《C语言实现 FTP 服务器》

C 语言实现 FTP 服务器 - 实验楼

C语言实现 FTP 服务器

一、实验简介

FTP 是一种标准协议，用于将一台计算机上的文件通过 Internet 复制到另一台计算机上。本实验根据 FTP 协议，用 C 实现了一个 FTP 服务器。通过该实验的学习，可以了解到 FTP 协议机制，更深入的理解 Linux 网络编程。

1.1 知识点

- FTP 协议
- Linux 系统编程
- Linux 套接字网络编程

1.2 效果截图

- 运行服务器程序（工作在8080端口）

```
shiyanlou:server/ $ ./ftserve 8080 &                                [21:13:54]
[1] 7238
```

- 运行客户端程序

```
shiyanlou:client/ $ ./ftclient 7523fcd596fb 8080                    [21:20:39]
Connected to 7523fcd596fb.
220 Welcome, server ready.
Name: anonymous
Password:
Successful login.
```

- 输入命令 `list`:

```
ftclient> list
-rwxrwxr-x 1 shiyanlou shiyanlou 42009 Sep  4 21:13 ftserve
-rw-rw-r-- 1 shiyanlou shiyanlou  6848 Aug 28 13:49 ftserve.c
-rw-rw-r-- 1 shiyanlou shiyanlou   431 Aug 28 00:40 ftserve.h
-rw-rw-r-- 1 shiyanlou shiyanlou 43200 Sep  4 21:13 ftserve.o
-rw-rw-r-- 1 shiyanlou shiyanlou   288 Aug 28 00:40 makefile
```

- 输入命令 `get`:

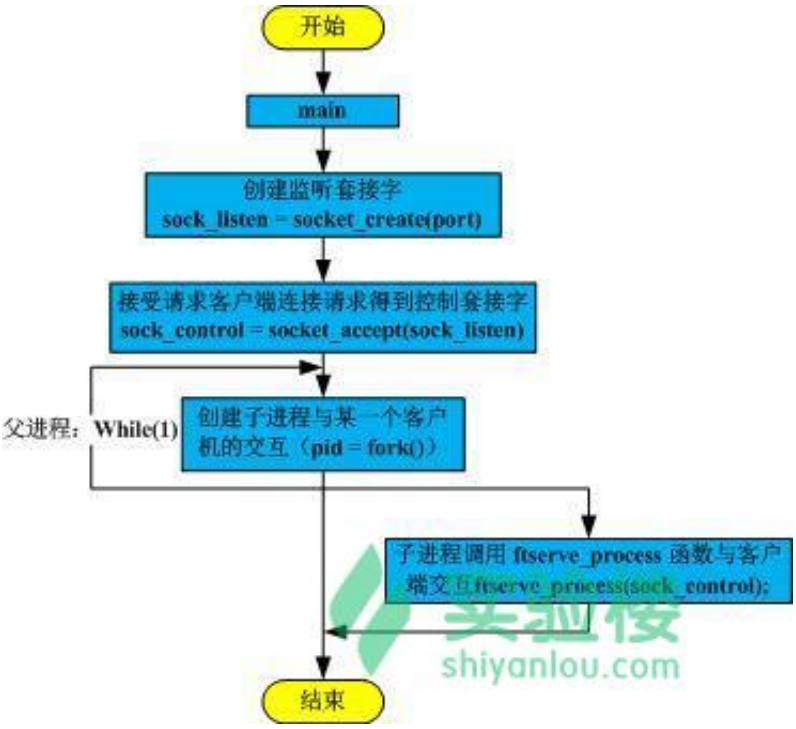
```
ftclient> get ftserve
226 Closing data connection. Requested file action successful.
```

- 检查是否下载成功:

