(buffer, "view_dir\r\n");
9      send(sock, buffer, size, 0);
10     while(1){
11         int depth;
12         int i;
13         int j;
14         int k;
15         memset(buffer, 0, size);
16         recv(sock, buffer, size,0);
17         if(strcmp(buffer, "0|d|end")==0){
```

```
18              printf("end\n");
19              break;
20          }
21          else{
22              i = 0;
23              j = 0;
24              k = 0;
25              int p=0;
26             depth = 0;
27             // подсчет количества знаков числа
28             while(buffer[i] != '|'){
29                    i++;
30                    k++;
31              }
32              for(j=0; j<k; j++){
33                  p = pow(10.0,i-1);
34                  depth+=p*((int)buffer[j]-48);
35                  i--;
36              }
37              //
                    ----------------------------------------------------
38              k++; // знак '|'
39              if(buffer[k] == 'f')
40                  p = 0;
41              else
42                  p = 1;
43              //
                    ----------------------------------------------------
44              k++; // знак '|'
45              k++;
46              // печать пробелов
47              for(i=0; i<depth; i++)
48                  printf("␣");
49              // печать названия
50              while(buffer[k] != '\0'){
51                  printf("%c", buffer[k]);
52                  k++;
53              }
54              // печать символа папки и перевод строки
55              if(p == 1)
56                  printf("/\n");
57              else
58                  printf("\n");
59          }
60  }
61      return 0;
62  }
```

```
63
64 #endif // view_dir_dir_DIR
```

help.h

```
1  #ifndef HELP
2  #define HELP
3  #include <string.h>
4
5  int get_help(){
6      printf("ch_dir - сменить текущую директорию на сервере\n"
           );
7      printf("mk_dir - создать папку в текущей директории\n");
8      //printf("download <имя файла> - загрузить файл с сервера
           \n");
9      printf("upload - загрузить файл на сервер из рабочей дире
           ктории\n");
10     printf("view - просмотр дерева каталогов сервера\n");
11     //printf("connect <IP> <номер порта> - подключиться к сер
           веру \n");
12     printf("disconnect - отключиться от сервера\n");
13     return 0;
14 }
15
16 #endif // HELP
```

main.c

```
1  #define WIN32_LEAN_AND_MEAN
2  #include <stdio.h>
3  #include <conio.h>
4  #include <string.h>
5  #include <winsock2.h>
6  #include <windows.h>
7  #include <locale.h>
8  #include <Ws2tcpip.h>
9  #pragma comment(lib, "Ws2_32.lib")
10
11 #include "view.h"
12 #include "help.h"
13 #include "get_dir.h"
14 #include "view_dir.h"
15 #include "chdir.h"
16 #include "upload.h"
17 #include "rm_dir.h"
18 #include "mk_dir.h"
19 #include "echo.h"
20 #include "download.h"
```

```
21
22 #define PORT 1234
23 //#define SERVERADDR "192.168.1.43"
24 //#define SERVERADDR "10.1.99.25"
25 #define MAXBUF  1024
26
27   int main()
28   {
29    char* SERVERADDR;
30    SERVERADDR = (char*)malloc(15);
31     setlocale(LC_ALL, "Russian");
32    char buffer[MAXBUF];
33
34    memset(buffer, 0, MAXBUF);
35    while(1){
36       printf("Cmd: ");
37       scanf("%s", buffer);
38       if(strcmp("connect", buffer) == 0){
39          printf("\nip: ");
40          scanf("%s", SERVERADDR);
41          break;
42       }
43       else{
44          printf("Waiting to connect command!\n");
45          continue;
46       }
47    }
48
49     //инициализация библиотеки Winsock
50     if (WSAStartup(0x202,(WSADATA *)&buffer[0]))
51     {
52       printf("WSAStart error %d\n",WSAGetLastError());
53      _getch();
54       return -1;
55     }
56
57     //создание сокета
58     SOCKET sock;
59     sock=socket(AF_INET,SOCK_STREAM,0);
60     if (sock < 0)
61     {
62       printf("Socket() error %d\n",WSAGetLastError());
63      _getch();
64       return -1;
65     }
66
67     //установка соединения
68     // заполнение структуры sockaddr_in
69     // указание адреса и порта сервера
```

```
70      sockaddr_in dest_addr;
71      dest_addr.sin_family=AF_INET;
72      dest_addr.sin_port=htons(PORT);
73      HOSTENT *hst;
74
75      // преобразование IP адреса из символьного в
76      // сетевой формат
77      if (inet_addr(SERVERADDR)!=INADDR_NONE)
78        dest_addr.sin_addr.s_addr=inet_addr(SERVERADDR);
79      else
80        // попытка получить IP адрес по доменному
81        // имени сервера
82        if (hst=gethostbyname(SERVERADDR))
83        // hst->h_addr_list содержит не массив адресов,
84        // а массив указателей на адреса
85        ((unsigned long *)&dest_addr.sin_addr)[0]=
86          ((unsigned long **)hst->h_addr_list)[0][0];
87        else
88        {
89          printf("Invalid address %s\n",SERVERADDR);
90          closesocket(sock);
91          WSACleanup();
92        _getch();
93          return -1;
94        }
95    printf("123\n");
96     if (connect(sock,(sockaddr *)&dest_addr,
97                 sizeof(dest_addr)))
98     {
99       printf("Connect error %d\n",WSAGetLastError());
100     _getch();
101      return -1;
102    }
103   //-------------------------------------------------
104   memset(buffer, 0, MAXBUF);
105    recv(sock , buffer , MAXBUF , 0); // прием слова
106    puts(buffer);
107    printf("Для получения справки введите команду help\n");
108    while(1){
109      memset(buffer, 0, MAXBUF);
110        printf("Cmd: ");
111        scanf("%s" , buffer);
112        // команды
113        if(strcmp(buffer, "view") == 0)
114         view_dir(buffer, sock, MAXBUF);
115        else if(strcmp(buffer, "view_dir") == 0)
116            view_dir(buffer, sock, MAXBUF);
117        else if(strcmp(buffer, "help") == 0)
118            get_help();
```

```
119         else if(strcmp(buffer, "get_dir") == 0)
120             get_dir(buffer, sock, MAXBUF);
121         else if(strcmp(buffer, "ch_dir") == 0)
122             ch_dir(buffer, sock, MAXBUF);
123         else if(strcmp(buffer, "upload") == 0)
124             upload(buffer, MAXBUF, sock);
125       else if(strcmp(buffer, "download") == 0)
126             download(buffer, MAXBUF, sock);
127         else if(strcmp(buffer, "mk_dir") == 0)
128             mk_dir(buffer, MAXBUF, sock);
129         else if(strcmp(buffer, "rm_dir") == 0)
130             rm_dir(buffer, MAXBUF, sock);
131       else if(strcmp(buffer, "echo") == 0)
132             echo(buffer, sock, MAXBUF);
133       else if(strcmp(buffer, "disconnect") == 0){
134             send(sock, "disconnect", 10, 0);
135          break;
136       }
137         else
138             printf("unknown comand\n");
139         printf("Для получения справки введите команду help\n"
               );
140     }
141    //------------------------------------------------
142    printf("\nExit\n");
143     closesocket(sock);
144     WSACleanup();
145    _getch();
146     return 0;
147   }
```

UDP SERVER

main.c

