

用C语言实现FTP协议客户端的主要功能

发表于 2013-03-25 | 分类于 [code](#) | [31 条评论](#)

最近在研究FTP客户端的实现，最初我直接使用的Cocoa中提供的CFFTPStream相关的函数，但最终发现用此方法实现的FTP客户端有很大的局限性，于是我便找到了一份在Windows上用C语言实现的FTP客户端代码，但在Mac OSX系统下却编译不过，于是我便根据这份代码改写了一个份在类Unix上可以正常使用的FTP函数，下面帖上所有的代码：

7月19日更新，修复了其实多处BUG,完善了FTP的List,上传,下载,的接口。

头文件(THFTPAPI.h):

```
1 //
2 //  THFTPAPI.h
3 //  MyFTP
4 //
5 //  Created by TanHao on 13-6-6.
6 //  Copyright (c) 2013年 http://www.tanhao.me. All rights reserved.
7 //
8
9 //链接服务器
10 int ftp_connect( char *host, int port, char *user, char *pwd );
11 //断开服务器
12 int ftp_quit( int c_sock);
13
14 //设置表示类型
15 int ftp_type( int c_sock, char mode );
16
17 //改变工作目录
18 int ftp_cwd( int c_sock, char *path );
19 //回到上一层目录
20 int ftp_cdup( int c_sock );
21 //创建目录
22 int ftp_mkd( int c_sock, char *path );
23 //列表
24 int ftp_list( int c_sock, char *path, void **data, unsigned long long *data_len);
25
26 //下载文件
27 int ftp_retrfile( int c_sock, char *s, char *d ,unsigned long long *stor_size, int *stop
28 //上传文件
29 int ftp_storfile( int c_sock, char *s, char *d ,unsigned long long *stor_size, int *stop
30
31 //修改文件名&移动目录
32 int ftp_renamefile( int c_sock, char *s, char *d );
33 //删除文件
34 int ftp_deletefile( int c_sock, char *s );
35 //删除目录
36 int ftp_deletefolder( int c_sock, char *s );
```