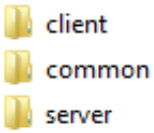
```
shiyanlou:client/ $ ll
total 156K
-rwxrwxr-x 1 shiyanlou shiyanlou  44K Sep  4 21:10 ftclient
-rw-rw-r-- 1 shiyanlou shiyanlou 6.8K Sep  4 17:58 ftclient.c
-rw-rw-r-- 1 shiyanlou shiyanlou  417 Aug 28 00:40 ftclient.h
-rw-rw-r-- 1 shiyanlou shiyanlou  49K Sep  4 21:10 ftclient.o
-rw-rw-r-- 1 shiyanlou shiyanlou  42K Sep  4 21:22 ftserve
-rw-rw-r-- 1 shiyanlou shiyanlou  293 Aug 28 00:40 makefile
```

可以看出来文件 ftserve 下载到本地。

1.3 程序框架



1.4 整个项目的组织结构



- client 文件夹保存的是客户端程序
- common 文件夹保存的是用于通信的通用程序
- server 文件夹保存的是服务器程序，本实验重点是服务器端程序

目录布局如下：

本项目全代码可在 http://labfile.oss.aliyuncs.com/courses/628/ftp.zip 下载。

二、FTP 协议

2.1 FTP 概念

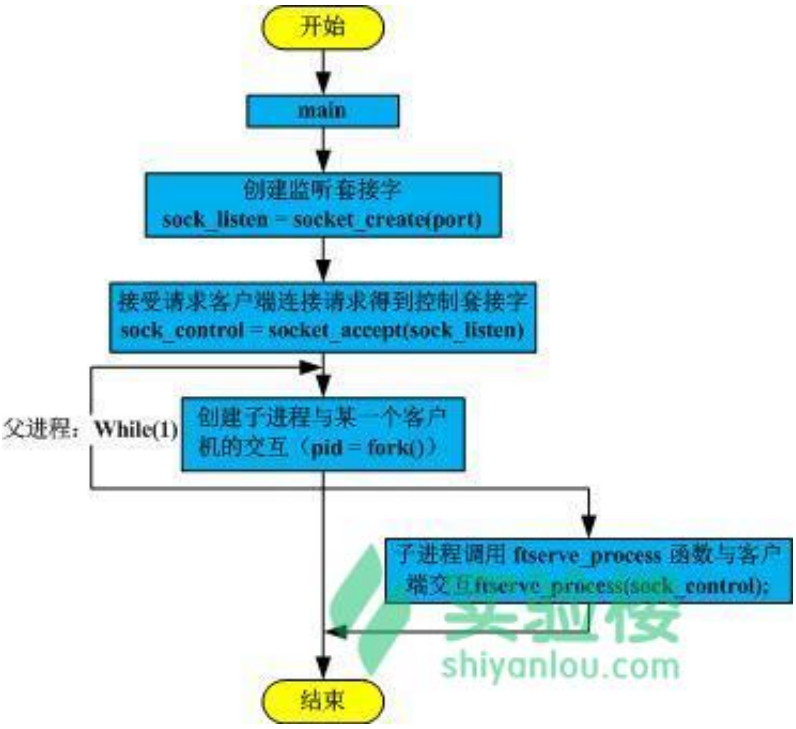
FTP（File Transfer Protocol）协议，中文名：文件传输协议。是用于在网络上进行文件传输的一套标准协议。它属于网络传输协议的应用层。FTP服务一般运行在20和21两个端口。端口20用于在客户端和服务器之间传输数据流，而端口21用于传输控制流，并且是命令通向ftp服务器的入口。

2.2 FTP 实现的目标

1. 促进文件的共享（计算机程序或数据）
2. 鼓励间接或者隐式的使用远程计算机
3. 向用户屏蔽不同主机中各种文件存储系统（File system）的细节
4. 可靠和高效的传输数据

三、main 函数监听客户连接请求

3.1 main 函数的框架



3.2 首先进行命令行合法性检测

在服务器端应该设置服务器的端口号等信息，通过命令行参数的形式传递给服务器程序：

```
if
printf "usage: ./ftserve port\n"
exit
```