```
1 #include<stdio.h>
2 #include<string.h>
3 #include<stdlib.h>
4 #include<arpa/inet.h>
5 #include<sys/socket.h>
6 #include <pthread.h>
7 #include "view.h"
8 #include "get_dir.h"
9 #include "view_dir.h"
10 #include "ch_dir.h"
11 #include "download.h"
12 #include "mk_dir.h"
```

```
13 #include "rm_dir.h"
14 #include "upload.h"
15
16 #define MAXBUF 1024  //Max length of buffer
17 #define PORT 1234   //The port on which to listen for
       incoming data
18 #define MAXCLIENTS 10
19 typedef struct {
20     int sockfd;
21     struct sockaddr_in client;
22     int clilen;
23 } btClient;
24
25 int numClients = 0;
26 btClient clients[MAXCLIENTS];
27 pthread_mutex_t clientsMutex;
28
29 btClient acceptConnection(int sockfd);
30 void* commander(void* cl);
31 void errorHandler(int err, char *errfun);
32 void getNewSocket(int* sockfd, uint16_t* port);
33
34 int main(void)
35 {
36     //
          ------------------------------------------------------------------
37     pthread_attr_t threadAttr;
38     pthread_attr_init(&threadAttr);
39     pthread_attr_setdetachstate(&threadAttr,
          PTHREAD_CREATE_DETACHED);
40
41     //----------init
          ------------------------------------------------------------------
42     struct sockaddr_in si_me, si_other;
43
44     int sock, i, slen = sizeof(si_other);
45      char dir[MAXBUF] = "/home/user/server/server_work";
46      // инициализация рабочей директории
47      chdir(dir);
48     //create a UDP socket
49     if ((sock=socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP)) ==
          -1)
50     {
51         perror("socket");
52         exit(-1);
53     }
54     else
```

```
55          printf("socket created\n");
56
57      // zero out the structure
58      memset((char *) &si_me, 0, sizeof(si_me));
59
60      si_me.sin_family = AF_INET;
61      si_me.sin_port = htons(PORT);
62      si_me.sin_addr.s_addr = htonl(INADDR_ANY);
63
64      //bind socket to port
65      if( bind(sock, (struct sockaddr*)&si_me, sizeof(si_me) )
            == -1)
66      {
67          perror("bind");
68          exit(1);
69      }
70      else
71          printf("bind!\n");
72      //
            ------------------------------------------------------------------------


73
74      while(1) {
75              btClient newClient = acceptConnection(sock);
76              printf("new client accept\n");
77              pthread_t thread;
78
79              if (pthread_create(&thread, NULL, commander, (
                    void*)&newClient) != 0) {
80                  printf("Error while creating new thread\n");
81              }
82              else
83                  printf("New thread created. Client sock = %d\
                        n", newClient.sockfd);
84          }
85      return 0;
86  }
87
88
89  void* commander(void* cl) {
90      printf("commander starting\n");
91      btClient* client = (btClient*) cl;
92
93      printf("set parameters\n");
94      struct sockaddr_in si_other = client->client;
95      int sock = client->sockfd;
96      int slen = client->clilen;
97      sendto(sock, "commander starting\n", sizeof("commander
            starting\n"), 0, (struct sockaddr*) &client->client,
```

```
        client->clilen);
98
99      char buffer[MAXBUF];
100    while(1){
101     bzero(buffer, MAXBUF);
102     int recv_len;
103     if ((recv_len = recvfrom(sock, buffer, MAXBUF, 0, (struct
            sockaddr *) &si_other, &slen)) == -1)
104     {
105         perror("recvfrom()");
106         exit(-1);
107     }
108     else
109      printf("command was received\n");
110
111      char dir[MAXBUF] = "/home/user/server/server_work";
112    // Проверка команд
113     printf("%s\n", buffer);
114     if (strcmp(buffer, "view\r\n") == 0){
115
116        view(dir, 5, sock, si_other, slen);
117
118        bzero( buffer, MAXBUF);
119        sprintf( buffer, "%d", 0);
120        strcat(buffer, "|");
121        strcat(buffer, "d");
122         strcat(buffer, "|");
123        strcat(buffer, "end");
124        //send(sock, buffer, MAXBUF, 0);
125        sendto(sock, buffer, MAXBUF, 0, (struct sockaddr*) &
              si_other, slen);
126         chdir("/home/user/server/server_work");
127     }
128   else if (strcmp(buffer, "view_dir\r\n") == 0){
129        chdir("/home/user/server/server_work");
130        view_dir(buffer, MAXBUF, sock, si_other, slen);
131
132        bzero( buffer, MAXBUF);
133        sprintf( buffer, "%d", 0);
134        strcat(buffer, "|");
135        strcat(buffer, "d");
136        strcat(buffer, "|");
137         strcat(buffer, "end");
138        //send(sock, buffer, MAXBUF, 0);
139         sendto(sock, buffer, MAXBUF, 0, (struct sockaddr*) &
              si_other, slen);
140        chdir("/home/user/server/server_work");
141    }
142    else if (strcmp(buffer,"disconnect")==0){
```

```
143         //завершение соединения
144         close(sock);
145        // break;
146     }
147     else if (strcmp(buffer,"get_dir\r\n")==0){
148         get_dir(buffer, MAXBUF, sock, si_other, slen);
149     }
150     else if (strcmp(buffer,"ch_dir\r\n")==0){
151         ch_dir(buffer, MAXBUF, sock, si_other, slen);
152     }
153     else if (strcmp(buffer,"upload\r\n")==0){
154         download(buffer, MAXBUF, sock, si_other, slen);
155     }
156     else if (strcmp(buffer,"download\r\n")==0){
157         upload(buffer, MAXBUF, sock, si_other, slen);
158     }
159     else if (strcmp(buffer,"mk_dir\r\n")==0){
160         mk_dir(buffer, MAXBUF, sock, S_IRWXU, si_other, slen);
161     }
162     else if (strcmp(buffer,"rm_dir\r\n")==0){
163         rm_dir(buffer, MAXBUF, sock, si_other, slen);
164     }
165     else{
166             printf("%s", buffer);
167             //send(sock, "void \n", 5, 0);
168             if (sendto(sock, "void\n", 5, 0, (struct sockaddr
                    *) &si_other, slen) == -1)
169             {
170                 perror("sendto()");
171                 exit(1);
172             }
173      }
174      }
175      close(client->sockfd);
176      return NULL;
177 }
178
179 btClient acceptConnection(int sockfd) {
180      struct sockaddr_in clientaddr; /* client addr */
181      int clientlen = sizeof(clientaddr); /* byte size of
            client's address */
182      char buf[MAXBUF];
183      int err = 0;
184
185      bzero(buf, MAXBUF);
186      err=recvfrom(sockfd, buf, MAXBUF, 0, (struct sockaddr *)
            &clientaddr, &clientlen);
187      errorHandler(err, "ERROR␣in␣recvfrom");
188
```

