**[老谭笔记](http://www.tanhao.me/)**窗体顶端

窗体底端

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](http://www.tanhao.me/categories/code/)   |   [31 条评论](http://www.tanhao.me/code/1312.html/#comments)

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 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 | // // THFTPAPI.h // MyFTP // // Created by TanHao on 13-6-6. // Copyright (c) 2013年 http://www.tanhao.me. All rights reserved. //  //链接服务器 int ftp\_connect( char \*host, int port, char \*user, char \*pwd ); //断开服务器 int ftp\_quit( int c\_sock);  //设置表示类型 int ftp\_type( int c\_sock, char mode );  //改变工作目录 int ftp\_cwd( int c\_sock, char \*path ); //回到上一层目录 int ftp\_cdup( int c\_sock ); //创建目录 int ftp\_mkd( int c\_sock, char \*path ); //列表 int ftp\_list( int c\_sock, char \*path, void \*\*data, unsigned long long \*data\_len);  //下载文件 int ftp\_retrfile( int c\_sock, char \*s, char \*d ,unsigned long long \*stor\_size, int \*stop); //上传文件 int ftp\_storfile( int c\_sock, char \*s, char \*d ,unsigned long long \*stor\_size, int \*stop);  //修改文件名&移动目录 int ftp\_renamefile( int c\_sock, char \*s, char \*d ); //删除文件 int ftp\_deletefile( int c\_sock, char \*s ); //删除目录 int ftp\_deletefolder( int c\_sock, char \*s ); |

实现(THFTPAPI.c):