3.2 获取服务器端口号并创建监听套接字

按理说应该在端口 21 上开启 FTP 服务，这里提供一个更灵活的方式，让操作人员可以自由选择端口。

- 获取端口号，并转换为整数：

```
port = atoi(argv[1]);
```

- 创建监听套接字：

创建方式可以参看课程 C 语言实现聊天室软件，这里不再赘述。本实验中调用 socket_create 自定义模块创建监听套接字：

```
if
    "Error creating socket"
exit
```

3.3 创建死循环监听客户机的请求

这部分的框架就是利用创建好的接听套接字 sock_listen 循环监听客户端的连接请求，一旦监听到客户端的连接请求，则创建子进程处理这个已连接的套接字。

- 监听套接字接受连接请求，得到控制套接字，用于传递控制信息：

```
sock_control = socket_accept(sock_listen)
```

- 创建子进程处理用户请求：

```
if      fork     0
    "Error forking child process"
```

- 子进程调用 ftserve_process 函数与客户端交互

```
elseif      0

close
close
exit 0
```

四、ftserve_process 模块

ftserve_process 模块提供对某个用户的服务请求。ftserve_process 模块的生命周期是客户端从连接上服务器的开始到结束。完成的功能主要有：

1. 用户认证
2. 处理用户的请求

4.1 模块的框架

4.2 用户认证

首先发送欢迎应答码：

```
send_response(sock_control, 220);
```

调用 `ftserve_login` 函数认证：

```
if
else

exit
```

4.3 处理用户的请求

首先这是一个利用死循环来不停的接收客户端的指令，来提供服务的：

```
while (1) {}
```

下面我们来看看循环体中的内容:

- 第一步：接收命令，并解析,获得命令和参数：

```
int rc = ftserve_recv_cmd(sock_control, cmd, arg);
```

- 第二步：创建和客户端的数据连接：

```
sock_data = ftserve_start_data_conn(sock_control)
```

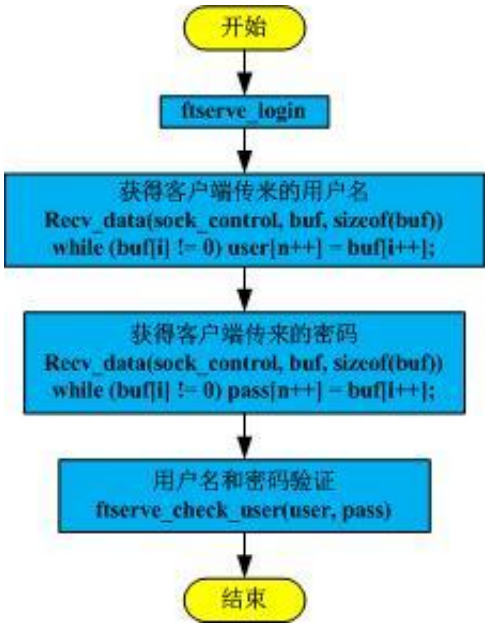
- 第三步：执行指令：

```
if  strcmp      "LIST"   0
elseif  strcmp      "RETR"   0
```

五、用户登录

函数 `ftserve_login` 的逻辑非常简单，先获取用户名和密码，然后调用`ftserve_check_user(user, pass)` 通过查看 ".auth" 文件 中的用户信息，验证该用户的合法性。

5.1 模块框架



5.2 获取用户名

- 首先应该接受客户端包含用户名的注册信息,并保存到 `buf` 中:

```
if                                  sizeof          -1
    "recv error\n"
exit 1
```

- 获得用户名。因为 `buf` 中的前四个字符为命令字符 `USER`，所以真正的用户名应该是从 `buf[5]` 开始的，代码如下:

```
int     5
int     0
while           0
                                331
```

5.3 获得用户密码

- 首先应该接受客户端包含用户密码的信息,并保存到 `buf` 中:

```
if                                  sizeof          -1
    "recv error\n"
exit 1
```

- 获得用户密码。因为 `buf` 中的前四个字符为命令字符 `PASS`，所以真正的用户名应该是从 `buf[5]` 开始的，代码如下：

```
i   5
n   0
while [illegible]   0
[illegible]
```