```
189     int newsockfd;
190     uint16_t newport;
191     getNewSocket(&newsockfd, &newport);
192     bzero(buf, MAXBUF);
193     sprintf(buf, "%d", newport);
194     err = sendto(sockfd, buf, strlen(buf), 0, (struct
            sockaddr *) &clientaddr, clientlen);
195     errorHandler(err, "ERROR in sendto");
196
197
198     // create new socket for this client
199     struct sockaddr_in newclientaddr; /* client addr */
200     int newclientlen = sizeof(newclientaddr); /* byte size of
            client's address */
201     bzero(buf, MAXBUF);
202     err = recvfrom(newsockfd, buf, MAXBUF, 0, (struct
            sockaddr *) &newclientaddr, &newclientlen);
203     printf("%s\n", buf);
204     errorHandler(err, "ERROR in recvfrom");
205
206     btClient client;
207     client.sockfd = newsockfd;
208     client.client = newclientaddr;
209     client.clilen = newclientlen;
210     return client;
211 }
212
213 int readFrom(btClient* client, char* message, int len) {
214     return recvfrom(client->sockfd,
215                     message,
216                     len,
217                     0,
218                     (struct sockaddr *) &client->client,
219                     &client->clilen);
220 }
221
222 void errorHandler(int err, char *errfun) {
223     if (err < 0) {
224         printf("Error in function %s\n", errfun);
225         exit(1);
226     }
227 }
228
229 void getNewSocket(int* sockfd, uint16_t* port) {
230     struct sockaddr_in serveraddr;
231     int err = 0;
232     *sockfd = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
233     *port = PORT + (numClients + 1);
234     errorHandler(*sockfd, "ERROR opening socket");
```

```
235
236     int optval = 1;
237     setsockopt(*sockfd, SOL_SOCKET, SO_REUSEADDR, (const void
            *)&optval , sizeof(int));
238
239     bzero((char *) &serveraddr, sizeof(serveraddr));
240     serveraddr.sin_family = AF_INET;
241     serveraddr.sin_addr.s_addr = htonl(INADDR_ANY);
242     serveraddr.sin_port =htons((unsigned short)*port);
243
244     err = bind(*sockfd, (struct sockaddr *) &serveraddr,
            sizeof(serveraddr));
245     errorHandler(err, "ERROR on binding");
246 }
```

headers

chdir.h

```
1  #ifndef CH_DIR
2  #define CH_DIR
3  #include <unistd.h>
4  #include <string.h>
5
6  int ch_dir(char *buffer, int size, int sock, struct
       sockaddr_in si_other, int slen){
7      bzero(buffer, size);
8      recv(sock, buffer, size, 0);
9      int i;
10     char dir[256] = "/home/user/server/server_work";
11     strcat(dir, buffer);
12     i = chdir(dir);
13     if (i == -1){
14         bzero(buffer, size);
15         strcpy(buffer, "Недействительный путь!");
16         //send(sock, buffer, size, 0);
17         sendto(sock, buffer, MAXBUF, 0, (struct sockaddr*) &
               si_other, slen);
18
19         bzero(buffer, size);
20         getcwd(buffer, size);
21         //send(sock, buffer, size, 0);
22          sendto(sock, buffer, MAXBUF, 0, (struct sockaddr*) &
               si_other, slen);
23     }
24     else{
25         bzero(buffer, size);
26         strcpy(buffer, "succes");
```

```
27          //send(sock, buffer, size, 0);
28           sendto(sock, buffer, MAXBUF, 0, (struct sockaddr*) &
                 si_other, slen);
29
30          bzero(buffer, size);
31          getcwd(buffer, size);
32          //send(sock, buffer, size, 0);
33           sendto(sock, buffer, MAXBUF, 0, (struct sockaddr*) &
                 si_other, slen);
34      }
35      printf("%s\n", dir);
36      return 0;
37  }
38
39  #endif // CH_DIR
```

download.h

```
1  #ifndef DOWNLOAD
2  #define DOWNLOAD
3  #include <string.h>
4  #include <stdio.h>
5  #include <unistd.h>
6
7  int download(char *buffer, size_t size, int sock, struct
       sockaddr_in si_other, int slen){
8      int r;
9      int p;
10     int i = 0;
11     int j = 0;
12     int k = 0;
13     long int fileSize = 0;
14     char send_buf[size], name[size];
15
16     //прием и формирование размера файла
17     bzero(buffer, size);
18     //recv(sock, buffer, size, 0);
19     recvfrom(sock, buffer, size, 0, (struct sockaddr *) &
           si_other, &slen);
20     while(buffer[i] != '\0'){
21           i++;
22           k++;
23      }
24      for(j=0; j<k; j++){
25          p = pow(10,i-1);
26          fileSize+=p*((int)buffer[j]-48);
27          i--;
28      }
29     printf("SIze:%d\n", fileSize);
30
```

```
31
32      // прием имени файла
33      bzero(buffer, size);
34      //recv(sock, buffer, size, 0);
35      recvfrom(sock, buffer, size, 0, (struct sockaddr *) &
            si_other, &slen);
36
37      // формирование пути
38      bzero(name, size);
39      getcwd(name, size);
40      strcat(name, "/");
41      strcat(name, buffer);
42      printf("%s\n", name);
43
44      //формирование пути с именем файла
45
46    FILE* f;
47    f = fopen(name, "wb");
48    if(f == NULL){
49        printf("File dont open\n");
50        return -1;
51    }
52
53     while(fileSize > 0){
54        if(fileSize >=size){
55            bzero(buffer, size);
56           // r = recv(sock, buffer, size, 0);
57            r = recvfrom(sock, buffer, size, 0, (struct
                sockaddr *) &si_other, &slen);
58            fwrite(buffer, 1, size, f); // записываем весь буф
                ер
59
60            // отправка клиенту
61            bzero(send_buf, size);
62            sprintf(send_buf, "%d",r );
63            //send(sock, send_buf, size, 0);
64            sendto(sock, send_buf, size, 0, (struct sockaddr*)
                  &si_other, slen);
65            sleep(1);
66        }
67        else{
68            bzero(buffer, size);
69           // r = recv(sock, buffer, size, 0);
70            r = recvfrom(sock, buffer, MAXBUF, 0, (struct
                sockaddr *) &si_other, &slen);
71            fwrite(buffer, 1, fileSize, f); //записываем тольк
                о сколько надо
72
73            // отправка клиенту
```