实现(THFTPAPI.c):

```
1 //
2 // THFTPAPI.c
3 // MyFTP
4 //
5 // Created by TanHao on 13-6-6.
6 // Copyright (c) 2013年 http://www.tanhao.me. All rights reserved.
7 //
8
9 #include "THFTPAPI.h"
10
11 #include <stdio.h>
12 #include <sys/types.h>
13 #include <sys/stat.h>
14 #include <fcntl.h>
15 #include <sys/socket.h>
16 #include <netdb.h>
17 #include <stdio.h>
18 #include <ctype.h>
19 #include <stdlib.h>
20 #include <unistd.h>
21 #include <string.h>
22 #include <sys/ioctl.h>
23
24 //创建一个socket并返回
25 int socket_connect(char *host,int port)
26 {
27     struct sockaddr_in address;
28     int s, opvalue;
29     socklen_t slen;
30
31     opvalue = 8;
32     slen = sizeof(opvalue);
33     memset(&address, 0, sizeof(address));
34
35     if ((s = socket(AF_INET, SOCK_STREAM, 0)) < 0 ||
36         setsockopt(s, IPPROTO_IP, IP_TOS, &opvalue, slen) < 0)
37         return -1;
38
39     //设置接收和发送超时
40     struct timeval timeo = {15, 0};
41     setsockopt(s, SOL_SOCKET, SO_SNDTIMEO, &timeo, sizeof(timeo));
42     setsockopt(s, SOL_SOCKET, SO_RCVTIMEO, &timeo, sizeof(timeo));
43
44     address.sin_family = AF_INET;
45     address.sin_port = htons((unsigned short)port);
46
47     struct hostent* server = gethostbyname(host);
48     if (!server)
49         return -1;
50
51     memcpy(&address.sin_addr.s_addr, server->h_addr, server->h_length);
52
53     if (connect(s, (struct sockaddr*) &address, sizeof(address)) == -1)
54         return -1;
55
56     return s;
57 }
58
59 //连接到一个ftp的服务器, 返回socket
60 int connect_server( char *host, int port )
61 {
```

```

62     int         ctrl_sock;
63     char         buf[512];
64     int          result;
65     ssize_t      len;
66
67     ctrl_sock = socket_connect(host, port);
68     if (ctrl_sock == -1) {
69         return -1;
70     }
71
72     len = recv( ctrl_sock, buf, 512, 0 );
73     buf[len] = 0;
74     sscanf( buf, "%d", &result );
75     if ( result != 220 ) {
76         close( ctrl_sock );
77         return -1;
78     }
79
80     return ctrl_sock;
81 }
82
83 //发送命令,返回结果
84 int ftp_sendcmd_re( int sock, char *cmd, void *re_buf, ssize_t *len)
85 {
86     char         buf[512];
87     ssize_t      r_len;
88
89     if ( send( sock, cmd, strlen(cmd), 0 ) == -1 )
90         return -1;
91
92     r_len = recv( sock, buf, 512, 0 );
93     if ( r_len < 1 ) return -1;
94     buf[r_len] = 0;
95
96     if (len != NULL) *len = r_len;
97     if (re_buf != NULL) sprintf(re_buf, "%s", buf);
98
99     return 0;
100 }
101
102 //发送命令,返回编号
103 int ftp_sendcmd( int sock, char *cmd )
104 {
105     char         buf[512];
106     int          result;
107     ssize_t      len;
108
109     result = ftp_sendcmd_re(sock, cmd, buf, &len);
110     if (result == 0)
111     {
112         sscanf( buf, "%d", &result );
113     }
114
115     return result;
116 }
117
118 //登录ftp服务器
119 int login_server( int sock, char *user, char *pwd )
120 {
121     char         buf[128];
122     int          result;
123
124     sprintf( buf, "USER %s\r\n", user );
125     result = ftp_sendcmd( sock, buf );
126     if ( result == 230 ) return 0;

```

```

127     else if ( result == 331 ) {
128         sprintf( buf, "PASS %s\r\n", pwd );
129         if ( ftp_sendcmd( sock, buf ) != 230 ) return -1;
130         return 0;
131     }
132     else
133         return -1;
134 }
135
136 int create_datasock( int ctrl_sock )
137 {
138     int     lsn_sock;
139     int     port;
140     int     len;
141     struct  sockaddr_in sin;
142     char     cmd[128];
143
144     lsn_sock = socket( PF_INET, SOCK_STREAM, IPPROTO_TCP );
145     if ( lsn_sock == -1 ) return -1;
146     memset( (char *)&sin, 0, sizeof(sin) );
147     sin.sin_family = AF_INET;
148     if( bind(lsn_sock, (struct sockaddr *)&sin, sizeof(sin)) == -1 ) {
149         close( lsn_sock );
150         return -1;
151     }
152
153     if( listen(lsn_sock, 2) == -1 ) {
154         close( lsn_sock );
155         return -1;
156     }
157
158     len = sizeof( struct sockaddr );
159     if ( getsockname( lsn_sock, (struct sockaddr *)&sin, (socklen_t *)&len ) == -1 )
160     {
161         close( lsn_sock );
162         return -1;
163     }
164     port = sin.sin_port;
165
166     if( getsockname( ctrl_sock, (struct sockaddr *)&sin, (socklen_t *)&len ) == -1 )
167     {
168         close( lsn_sock );
169         return -1;
170     }
171
172     sprintf( cmd, "PORT %d,%d,%d,%d,%d,%d\r\n",
173             sin.sin_addr.s_addr&0x000000FF,
174             (sin.sin_addr.s_addr&0x0000FF00)>>8,
175             (sin.sin_addr.s_addr&0x00FF0000)>>16,
176             (sin.sin_addr.s_addr&0xFF000000)>>24,
177             port>>8, port&0xff );
178
179     if ( ftp_sendcmd( ctrl_sock, cmd ) != 200 ) {
180         close( lsn_sock );
181         return -1;
182     }
183     return lsn_sock;
184 }
185
186 //连接到PASV接口
187 int ftp_pasv_connect( int c_sock )
188 {
189     int     r_sock;
190     int     send_re;
191     ssize_t len;