5.4 用户名和密码验证

调用 `ftserve_check_user(user, pass)` 函数验证用户的合法性，输入为刚才获得的用户名(user)和密码(pass)，代码如下：

```
return (ftserve_check_user(user, pass));
```

- `ftserve_check_user(user, pass)` 函数最主要的步骤就是和 ".auth" 文件的信息做比对。具体代码如下：

```
if   strcmp [illegible]   0     strcmp [illegible]   0
[illegible]   1
break
```

六、实验总结

通过本实验，可以更好的理解 FTP 协议，套接字编程。熟悉 C/S 编程模型。

- 本实验的完整代码如下：

```
#include"ftserve.h"/* 主函数入口 */intmain int     char
[illegible]
int [illegible]

/* 命令行合法性检测 */if     2
[illegible]
printf "usage: ./ftserve port\n"
exit 0
```

```
/* 将命令行传进来的服务器端口号（字符串）转换为整数 */
                1

/* 创建监听套接字 */if                                          0

        "Error creating socket"
exit 1

/* 循环接受不同的客户机请求 */while 1

/* 监听套接字接受连接请求，得到控制套接字，用于传递控制信息 */if
                                                    0
break

/* 创建子进程处理用户请求 */if                    0
        "Error forking child process"

/* 子进程调用 ftserve_process 函数与客户端交互 */elseif          0

                    // 子进程关闭父进程的监听套接字

                    //用户请求处理完毕，关闭该套接字exit 0

                    // 父进程关闭子进程的控制套接字


return0

/**
* 通过数据套接字发送特定的文件
* 控制信息交互通过控制套接字
* 处理无效的或者不存在的文件名
*/
voidftserve_retr int                    int              char

          NULL
char
size_t
                    "r"   // 打开文件if
                          550   // 发送错误码（550 Requested
```

```
action not taken)else

                        150   // 发送 okay (150 File status
okay)do

                    1                   // 读文件内容if
0
printf "error in fread()\n"

if                                  0    0  // 发送数据（文件内容）
        "error sending file\n"


while       0

                        226   // 发送消息: 226: closing conn,
file transfer successful


/**
* 响应请求: 发送当前所在目录的目录项列表
* 关闭数据连接
* 错误返回 -1, 正确返回 0
*/
intftserve_list int              int
char
size_t

int          "ls -l | tail -n+2 > tmp.txt"   //利用系统调用函数
system 执行命令, 并重定向到 tmp.txt 文件if       0

exit 1


        "tmp.txt"  "r"
if
exit 1

/* 定位到文件的开始处 */
              0

                        1
```

```
memset    0

/* 通过数据连接，发送tmp.txt 文件的内容 */while
    1          0

if                    0    0
    "err"

memset    0


                226   // 发送应答码 226（关闭数据连接，请求的文件操作成功）return0

/**
* 创建到客户机的一条数据连接
* 成功返回数据连接的套接字
* 失败返回 -1
*/
intftserve_start_data_conn int
char    1024
int

if                sizeof    0    0

    "Error while waiting"
return-1


struct
socklen_t      sizeof
                    struct
// 获得与控制套接字关联的外部地址（客户端地址）
                                    sizeof

/* 创建到客户机的数据连接 */if
                                0
return-1

return
```

```
/**
* 用户资格认证
* 认证成功返回 1，否则返回 0
*/
intftserve_check_user char        char
char
char
char
char
char          NULL
size_t
size_t        0

int           0

          ".auth"  "r"    //打开认证文件（记录用户名和密码）if
NULL

       "file not found"
exit 1

/* 读取".auth" 文件中的用户名和密码，验证用户身份的合法性 */while
                                          -1

memset        0
strcpy

          " "
strcpy

if          NULL

            NULL  " "
strcpy

/* 去除字符串中的空格和换行符 */
                    int strlen

if   strcmp                  0      strcmp                  0

       1  // 匹配成功，标志变量 auth = 1，并返回break
```

```
free
return

/* 用户登录*/intftserve_login int
char
char
char
memset 0
memset 0
memset 0

/* 获得客户端传来的用户名 */if
sizeof -1
"recv error\n"
exit 1

int 5
int 0
while 0  //buf[0-4]="USER"

/* 用户名正确，通知用户输入密码 */
331

/* 获得客户端传来的密码 */memset 0
if sizeof -1
"recv error\n"
exit 1

5
0
while 0  // buf[0 - 4] = "PASS"

return // 用户名和密码验证，并返回

/* 接收客户端的命令并响应，返回响应码 */intftserve_recv_cmd int
char char
int 200
```

```
char

memset      0
memset    0  5
memset    0

/* 接受客户端的命令 */if
sizeof              -1

    "recv error\n"
return-1

/* 解析出用户的命令和参数 */strncpy          4
char          5
strcpy

if  strcmp     "QUIT"   0
    221

elseif   strcmp     "USER"    0     strcmp     "PASS"    0
   strcmp     "LIST"    0     strcmp     "RETR"    0
    200

else
    502  // 无效的命令

return

/* 处理客户端请求 */voidftserve_process int
int
char    5
char

                          220   // 发送欢迎应答码/* 用户认证 */if
                             1  // 认证成功
                          230
else

                          430   // 认证失败exit 0

/* 处理用户的请求 */while  1
```

```
/* 接收命令，并解析,获得命令和参数 */int

if      0       221   // 用户输入命令 "QUIT"break

if      200

/* 创建和客户端的数据连接 */if

                                        0

exit 1

/* 执行指令 */if  strcmp     "LIST"   0

elseif  strcmp     "RETR"   0

                // 关闭连接
```