```
74              bzero(send_buf, size);
75              sprintf(send_buf, "%d",r );
76              //send(sock, send_buf, size, 0);
77              sendto(sock, send_buf, size, 0, (struct sockaddr*)
                    &si_other, slen);
78              sleep(1);
79          }
80          fileSize -=size;
81      }
82
83      printf("succes!\n");
84      fclose(f);
85      //free(f_buf);
86      return 0;
87  }
88
89  #endif // DOWNLOAD
```

getdir.h

```
1  #ifndef GET_DIR
2  #define GET_DIR
3  #include <unistd.h>
4
5  int get_dir(char *buffer, size_t size, int clientfd, struct
       sockaddr_in si_other, int slen){
6      bzero( buffer, size);
7      getcwd(buffer, size);
8      // отправка клиенту
9      //send(clientfd, buffer, size, 0);
10      sendto(clientfd, buffer, MAXBUF, 0, (struct sockaddr*) &
            si_other, slen);
11     return 0;
12 }
13
14
15 #endif // GET_DIR
```

mkdir.h

```
1  #ifndef MK_DIR
2  #define MK_DIR
3  #include <sys/types.h>
4  #include <sys/stat.h>
5  #include <string.h>
6
7  int mk_dir(char* buffer, int size, int sock, mode_t mode,
       struct sockaddr_in si_other, int slen){
8      int s;
9      bzero(buffer, size);
```

```
10      //recv(sock, buffer, size, 0);
11      recvfrom(sock, buffer, MAXBUF, 0, (struct sockaddr *) &
            si_other, &slen);
12      s = mkdir(buffer, mode);
13      if(s == 0){
14          bzero(buffer, size);
15          strcpy(buffer,"Directory␣created\n");
16          //send(sock, buffer, size, 0);
17          sendto(sock, buffer, MAXBUF, 0, (struct sockaddr*) &
                si_other, slen);
18      }
19      else{
20          bzero(buffer, size);
21          strcpy(buffer,"Directory␣don't␣created\n");
22          //send(sock, buffer, size, 0);
23          sendto(sock, buffer, MAXBUF, 0, (struct sockaddr*) &
                si_other, slen);
24      }
25
26      return 0;
27  }
28
29  #endif // MK_DIR
```

rmdir.h

```
1   #ifndef RM_DIR
2   #define RM_DIR
3   #include <unistd.h>
4
5   int rm_dir(char* buffer, int size, int sock, struct
        sockaddr_in si_other, int slen){
6       int s;
7       bzero(buffer, size);
8       //recv(sock, buffer, size, 0);
9       recvfrom(sock, buffer, MAXBUF, 0, (struct sockaddr *) &
            si_other, &slen);
10      s = rmdir(buffer);
11
12      if(s == 0){
13          bzero(buffer, size);
14          strcpy(buffer,"Directory␣removed\n");
15          //send(sock, buffer, size, 0);
16          sendto(sock, buffer, MAXBUF, 0, (struct sockaddr*) &
                si_other, slen);
17      }
18      else{
19          bzero(buffer, size);
20          strcpy(buffer,"Directory␣don't␣removed\n");
21          //send(sock, buffer, size, 0);
```

```
22          sendto(sock, buffer, MAXBUF, 0, (struct sockaddr*) &
                si_other, slen);
23      }
24
25      return 0;
26  }
27  #endif // RM_DIR
```

upload.h

```
1   #ifndef UPLOAD
2   #define UPLOAD
3   #include <string.h>
4   #include <stdio.h>
5   #include <unistd.h>
6
7   int upload(char* buffer, int size, int sock, struct
        sockaddr_in si_other, int slen){
8       char name[2048], trace[2048];
9       int fileSize;
10      int message_length;
11      FILE* f;
12
13      //прием имени файла
14      bzero(name, size);
15      //recv(sock, name, size, 0);
16      int r = recvfrom(sock, name, size, 0, (struct sockaddr *)
            &si_other, &slen);
17      printf("%d\n", r);
18
19      //формирование пути открытия файла
20      bzero(trace, size);
21      getcwd(trace, size);
22      strcat(trace, "/");
23      strcat(trace, name);
24      printf("%s\n", trace);
25
26      f = fopen(name, "rb");
27      if(f == NULL){
28          printf("File dont open\n");
29          bzero(buffer, size);
30          strcpy(buffer, "false");
31          //send(sock, buffer, size, 0);
32          sendto(sock, buffer, size, 0, (struct sockaddr*) &
                si_other, slen);
33          sleep(1);
34          return -1;
35      }
36      else{
37          bzero(buffer, size);
```

```
38          strcpy(buffer, "true");
39          //send(sock, buffer, size, 0);
40          sendto(sock, buffer, size, 0, (struct sockaddr*) &
                si_other, slen);
41          sleep(1);
42      }
43
44
45      //отправка размера
46      fseek(f, 0, SEEK_END);
47      fileSize = ftell(f);
48      fseek(f, 0, SEEK_SET);
49      printf("Size of file: %d\n", fileSize);
50      bzero(buffer, size);
51      //strcpy(buffer, (char)fileSize);
52      sprintf(buffer, "%d", fileSize);
53      //send(sock, buffer, size, 0);
54      sendto(sock, buffer, size, 0, (struct sockaddr*) &
            si_other, slen);
55      sleep(1);
56
57      while(fileSize > 0){
58          bzero(buffer, size);
59          if(fileSize >=1024){
60              message_length = fread(buffer, size, 1, f); // ко
                    личество
61
62              if(message_length != 0)
63                      //send(sock, buffer, size, 0);
64                      sendto(sock, buffer, size, 0, (struct
                            sockaddr*) &si_other, slen);
65                      sleep(1);
66          }
67          else{
68              message_length = fread(buffer, fileSize, 1, f);
69
70              if(message_length != 0)
71                      //send(sock, buffer, fileSize, 0);
72                      sendto(sock, buffer, fileSize, 0, (struct
                            sockaddr*) &si_other, slen);
73                      sleep(1);
74          }
75          fileSize -= size;
76      }
77      fclose(f);
78
79      return 0;
80  }
81
```