```

```

192     int      addr[6];
193     char      buf[512];
194     char      re_buf[512];
195
196     //设置PASV被动模式
197     bzero(buf, sizeof(buf));
198     sprintf( buf, "PASV\r\n");
199     send_re = ftp_sendcmd_re( c_sock, buf, re_buf, &len);
200     if (send_re == 0) {
201         sscanf(re_buf, "%*[^](%d,%d,%d,%d,%d,%d)", &addr[0], &addr[1], &addr[2], &addr[3],
202     }
203
204     //连接PASV端口
205     bzero(buf, sizeof(buf));
206     sprintf( buf, "%d.%d.%d.%d", addr[0], addr[1], addr[2], addr[3]);
207     r_sock = socket_connect(buf, addr[4]*256+addr[5]);
208
209     return r_sock;
210 }
211
212 //表示类型
213 int ftp_type( int c_sock, char mode )
214 {
215     char      buf[128];
216     sprintf( buf, "TYPE %c\r\n", mode );
217     if ( ftp_sendcmd( c_sock, buf ) != 200 )
218         return -1;
219     else
220         return 0;
221 }
222
223 //改变工作目录
224 int ftp_cwd( int c_sock, char *path )
225 {
226     char      buf[128];
227     int      re;
228     sprintf( buf, "CWD %s\r\n", path );
229     re = ftp_sendcmd( c_sock, buf );
230     if ( re != 250 )
231         return -1;
232     else
233         return 0;
234 }
235
236 //回到上一层目录
237 int ftp_cdup( int c_sock )
238 {
239     int      re;
240     re = ftp_sendcmd( c_sock, "CDUP\r\n" );
241     if ( re != 250 )
242         return re;
243     else
244         return 0;
245 }
246
247 //创建目录
248 int ftp_mkd( int c_sock, char *path )
249 {
250     char      buf[512];
251     int      re;
252     sprintf( buf, "MKD %s\r\n", path );
253     re = ftp_sendcmd( c_sock, buf );
254     if ( re != 257 )
255         return re;
256     else

```

```

257         return 0;
258     }
259
260     //列表
261     int ftp_list( int c_sock, char *path, void **data, unsigned long long *data_len)
262     {
263         int      d_sock;
264         char      buf[512];
265         int      send_re;
266         int      result;
267         ssize_t   len,buf_len,total_len;
268
269         //连接到PASV接口
270         d_sock = ftp_pasv_connect(c_sock);
271         if (d_sock == -1) {
272             return -1;
273         }
274
275         //发送LIST命令
276         bzero(buf, sizeof(buf));
277         sprintf( buf, "LIST %s\r\n", path);
278         send_re = ftp_sendcmd( c_sock, buf );
279         if (send_re >= 300 || send_re == 0)
280             return send_re;
281
282         len=total_len = 0;
283         buf_len = 512;
284         void *re_buf = malloc(buf_len);
285         while ( (len = recv( d_sock, buf, 512, 0 )) > 0 )
286         {
287             if (total_len+len > buf_len)
288             {
289                 buf_len *= 2;
290                 void *re_buf_n = malloc(buf_len);
291                 memcpy(re_buf_n, re_buf, total_len);
292                 free(re_buf);
293                 re_buf = re_buf_n;
294             }
295             memcpy(re_buf+total_len, buf, len);
296             total_len += len;
297         }
298         close( d_sock );
299
300         //向服务器接收返回值
301         bzero(buf, sizeof(buf));
302         len = recv( c_sock, buf, 512, 0 );
303         buf[len] = 0;
304         sscanf( buf, "%d", &result );
305         if ( result != 226 )
306         {
307             free(re_buf);
308             return result;
309         }
310
311         *data = re_buf;
312         *data_len = total_len;
313
314         return 0;
315     }
316
317     //下载文件
318     int ftp_retrfile( int c_sock, char *s, char *d ,unsigned long long *stor_size, int *sto
319     {
320         int      d_sock;
321         ssize_t   len,write_len;