参考资料

- 《UNIX环境高级编程》
- https://github.com/beckysag/ftp

C 语言实现 FTP 客户端 - 实验楼

Source URL: https://www.shiyanlou.com/courses/628/labs/2110/document

C语言实现多线程排

序

一、实验简介

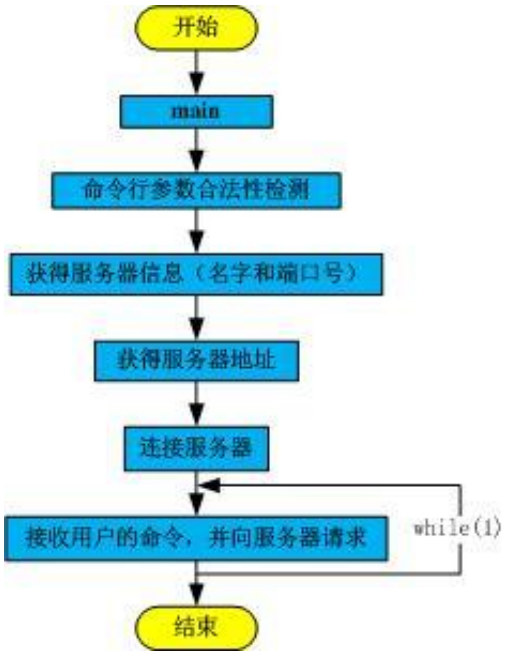
本实验实现的 FTP 是配合第一个实验 FTP 服务器的程序。用来和 FTP 服务器通信。

1.1 知识点

- FTP 协议
- Linux 系统编程
- Linux 套接字网络编程

1.3 程序框架

- 本项目的主框架如下：



二、main 函数

2.1 准备工作

命令行参数合法性检测:

```
if
printf "usage: ./ftclient hostname port\n"
exit
```

从命令行中获得主机名和端口号:

```
char
char
```

获得服务器地址,这里用到函数 `getaddrinfo`:

```
memset 0 sizeof struct
if 0
printf "getaddrinfo() error %s"
exit 1
```

与服务器取得连接:

- 主要步骤:

1. 创建控制套接字
2. 和服务器连接

通过 `sock_control = socket(rp->ai_family, rp->ai_socktype, rp->ai_protocol);` 语句创建控制套接字。