```
82 #endif // UPLOAD
```

view.h

```
1    #ifndef VIEW_H
2  #define VIEW_H
3
4  #include <string.h>
5  #include <unistd.h>
6  #include <dirent.h>
7  #include <sys/stat.h>
8  #include <stdlib.h>
9
10 #define MAXBUF 1024
11 void view(char *dir, int depth, int clientfd, struct
       sockaddr_in si_other, int slen) {
12
13     //chdir("/home/user/server/server_work");
14     // буфер для посылки
15     char buffer[MAXBUF];
16     DIR *dp;
17     struct dirent *entry;
18      struct stat statbuf;
19      if ((dp = opendir(dir)) == NULL) {
20              fprintf(stderr, "cannot␣open␣directory:␣%s\n",
                    dir);
21              return;
22     }
23
24     chdir(dir);
25     while((entry = readdir(dp)) != NULL) {
26          lstat(entry->d_name, &statbuf);
27          if (S_ISDIR(statbuf.st_mode)) {
28              //Находит каталог, но игнорирует . и ..
29                 if (strcmp(".", entry->d_name) == 0 || strcmp
                       ("..", entry->d_name) == 0)
30                             continue;
31
32                  bzero( buffer, MAXBUF);
33                  sprintf( buffer, "%d", depth);
34                  strcat(buffer, "|");
35                  strcat(buffer, "d");
36                  strcat(buffer, "|");
37                  strcat(buffer, entry->d_name);
38                  //send(clientfd, buffer, MAXBUF, 0);
39                  sendto(clientfd, buffer, MAXBUF, 0, (struct
                        sockaddr*) &si_other, slen);
40
41                 int new_depth = depth + 6;
42                  // Рекурсивный вызов с новый отступом
```

```
43                  view(entry->d_name, new_depth, clientfd,
                        si_other, slen);
44          }
45           else {
46              bzero( buffer, MAXBUF);
47              sprintf( buffer, "%d", depth);
48              strcat(buffer, "|");
49              strcat(buffer, "f");
50              strcat(buffer, "|");
51              strcat(buffer, entry->d_name);
52              //send(clientfd, buffer, MAXBUF, 0);
53              sendto(clientfd, buffer, MAXBUF, 0, (struct
                    sockaddr*) &si_other, slen);
54           }
55   }
56  chdir("..");
57  closedir(dp);
58  //chdir("/home/user/server/server_work");
59 }
60 #endif // VIEW_H
```

viewdir.h

```
1
2 #ifndef VIEW_DIR
3 #define VIEW_DIR
4 #include "view.h"
5
6 int view_dir(char *buffer, int buf_size, int sock, struct
    sockaddr_in si_other, int slen){
7     bzero(buffer, buf_size);
8     getcwd(buffer, buf_size);
9     view(buffer, 5, sock, si_other, slen);
10    return 0;
11 }
12
13 #endif // VIEW_DIR
```

UDP CLIENT

main.c

```
1 #define WIN32_LEAN_AND_MEAN
2 #include <stdio.h>
3 #include <conio.h>
4 #include <string.h>
5 #include <winsock2.h>
6 #include <windows.h>
```

```
7  #include <locale.h>
8  #include <Ws2tcpip.h>
9  #pragma comment(lib, "Ws2_32.lib")
10
11 #include "view.h"
12 #include "help.h"
13 #include "get_dir.h"
14 #include "view_dir.h"
15 #include "chdir.h"
16 #include "upload.h"
17 #include "rm_dir.h"
18 #include "mk_dir.h"
19 #include "echo.h"
20 #include "download.h"
21
22 #define PORT 1234
23 //#define SERVERADDR "192.168.1.43"
24 //#define SERVERADDR "10.1.99.25"
25 #define SERVERADDR "10.1.99.119"
26 #define MAXBUF  1024
27
28   int main()
29   {
30    //char* SERVERADDR;
31    //SERVERADDR = (char*)malloc(15);
32     setlocale(LC_ALL, "Russian");
33    char buffer[MAXBUF];
34
35    memset(buffer, 0, MAXBUF);
36    /*while(1){
37       printf("Cmd: ");
38       scanf("%s", buffer);
39       if(strcmp("connect", buffer) == 0){
40          printf("\nip: ");
41          scanf("%s", SERVERADDR);
42          break;
43       }
44       else{
45          printf("Waiting to connect command!\n");
46          continue;
47       }
48    }*/
49
50     // Шаг 1 - инициализация библиотеки Winsocks
51     if (WSAStartup(0x202,(WSADATA *)&buffer[0]))
52     {
53       printf("WSAStartup error: %d\n",
54              WSAGetLastError());
55       return -1;
```

```
56      }
57
58      // Шаг 2 - открытие сокета
59      SOCKET sock=socket(AF_INET, SOCK_DGRAM, 0);
60      if (sock==INVALID_SOCKET)
61      {
62        printf("socket() error: %d\n",WSAGetLastError());
63        WSACleanup();
64        return -1;
65      }
66
67      // Шаг 3 - обмен сообщений с сервером
68      HOSTENT *hst;
69      sockaddr_in dest_addr;
70     int slen = sizeof(dest_addr);
71
72      dest_addr.sin_family=AF_INET;
73      dest_addr.sin_port=htons(PORT);
74
75      // определение IP-адреса узла
76      if (inet_addr(SERVERADDR))
77        dest_addr.sin_addr.s_addr=inet_addr(SERVERADDR);
78      else
79        if (hst=gethostbyname(SERVERADDR))
80          dest_addr.sin_addr.s_addr=((unsigned long **)
81                hst->h_addr_list)[0][0];
82      else
83        {
84          printf("Unknown host: %d\n",WSAGetLastError());
85          closesocket(sock);
86          WSACleanup();
87          return -1;
88        }
89     //-------------------------------------------------------
90      //первая отправка
91       printf("Идет первая отправка...\n");
92       sendto(sock, "123", 3, 0, (struct sockaddr*) &dest_addr,
             slen);
93       //прием порта
94       memset(buffer, 0, MAXBUF);
95       int r;
96       if( (r=recvfrom(sock, buffer, MAXBUF, 0, (struct
            sockaddr *) &dest_addr, &slen)) != -1 ){
97        printf("Порт принят!\n");
98       }
99     //закрытие старого сокета
100    closesocket(sock);
101     WSACleanup();
102    printf("Закрыт старый сокет\n");
```