```

```

322     char    buf[512];
323     int     handle;
324     int     result;
325
326     //打开本地文件
327     handle = open( d, O_WRONLY|O_CREAT|O_TRUNC, S_IREAD|S_IWRITE );
328     if ( handle == -1 ) return -1;
329
330     //设置传输模式
331     ftp_type(c_sock, 'I');
332
333     //连接到PASV接口
334     d_sock = ftp_pasv_connect(c_sock);
335     if (d_sock == -1)
336     {
337         close(handle);
338         return -1;
339     }
340
341     //发送STOR命令
342     bzero(buf, sizeof(buf));
343     sprintf( buf, "RETR %s\r\n", s );
344     result = ftp_sendcmd( c_sock, buf );
345     if (result >= 300 || result == 0)
346     {
347         close(handle);
348         return result;
349     }
350
351     //开始向PASV读取数据
352     bzero(buf, sizeof(buf));
353     while ( (len = recv( d_sock, buf, 512, 0 )) > 0 ) {
354         write_len = write( handle, buf, len );
355         if (write_len != len || (stop != NULL && *stop))
356         {
357             close( d_sock );
358             close( handle );
359             return -1;
360         }
361
362         if (stor_size != NULL)
363         {
364             *stor_size += write_len;
365         }
366     }
367     close( d_sock );
368     close( handle );
369
370     //向服务器接收返回值
371     bzero(buf, sizeof(buf));
372     len = recv( c_sock, buf, 512, 0 );
373     buf[len] = 0;
374     sscanf( buf, "%d", &result );
375     if ( result >= 300 ) {
376         return result;
377     }
378     return 0;
379 }
380
381 //上传文件
382 int ftp_storfile( int c_sock, char *s, char *d ,unsigned long long *stor_size, int *sto
383 {
384     int     d_sock;
385     ssize_t len,send_len;
386     char    buf[512];

```

```

387     int     handle;
388     int send_re;
389     int result;
390
391     //打开本地文件
392     handle = open( s, O_RDONLY);
393     if ( handle == -1 ) return -1;
394
395     //设置传输模式
396     ftp_type(c_sock, 'I');
397
398     //连接到PASV接口
399     d_sock = ftp_pasv_connect(c_sock);
400     if (d_sock == -1)
401     {
402         close(handle);
403         return -1;
404     }
405
406     //发送STOR命令
407     bzero(buf, sizeof(buf));
408     sprintf( buf, "STOR %s\r\n", d );
409     send_re = ftp_sendcmd( c_sock, buf );
410     if (send_re >= 300 || send_re == 0)
411     {
412         close(handle);
413         return send_re;
414     }
415
416     //开始向PASV通道写数据
417     bzero(buf, sizeof(buf));
418     while ( (len = read( handle, buf, 512)) > 0)
419     {
420         send_len = send(d_sock, buf, len, 0);
421         if (send_len != len ||
422             (stop != NULL && *stop))
423         {
424             close( d_sock );
425             close( handle );
426             return -1;
427         }
428
429         if (stor_size != NULL)
430         {
431             *stor_size += send_len;
432         }
433     }
434     close( d_sock );
435     close( handle );
436
437     //向服务器接收返回值
438     bzero(buf, sizeof(buf));
439     len = recv( c_sock, buf, 512, 0 );
440     buf[len] = 0;
441     sscanf( buf, "%d", &result );
442     if ( result >= 300 ) {
443         return result;
444     }
445     return 0;
446 }
447
448 //修改文件名&移动目录
449 int ftp_renamefile( int c_sock, char *s, char *d )
450 {
451     char     buf[512];