通过 `connect(sock_control, res->ai_addr, res->ai_addrlen)==0;` 和服务器连接。

获取用户的名字和密码:

```
ftclient_login();
```

2.2 处理用户命令:

得到用户输入的命令:

```
if                                    sizeof          0

printf "Invalid command\n"
continue
```

发送命令到服务器:

```
if  send                        int             0    0

close
exit 1
```

接收服务器响应（来自控制套接字）,询问服务器是否可以支持该命令:

```
retcode = read_reply();
```

如果服务器支持该命令，则打开数据连接:

```
if

      "Error opening socket for data connection"
exit
```

执行命令:

如果是 `LIST` 命令，则调用 `ftclient_list(data_sock, sock_control);` 语句与服务器交互:

```
if  strcmp          "LIST"     0
```

如果是 `RETR` 命令，则调用 `ftclient_get(data_sock, sock_control, cmd.arg);` 语句与服务器交互:

```
elseif  strcmp          "RETR"     0

if                  550

            550

continue
```

三、ftclient_get 函数

`ftclient_get` 函数完成的功能是从数据连接上接收服务器传来的数据（文件内容），并写入本地文件。

- 将服务器传来的数据（文件内容）写入本地建立的文件

```
while (1) {
    recv(data_sock, data_buf, BUFSIZE, 0);
    if (n == 0)
        break;
    fwrite(data_buf, 1, n, fp);
}
```

四、ftclient_list 函数

`ftclient_list` 函数的功能是从数据连接上接收服务器传来的数据（服务器 `ls` 的结果），并打印。

- 等待服务器启动的信息:

```
if (recv(ctrl_sock, ctrl_buf, sizeof(ctrl_buf), 0) < 0) {
    perror("client: error reading message from server\n");
    return -1;
}
```

- 接收服务器传来的数据:

```
while (1) {
    n = recv(data_sock, data_buf, BUFSIZE, 0);
    if (n == 0)
        break;
    printf("%s", data_buf);
    memset(data_buf, 0, sizeof(data_buf));
}
```

- 等待服务器完成的消息:

```
if (recv(ctrl_sock, ctrl_buf, sizeof(ctrl_buf), 0) < 0) {
    perror("client: error reading message from server\n");
    return -1;
}
```