```
103
104     int newport = atoi(buffer);
105     printf("Установлен␣новый␣порт\n");
106
107     WSADATA wsa2;
108     if (WSAStartup(MAKEWORD(2, 2), &wsa2) != 0) {
109        exit(EXIT_FAILURE);
110     }
111
112     struct sockaddr_in new_si_other;
113     int sockfd, new_slen = sizeof(new_si_other);
114
115     //новый сокет
116     if ((sockfd = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP)) ==
            SOCKET_ERROR) {
117        exit(EXIT_FAILURE);
118     }
119     else
120        printf("Создан␣новый␣сокет\n");
121
122     //установка параметров
123     memset((char *)&new_si_other, 0, new_slen);
124     new_si_other.sin_family = AF_INET;
125     new_si_other.sin_port = htons(newport);
126     new_si_other.sin_addr.S_un.S_addr = inet_addr(SERVERADDR);
127     printf("Установлены␣параметры\n");
128
129     //новая отправка
130     printf("Новая␣отправка\n");
131     if((r=sendto(sockfd, "345", 3, 0, (struct sockaddr *) &
           new_si_other, new_slen)) != -1)
132        printf("Успешная␣новая␣отправка\n");
133     else
134        printf("Новая␣отправка␣не␣удалась\n");
135
136     // ожилание приема сообщение о работе диспетчера
137     memset(buffer, 0, MAXBUF);
138     if ((r=recvfrom(sockfd, buffer, MAXBUF, 0, (struct
           sockaddr *) &new_si_other, &new_slen)) != -1){
139        printf("Диспетчер␣работает\n");
140        printf("%s", buffer);
141     }
142     else
143        printf("Диспетчер␣не␣работает\n");
144     printf("%s", buffer);
145     //-------------------------------------------------------
146      printf("Для␣получения␣справки␣введите␣команду␣help\n");
147      while(1){
148        memset(buffer, 0, MAXBUF);
```

```
149         printf("Cmd: ");
150         scanf("%s" , buffer);
151         // команды
152         if(strcmp(buffer, "view") == 0){
153          view(buffer, sockfd, MAXBUF, new_si_other, new_slen)
                ;
154       }
155         else if(strcmp(buffer, "view_dir") == 0)
156             view_dir(buffer, sockfd, MAXBUF, new_si_other,
                   new_slen);
157         else if(strcmp(buffer, "help") == 0)
158             get_help();
159         else if(strcmp(buffer, "get_dir") == 0)
160             get_dir(buffer, sockfd, MAXBUF, new_si_other,
                   new_slen);
161         else if(strcmp(buffer, "ch_dir") == 0)
162             ch_dir(buffer, sockfd, MAXBUF, dest_addr,
                   new_slen);
163         else if(strcmp(buffer, "upload") == 0)
164             upload(buffer, MAXBUF, sockfd, new_si_other,
                   new_slen);
165       else if(strcmp(buffer, "download") == 0)
166             download(buffer, MAXBUF, sockfd, new_si_other,
                   new_slen);
167         else if(strcmp(buffer, "mk_dir") == 0)
168             mk_dir(buffer, MAXBUF, sockfd, new_si_other,
                   new_slen);
169         else if(strcmp(buffer, "rm_dir") == 0)
170             rm_dir(buffer, MAXBUF, sockfd, new_si_other,
                   new_slen);
171       else if(strcmp(buffer, "echo") == 0)
172             echo(buffer, sockfd, MAXBUF, new_si_other,
                   new_slen);
173       else if(strcmp(buffer, "disconnect") == 0){
174             sendto(sockfd, "disconnect", 10, 0, (struct
                   sockaddr*) &new_si_other, new_slen);
175          break;
176       }
177         else
178             printf("unknown comand\n");
179         printf("Для получения справки введите команду help\n"
              );
180     }
181    //-----------------------------------------------
182    printf("\nExit\n");
183     closesocket(sockfd);
184     WSACleanup();
185    _getch();
186     return 0;
```

```
187   }
```

headers

chdir.h

```
1  #ifndef CHDIR
2  #define CHDIR
3  #include <string.h>
4
5  int ch_dir(char *buffer, int sock, int size, struct
       sockaddr_in dest_addr, int slen){
6     memset(buffer, 0, size);
7      strcpy(buffer, "ch_dir\r\n");
8      //send(sock, buffer, size, 0);
9     sendto(sock, buffer, size, 0, (struct sockaddr*) &
          dest_addr, slen);
10
11     printf("путь: ");
12     memset(buffer, 0, size);
13     scanf("%s", buffer);
14     //send(sock, buffer, size, 0);
15    sendto(sock, buffer, size, 0, (struct sockaddr*) &
          dest_addr, slen);
16
17     //прием ошибки либо сообщения об успехе
18     memset(buffer, 0, size);
19     //recv(sock, buffer, size, 0);
20    recvfrom(sock, buffer, size, 0, (struct sockaddr *) &
          dest_addr, &slen);
21     if(strcmp(buffer, "succes") != 0)
22        printf("Неверный каталог!");
23
24     memset(buffer, 0, size);
25     //recv(sock, buffer, size, 0);
26    recvfrom(sock, buffer, size, 0, (struct sockaddr *) &
          dest_addr, &slen);
27     printf("Текущий каталог: %s\n", buffer);
28     return 0;
29 }
30
31 #endif // CHDIR
```

download.h

```
1  #ifndef DOWNLOAD
2  #define DOWNLOAD
3  #include <string.h>
```

```
4  #include <stdio.h>
5  #include <math.h>
6  #include <sys/stat.h>
7  #include <time.h>
8  #define MAXBUF  1024
9
10 int download(char *buffer, size_t size, int sock, struct
       sockaddr_in dest_addr, int slen){
11     int r;
12     int p;
13     int i = 0;
14     int j = 0;
15     int k = 0;
16     int fileSize = 0;
17    time_t start_time, finish_time;
18     char send_buf[MAXBUF], name[MAXBUF], trace[MAXBUF];
19
20    //отправка команды
21    memset(buffer, 0, size);
22     strcpy(buffer, "download\r\n");
23     //send(sock, buffer, size, 0);
24    sendto(sock, buffer, size, 0, (struct sockaddr*) &
          dest_addr, slen);
25
26    // отправка на сервер имени файла
27     printf("Имя на сервере: ");
28     memset(name, 0, size);
29     scanf("%s",name);
30     //send(sock, name, size, 0);
31    sendto(sock, name, size, 0, (struct sockaddr*) &dest_addr,
           slen);
32
33    //ожидание подтверждения открытия файла
34    memset(buffer, 0, size);
35    //recv(sock, buffer, size, 0);
36    recvfrom(sock, buffer, size, 0, (struct sockaddr *) &
          dest_addr, &slen);
37    if(strcmp(buffer, "false") == 0){
38       printf("Incorrect name - file dont open!\n");
39       return -1;
40    }
41
42    // где сохранять
43    printf("Путь (куда сохранить): ");
44     memset(trace, 0, size);
45     scanf("%s",trace);
46    printf("\n");
47    //формирование пути с именем файла
48    strcat(trace, "\\");
```