```



```

452     int     re;
453
454     sprintf( buf, "RNFR %s\r\n", s );
455     re = ftp_sendcmd( c_sock, buf );
456     if ( re != 350 ) return re;
457     sprintf( buf, "RNT0 %s\r\n", d );
458     re = ftp_sendcmd( c_sock, buf );
459     if ( re != 250 ) return re;
460     return 0;
461 }
462
463 //删除文件
464 int ftp_deletefile( int c_sock, char *s )
465 {
466     char     buf[512];
467     int      re;
468
469     sprintf( buf, "DELE %s\r\n", s );
470     re = ftp_sendcmd( c_sock, buf );
471     if ( re != 250 ) return re;
472     return 0;
473 }
474
475 //删除目录
476 int ftp_deletefolder( int c_sock, char *s )
477 {
478     char     buf[512];
479     int      re;
480
481     sprintf( buf, "RMD %s\r\n", s );
482     re = ftp_sendcmd( c_sock, buf );
483     if ( re != 250 ) return re;
484     return 0;
485 }
486
487 //链接服务器
488 int ftp_connect( char *host, int port, char *user, char *pwd )
489 {
490     int      c_sock;
491     c_sock = connect_server( host, port );
492     if ( c_sock == -1 ) return -1;
493     if ( login_server( c_sock, user, pwd ) == -1 ) {
494         close( c_sock );
495         return -1;
496     }
497     return c_sock;
498 }
499
500 //断开服务器
501 int ftp_quit( int c_sock )
502 {
503     int re = 0;
504     re = ftp_sendcmd( c_sock, "QUIT\r\n" );
505     close( c_sock );
506     return re;
507 }

```

下载该代码文件:[THFTPAPI](#)

#C

#FTP

#OSX

#Unix

#Windows

#代码

#源码

31 条评论

老谭笔记

1 登录 ▾

♥ Recommend

🔗 分享

按评分高低排序 ▾



加入讨论...



Darren • 4个月前

首先谢谢老谭分享。

我想问一下，用您API中的ftp_retrfile下载文件，txt和bmp是对的，但是下载doc或者是pdf文件就是错的。

^ | ▾ • 回复 • 分享 ▾



Lanhao_Cheng → Darren • 4个月前

另外用write会好一些

^ | ▾ • 回复 • 分享 ▾



Lanhao_Cheng → Darren • 4个月前

ASCII或者Binary

^ | ▾ • 回复 • 分享 ▾



Lanhao_Cheng • 5个月前

老谭你好，请问如何实现主动模式呢

^ | ▾ • 回复 • 分享 ▾



韩旭 • 6个月前

老谭 你的.H文件可以发给我看一下

^ | ▾ • 回复 • 分享 ▾



老谭 → 韩旭 • 6个月前

文章后面不是有附件下载么

^ | ▾ • 回复 • 分享 ▾




瘞鹿鸣佐 • 8个月前

请问ftp_list函数的参数列表void **data, unsigned long long *data_len分别指的是什么，我是想用ftp_list函数遍历目标目录下的文件夹，并返回目录下的所有文件夹和文件的名字。

^ | ▾ • 回复 • 分享 ▾



老谭 → 瘞鹿鸣佐 • 8个月前

 这个data就是返回列表的信息,你可以输出来看一下,具体每一项就需要你自己写代码来解析分割这个data了

^ | v • 回复 • 分享



瘞鹿鸣佐 → 老谭 • 8个月前

调用ftp_list函数参数列表的void **data怎样使用呢,给data传递什么参数呢

^ | v • 回复 • 分享



Leon • 8个月前

你好,想问下上传下载文件函数中stor_size和stop这个输入参数是什么意思,怎么用啊?

^ | v • 回复 • 分享



老谭 → Leon • 8个月前

stor_size是指已经上传/下载的大小,而stop是用来外部中断当前这个上传/下载任务.因为这个上传/下载的接口是阻塞的,所以可以用另一个线程通过stor_size来查询进度,或通过设置stop来中断任务

^ | v • 回复 • 分享



逍遥de呆子 • 8个月前

老谭,不可将使用方法的源码贴上来啊?