五、实验总结

本实验基于 FTP 协议实现了一个文件传输协议的客户端，用于与上个实验的 FTP 服务器通信。

- 本部分的完整代码如下：

```
#include"ftclient.h"int

/**
* 接收服务器响应
* 错误返回 -1，正确返回状态码
*/
intread_reply
int      0
if                              sizeof      0    0

    "client: error reading message from server\n"
return-1

return

/**
* 打印响应信息
*/
voidprint_reply int
switch

case220
printf "220 Welcome, server ready.\n"
break
case221
printf "221 Goodbye!\n"
break
case226
printf "226 Closing data connection. Requested file action
successful.\n"
break
case550
printf "550 Requested action not taken. File unavailable.\n"
break


/**
* 解析命令行到结构体
```

```
*/
intftclient_read_command char    int     struct
memset          0  sizeof
memset          0  sizeof

printf "ftclient> "   // 输入提示符
      stdout
                    // 等待用户输入命令char      NULL
          " "
          NULL  " "

if      NULL
strncpy          strlen

if  strcmp     "list"     0
strcpy          "LIST"

elseif  strcmp     "get"     0
strcpy          "RETR"

elseif  strcmp     "quit"     0
strcpy          "QUIT"

elsereturn-1  // 不合法memset     0  400
strcpy                  // 存储命令到 buf 开始处/* 如果命令带有
参数，追加到 buf */if       NULL

strcat     " "
strncat          strlen

return0


/**
* 实现 get <filename> 命令行
*/
intftclient_get int          int          char
char
int
          "w"   // 创建并打开名字为 arg 的文件/* 将服务器
传来的数据（文件内容）写入本地建立的文件 */while
                0     0
          1
```

```
if  0
    "error\n"

return0

/**
* 打开数据连接
*/
intftclient_open_conn int
int

/* 在控制连接上发起一个 ACK 确认 */int  1
if  char  sizeof  0  0

printf "client: ack write error :%d\n"
exit 1

int
return

/**
* 实现 list 命令
*/
intftclient_list int  int
size_t
char
int  0

/* 等待服务器启动的信息 */if  sizeof  0
0

    "client: error reading message from server\n"
return-1

memset  0  sizeof

/* 接收服务器传来的数据 */while
  0  0

printf "%s"
```

```
memset  0  sizeof

if  0
    "error"

/* 等待服务器完成的消息 */if  sizeof  0
0

    "client: error reading message from server\n"
return-1

return0

/**
* 输入含有命令(code)和参数(arg)的 command(cmd) 结构
* 连接 code + arg,并放进一个字符串，然后发送给服务器
*/
intftclient_send_cmd struct
char
int

sprintf  "%s %s"

/* 发送命令字符串到服务器 */
                    int strlen  0
if    0

    "Error sending command to server"
return-1

return0

/**
* 获取登录信息
* 发送到服务器认证
*/
voidftclient_login
struct
char  256
memset  0  256

/* 获取用户名 */printf "Name: "
```

```
        stdout
    256

/* 发送用户名到服务器 */strcpy    "USER"
strcpy

/* 等待应答码 331 */int
                sizeof    0

/* 获得密码 */
    stdout
char        "Password: "

/* 发送密码到服务器 */strcpy    "PASS"
strcpy

/* 等待响应 */int
switch

case430
printf "Invalid username/password.\n"
exit 0
case230
printf "Successful login.\n"
break
default
    "error reading message from server"
exit 1
break


/* 主函数入口 */intmain int      char
int
char
struct
struct

/* 命令行参数合法性检测 */if      3

printf "usage: ./ftclient hostname port\n"
exit 0
```

```
char                    1   //所要连接的服务器主机名char                    2
//所要链接到服务器程序端口号/* 获得和服务器名匹配的地址
*/memset                0  sizeof struct
if          0
printf "getaddrinfo() error %s"
exit 1

/* 找到对应的服务器地址并连接 */for                        NULL
                // 创建控制套接字if                    0
continue

if                                                    0  // 和
服务器连接break

else
        "connecting stream socket"
exit 1

/* 连接成功，打印信息 */printf "Connected to %s.\n"

/* 获取用户的名字和密码 */

while  1
  // 循环，直到用户输入 quit/* 得到用户输入的命令 */if
                        sizeof                    0
printf "Invalid command\n"
continue  // 跳过本次循环，处理下一个命令
```

```
/* 发送命令到服务器 */if
 int strlen  0   0
exit 1

  //读取服务器响应（服务器是否可以支持该命令？）if  221  // 退出命令
  221
break

if  502
printf "%d Invalid command.\n"  // 不合法的输入，显示错误信息else
// 命令合法（RC = 200),处理命令/* 打开数据连接 */if
  0
  "Error opening socket for data connection"
exit 1

/* 执行命令 */if  strcmp  "LIST"  0

elseif  strcmp  "RETR"  0
if  550  // 等待回复
  550
continue


  // 循环得到更多的用户输入

  // 关闭套接字控制连接return0
```

六、源码下载及使用

- 在 http://labfile.oss.aliyuncs.com/courses/628/ftp.zip 下载源码。
- unzip ftp.zip
- 完成 ftserve 程序的编译和链接：
- 完成 ftclient 程序的编译和链接：
- 运行服务器：

```
$ server/ftserve PORTNO
```

`PORTNO` 是自定义的端口号。

- 运行客户端：

```
$ client/ftclient HOSTNAME PORTNO
```

`HOSTNAME` 是服务器主机名，在服务器主机上输入 `hostname` 命令可以查看主机名。

- 登陆：

```
Name
Password
```

- 支持的命令清单：

```
list
```

参考资料

- 《UNIX环境高级编程》
- https://github.com/beckysag/ftp