```
49    strcat(trace, name);
50
51    FILE* f;
52    f = fopen(trace, "wb");
53    if(f == NULL){
54        printf("File dont create\n");
55        return -1;
56    }
57
58     //прием и формирование размера файла
59     memset(buffer, 0, size);
60     //recv(sock, buffer, size, 0);
61    recvfrom(sock, buffer, size, 0, (struct sockaddr *) &
          dest_addr, &slen);
62     while(buffer[i] != '\0'){
63           i++;
64           k++;
65      }
66      for(j=0; j<k; j++){
67           p = pow(10.0,i-1);
68           fileSize+=p*((int)buffer[j]-48);
69           i--;
70      }
71     //printf("size string: %s\n", buffer);
72     printf("SIze:%d\n", fileSize);
73
74    int fileSize_copy = fileSize;
75    // запомнить время начала
76    start_time = time(NULL);
77    while(fileSize > 0){
78        if(fileSize >=1024){
79            memset(buffer, 0, size);
80            //r = recv(sock, buffer, size, 0);
81          r = recvfrom(sock, buffer, size, 0, (struct sockaddr
                *) &dest_addr, &slen);
82            fwrite(buffer, 1, size, f); // записываем весь буф
                ер
83
84            // печать
85            memset(send_buf, 0, size);
86            printf("Принято %d\n",r );
87        }
88        else{
89            memset(buffer,0, size);
90            //r = recv(sock, buffer, size, 0);
91          r = recvfrom(sock, buffer, size, 0, (struct sockaddr
                *) &dest_addr, &slen);
92            fwrite(buffer, 1, fileSize, f); //записываем тольк
                о сколько надо
```

```
93
94              // отправка клиенту
95              memset(send_buf,0, size);
96              printf("Принято %d\n",r );
97          }
98          fileSize -=1024;
99      }
100     // запомнить время завершения передачи
101     finish_time = time(NULL);
102     fclose(f);
103     //проверка
104     struct stat fi;
105     stat(trace,&fi);
106     if(fileSize > fi.st_size)
107        printf("Файл получен с ошибкой! Осуществите повторный п
           рием!\n");
108     printf("Время передачи: %f\n", difftime(finish_time,
           start_time));
109     return 0;
110 }
111
112 #endif // DOWNLOAD
```

getdir.h

```
1  #ifndef GET_DIR
2  #define GET_DIR
3  #include <string.h>
4
5  int get_dir(char *buffer, int sock, int size, struct
       sockaddr_in dest_addr, int slen){
6       memset(buffer, 0, size);
7       strcpy(buffer, "get_dir\r\n");
8       //send(sock, buffer, size, 0);
9      sendto(sock, buffer, size, 0, (struct sockaddr*) &
           dest_addr, slen);
10
11      // прием
12      memset(buffer, 0, size);
13      //recv(sock, buffer, size,0);
14     recvfrom(sock, buffer, size, 0, (struct sockaddr *) &
           dest_addr, &slen);
15      printf("Текущий каталог: %s\n", buffer);
16
17      return 0;
18 }
19 #endif // GET_DIR
```

mkdir.h

```
#ifndef MK_DIR
#define MK_DIR
#include <string.h>

int mk_dir(char* buffer, int size, int sock, struct
    sockaddr_in dest_addr, int slen){
     memset(buffer, 0, size);
     strcpy(buffer, "mk_dir\r\n");
     //send(sock, buffer, size, 0);
    sendto(sock, buffer, size, 0, (struct sockaddr*) &
       dest_addr, slen);

     memset(buffer, 0, size);
     printf("Имя папки: ");
     scanf("%s", buffer);
     //send(sock, buffer, size, 0);
    sendto(sock, buffer, size, 0, (struct sockaddr*) &
       dest_addr, slen);

     memset(buffer, 0, size);
     //recv(sock, buffer, size, 0);
    recvfrom(sock, buffer, size, 0, (struct sockaddr *) &
       dest_addr, &slen);
     printf("%s", buffer);
     return 0;
}

#endif // MK_DIR
```

rmdir.h

```
#ifndef RM_DIR
#define RM_DIR
#include <string.h>

int rm_dir(char* buffer, int size, int sock, struct
    sockaddr_in dest_addr, int slen){
     memset(buffer, 0, size);
     strcpy(buffer, "rm_dir\r\n");
     //send(sock, buffer, size, 0);
    sendto(sock, buffer, size, 0, (struct sockaddr*) &
       dest_addr, slen);

     memset(buffer, 0, size);
     printf("Имя папки: ");
     scanf("%s", buffer);
     printf("\n");
     //send(sock, buffer, size, 0);
    sendto(sock, buffer, size, 0, (struct sockaddr*) &
       dest_addr, slen);
```

```
17
18      memset(buffer, 0, size);
19      //recv(sock, buffer, size, 0);
20      recvfrom(sock, buffer, size, 0, (struct sockaddr *) &
            dest_addr, &slen);
21      printf("%s", buffer);
22
23      return 0;
24  }
25
26  #endif // RM_DIR
```

upload.h

```
1   #ifndef UPLOAD
2   #define UPLOAD
3   #include <string.h>
4   #include <stdio.h>
5   #include <time.h>
6   #include <stdlib.h>
7
8   int upload(char* buffer, int size, int sock, struct
        sockaddr_in dest_addr, int slen){
9       char name[1024];
10      //int size = ssize;
11      //char *f_buf;//[1024];
12      long int fileSize;
13      long int message_length;
14
15      time_t start_time, finish_time;
16      FILE* f;
17
18      //f_buf = (char*) malloc(fbuf_size);
19
20      printf("Путь: ");
21      memset(name, 0, 1024);
22      scanf("%s", name);
23
24      f = fopen(name, "rb");
25      if(f == NULL){
26          printf("File dont open\n");
27          return -1;
28      }
29
30      memset(buffer, 0, size);
31      strcpy(buffer, "upload\r\n");
32      //send(sock, buffer, size, 0);
33      sendto(sock, buffer, size, 0, (struct sockaddr*) &
            dest_addr, slen);
34      Sleep(0.01);
```