^ | v • 回复 • 分享



老谭 → 逍遥de呆子 • 8个月前

使用就太简单了,通过ftp_connect连接服务返回socket的描述符,然后通过这个描述符,执行list,mkdir,up,down等所有操作即可

^ | v • 回复 • 分享



围脖杨杨 • 8个月前

您好,跟您咨询下,connect_server函数中recv函数接收的缓冲区里总接收到乱码而不是期望的220.我的环境是AIX 接收的6位的字符串,打印ascii码如下: [255][254][37][255][253][24]

^ | v • 回复 • 分享



围脖杨杨 → 围脖杨杨 • 8个月前

额, telnet太多了, 参数端口传错了。。。。

^ | v • 回复 • 分享



myz1104 • 1年前

老谭,有没有多线程的版本?

^ | v • 回复 • 分享



老谭 → myz1104 • 1年前

没有哦,不过可以在外部套个线程

^ | v • 回复 • 分享



宁伟 • 1年前

你好,问一下ftp有实现文件夹上传的指令吗

^ | v • 回复 • 分享



老谭 → 宁伟 • 1年前

文件夹的上传其实还是创建文件夹+上传文件完成的

^ | v • 回复 • 分享



宁伟 → 老谭 • 1年前

难道服务器端有创建文件夹的功能??

^ | v • 回复 • 分享



caocoa • 2年前

老谭，发现一个bug，ftp_sendcmd_re函数的send之后，应该一直receive，遇到receive读取512个字节一次后，还有部分等到下一次receive函数才解释道，我暂时改成receive读取8192个字节[可怜]

^ | v • 回复 • 分享



老谭 → caocoa • 2年前

恩，确实这儿不是很严谨，如果recv返回的值等于了512，应该循环recv，直到返回的值小于512才能证明已经接收结束。我当时是看了协议，这个方法只用于发送普通的命令，返回值似乎都没有超过512字节的。

^ | v • 回复 • 分享



caocoa → 老谭 • 2年前

[衰]老谭，bug基本上都是recv没有结束，导致下次通信的时候紊乱了,期待你的改进版，谢了

^ | v • 回复 • 分享



caocoa → 老谭 • 2年前

额。。这个返回值主要看服务器那边了。。。我试着循环recv，但是不知道re_buf怎么一直不断添加接受的buf数据。。我c语言太菜了。不知道怎么改了。re_buf是存接受的数据吧

while (YES)

{

r_len = recv(sock, buf, 512, 0);

if (r_len < 1) return -1;

buf[r_len] = 0;

if (len != NULL) *len = r_len;

if (re_buf != NULL) sprintf(re_buf, "%s", buf);

}[衰]

^ | v • 回复 • 分享



caocoa • 2年前

老谭，发现一个bug，ftp_sendcmd_re函数的send之后，应该一直receive，遇到receive读取512个字节一次后，还有部分等到下一次receive函数才解释道，我暂时改成receive读取8192个字节

^ | v • 回复 • 分享



caococa • 2年前

运行到 `d_sock = accept(l_sock, &sin, (socklen_t *)&len);` 程序自动退出。。晕死

^ | v • 回复 • 分享 ›



caococa → caococa • 2年前

谭老师。。。求指点。。。怎么上传文件到FTP啊

^ | v • 回复 • 分享 ›



老谭 → caococa • 2年前

之前上传的代码有很多BUG，并且功能还不完善，我现在都已经修复了，并且添加了列表、下载的代码，你可以试试哈~~

^ | v • 回复 • 分享 ›

由 [Hexo](#) 强力驱动 | 主题 - [NexT.Mist](#)

© 2011 - 2015 ♥ TanHao 蜀ICP备12004580号

友情链接: [威言威语](#) [路路库](#) [FourFire](#) [MacCocoa](#) [CocoaChina](#)