```
35
36
37      //отправка размера
38      fseek(f, 0, SEEK_END);
39      fileSize = ftell(f);
40      fseek(f, 0, SEEK_SET);
41      printf("Size of file: %d\n", fileSize);
42      memset(buffer, 0, size);
43      //strcpy(buffer, (char)fileSize);
44      sprintf(buffer, "%d", fileSize);
45      //send(sock, buffer, size, 0);
46    sendto(sock, buffer, size, 0, (struct sockaddr*) &
          dest_addr, slen);
47    Sleep(1);
48
49
50    // отправка на сервер имени файла
51     printf("Имя на сервере: ");
52     memset(buffer, 0, size);
53     scanf("%s",buffer);
54     //send(sock, buffer, size, 0);
55    sendto(sock, buffer, size, 0, (struct sockaddr*) &
          dest_addr, slen);
56    Sleep(0.01);
57
58    // запомнить время начала
59    start_time = time(NULL);
60
61    while(fileSize > 0){
62         memset(buffer, 0, size);
63         if(fileSize >= size){
64          memset(buffer, 0, size);
65             message_length = fread(buffer, size, 1, f); // ко
                  личество
66
67             if(message_length != 0){
68                     //send(sock, buffer, size, 0);
69                sendto(sock, buffer, size, 0, (struct sockaddr
                      *) &dest_addr, slen);
70                Sleep(0.01);
71           }
72             memset(buffer, 0, size);
73             //recv(sock, buffer, size, 0);
74           recvfrom(sock, buffer, size, 0, (struct sockaddr *)
                &dest_addr, &slen);
75             printf("Передача - было принято сервером: %s\n",
                   buffer);
76         }
77         else{
```

```
78          memset(buffer, 0, size);
79             message_length = fread(buffer, fileSize, 1, f);
80
81             if(message_length != 0){
82                     //send(sock, buffer, fileSize, 0);
83                sendto(sock, buffer, fileSize, 0, (struct
                      sockaddr*) &dest_addr, slen);
84                Sleep(0.01);
85          }
86             memset(buffer, 0, size);
87             //recv(sock, buffer, size, 0);
88          recvfrom(sock, buffer, size, 0, (struct sockaddr *)
               &dest_addr, &slen);
89             printf("Передача - было принято сервером: %s\n",
                   buffer);
90         }
91         fileSize -= size;
92     }
93
94    finish_time = time(NULL);
95     fclose(f);
96    printf("Время передачи: %f\n", difftime(finish_time,
          start_time));
97
98     return 0;
99 }
100
101 #endif // UPLOAD
```

view.h

```
1  #ifndef VIEW_H
2  #define VIEW_H
3  #include <string.h>
4  #include <math.h>
5
6  int view(char* buffer, int sock, int buf_size, struct
       sockaddr_in dest_addr, int slen){
7      printf("Рабочий каталог: server_work/\n");
8      memset(buffer, 0, buf_size);
9      strcpy(buffer, "view\r\n");
10     //send(sock, buffer, buf_size, 0);
11    sendto(sock, buffer, buf_size, 0, (struct sockaddr*) &
          dest_addr, slen);
12
13     while(1){
14         int depth;
15         int i;
16         int j;
17         int k;
```

```
18          memset(buffer, 0, buf_size);
19          //recv(sock, buffer, buf_size,0);
20        recvfrom(sock, buffer, buf_size, 0, (struct sockaddr *)
              &dest_addr, &slen);
21          if(strcmp(buffer, "0|d|end")==0){
22              printf("end\n");
23              break;
24          }
25          else{
26              i = 0;
27              j = 0;
28              k = 0;
29              int p=0;
30             depth = 0;
31             // подсчет количества знаков числа
32             while(buffer[i] != '|'){
33                    i++;
34                    k++;
35              }
36              for(j=0; j<k; j++){
37                  p = pow(10.0,i-1);
38                  depth+=p*((int)buffer[j]-48);
39                  i--;
40              }
41              //
                    ----------------------------------------------------

42              k++; // знак '|'
43              if(buffer[k] == 'f')
44                  p = 0;
45              else
46                  p = 1;
47              //
                    ----------------------------------------------------

48              k++; // знак '|'
49              k++;
50              // печать пробелов
51              for(i=0; i<depth; i++)
52                  printf("␣");
53              // печать названия
54              while(buffer[k] != '\0'){
55                  printf("%c", buffer[k]);
56                  k++;
57              }
58              // печать символа папки и перевод строки
59              if(p == 1)
60                  printf("/\n");
61              else
```

```
                printf("\n");
        }
 }
    return 0;
}

#endif // VIEW_H
```

viewdir.h

```
#ifndef view_dir_DIR_H
#define view_dir_DIR_H
#include <string.h>
#include <math.h>

int view_dir(char* buffer, int sock, int size, struct
   sockaddr_in dest_addr, int slen){
    memset(buffer, 0, size);
    strcpy(buffer, "view_dir\r\n");
    //send(sock, buffer, size, 0);
   sendto(sock, buffer, size, 0, (struct sockaddr*) &
      dest_addr, slen);
    while(1){
        int depth;
        int i;
        int j;
        int k;
        memset(buffer, 0, size);
        //recv(sock, buffer, size,0);
      recvfrom(sock, buffer, size, 0, (struct sockaddr *) &
         dest_addr, &slen);
        if(strcmp(buffer, "0|d|end")==0){
            printf("end\n");
            break;
        }
        else{
            i = 0;
            j = 0;
            k = 0;
            int p=0;
           depth = 0;
           // подсчет количества знаков числа
           while(buffer[i] != '|'){
                 i++;
                 k++;
            }
            for(j=0; j<k; j++){
                p = pow(10.0,i-1);
                depth+=p*((int)buffer[j]-48);
                i--;
```

```
38              }
39              //
                    ------------------------------------------------

40              k++; // знак '|'
41              if(buffer[k] == 'f')
42                  p = 0;
43              else
44                  p = 1;
45              //
                    ------------------------------------------------

46              k++; // знак '|'
47              k++;
48              // печать пробелов
49              for(i=0; i<depth; i++)
50                  printf("␣");
51              // печать названия
52              while(buffer[k] != '\0'){
53                  printf("%c", buffer[k]);
54                  k++;
55              }
56              // печать символа папки и перевод строки
57              if(p == 1)
58                  printf("/\n");
59              else
60                  printf("\n");
61          }
62  }
63      return 0;
64  }
65
66  #endif // view_dir_dir_DIR
```

help.h

```
1  #ifndef HELP
2  #define HELP
3  #include <string.h>
4
5  int get_help(){
6      printf("ch_dir␣-␣сменить␣текущую␣директорию␣на␣сервере\n"
           );
7      printf("mk_dir␣-␣создать␣папку␣в␣текущей␣директории\n");
8      //printf("download <имя файла> - загрузить файл с сервера
           \n");
9      printf("upload␣-␣загрузить␣файл␣на␣сервер␣из␣рабочей␣дире
           ктории\n");
10     printf("view␣-␣просмотр␣дерева␣каталогов␣сервера\n");
```

```
11      //printf("connect <IP> <номер порта> - подключиться к сер
            веру \n");
12      printf("disconnect - отключиться от сервера\n");
13      return 0;
14  }
15
16  #endif // HELP
```