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 3666 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 | // // THFTPAPI.c // MyFTP // // Created by TanHao on 13-6-6. // Copyright (c) 2013年 http://www.tanhao.me. All rights reserved. //  #include "THFTPAPI.h"  #include <stdio.h> #include <sys/types.h> #include <sys/stat.h> #include <fcntl.h> #include <sys/socket.h> #include <netdb.h> #include <stdio.h> #include <ctype.h> #include <stdlib.h> #include <unistd.h> #include <string.h> #include <sys/ioctl.h>  //创建一个socket并返回 int socket\_connect(char \*host,int port) {  struct sockaddr\_in address;  int s, opvalue;  socklen\_t slen;    opvalue = 8;  slen = sizeof(opvalue);  memset(&address, 0, sizeof(address));    if ((s = socket(AF\_INET, SOCK\_STREAM, 0)) < 0 ||  setsockopt(s, IPPROTO\_IP, IP\_TOS, &opvalue, slen) < 0)  return -1;    //设置接收和发送超时  struct timeval timeo = {15, 0};  setsockopt(s, SOL\_SOCKET, SO\_SNDTIMEO, &timeo, sizeof(timeo));  setsockopt(s, SOL\_SOCKET, SO\_RCVTIMEO, &timeo, sizeof(timeo));    address.sin\_family = AF\_INET;  address.sin\_port = htons((unsigned short)port);    struct hostent\* server = gethostbyname(host);  if (!server)  return -1;    memcpy(&address.sin\_addr.s\_addr, server->h\_addr, server->h\_length);    if (connect(s, (struct sockaddr\*) &address, sizeof(address)) == -1)  return -1;    return s; }  //连接到一个ftp的服务器，返回socket int connect\_server( char \*host, int port ) {   int ctrl\_sock;  char buf[512];  int result;  ssize\_t len;    ctrl\_sock = socket\_connect(host, port);  if (ctrl\_sock == -1) {  return -1;  }    len = recv( ctrl\_sock, buf, 512, 0 );  buf[len] = 0;  sscanf( buf, "%d", &result );  if ( result != 220 ) {  close( ctrl\_sock );  return -1;  }    return ctrl\_sock; }  //发送命令,返回结果 int ftp\_sendcmd\_re( int sock, char \*cmd, void \*re\_buf, ssize\_t \*len) {  char buf[512];  ssize\_t r\_len;    if ( send( sock, cmd, strlen(cmd), 0 ) == -1 )  return -1;    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);    return 0; }  //发送命令,返回编号 int ftp\_sendcmd( int sock, char \*cmd ) {  char buf[512];  int result;  ssize\_t len;    result = ftp\_sendcmd\_re(sock, cmd, buf, &len);  if (result == 0)  {  sscanf( buf, "%d", &result );  }    return result; }  //登录ftp服务器 int login\_server( int sock, char \*user, char \*pwd ) {  char buf[128];  int result;    sprintf( buf, "USER %s\r\n", user );  result = ftp\_sendcmd( sock, buf );  if ( result == 230 ) return 0;  else if ( result == 331 ) {  sprintf( buf, "PASS %s\r\n", pwd );  if ( ftp\_sendcmd( sock, buf ) != 230 ) return -1;  return 0;  }  else  return -1; }  int create\_datasock( int ctrl\_sock ) {  int lsn\_sock;  int port;  int len;  struct sockaddr\_in sin;  char cmd[128];    lsn\_sock = socket( PF\_INET, SOCK\_STREAM, IPPROTO\_TCP );  if ( lsn\_sock == -1 ) return -1;  memset( (char \*)&sin, 0, sizeof(sin) );  sin.sin\_family = AF\_INET;  if( bind(lsn\_sock, (struct sockaddr \*)&sin, sizeof(sin)) == -1 ) {  close( lsn\_sock );  return -1;  }    if( listen(lsn\_sock, 2) == -1 ) {  close( lsn\_sock );  return -1;  }    len = sizeof( struct sockaddr );  if ( getsockname( lsn\_sock, (struct sockaddr \*)&sin, (socklen\_t \*)&len ) == -1 )  {  close( lsn\_sock );  return -1;  }  port = sin.sin\_port;    if( getsockname( ctrl\_sock, (struct sockaddr \*)&sin, (socklen\_t \*)&len ) == -1 )  {  close( lsn\_sock );  return -1;  }    sprintf( cmd, "PORT %d,%d,%d,%d,%d,%d\r\n",  sin.sin\_addr.s\_addr&0x000000FF,  (sin.sin\_addr.s\_addr&0x0000FF00)>>8,  (sin.sin\_addr.s\_addr&0x00FF0000)>>16,  (sin.sin\_addr.s\_addr&0xFF000000)>>24,  port>>8, port&0xff );    if ( ftp\_sendcmd( ctrl\_sock, cmd ) != 200 ) {  close( lsn\_sock );  return -1;  }  return lsn\_sock; }  //连接到PASV接口 int ftp\_pasv\_connect( int c\_sock ) {  int r\_sock;  int send\_re;  ssize\_t len;  int addr[6];  char buf[512];  char re\_buf[512];    //设置PASV被动模式  bzero(buf, sizeof(buf));  sprintf( buf, "PASV\r\n");  send\_re = ftp\_sendcmd\_re( c\_sock, buf, re\_buf, &len);  if (send\_re == 0) {  sscanf(re\_buf, "%\*[^(](%d,%d,%d,%d,%d,%d)",&addr[0],&addr[1],&addr[2],&addr[3],&addr[4],&addr[5]);  }    //连接PASV端口  bzero(buf, sizeof(buf));  sprintf( buf, "%d.%d.%d.%d",addr[0],addr[1],addr[2],addr[3]);  r\_sock = socket\_connect(buf,addr[4]\*256+addr[5]);    return r\_sock; }  //表示类型 int ftp\_type( int c\_sock, char mode ) {  char buf[128];  sprintf( buf, "TYPE %c\r\n", mode );  if ( ftp\_sendcmd( c\_sock, buf ) != 200 )  return -1;  else  return 0; }  //改变工作目录 int ftp\_cwd( int c\_sock, char \*path ) {  char buf[128];  int re;  sprintf( buf, "CWD %s\r\n", path );  re = ftp\_sendcmd( c\_sock, buf );  if ( re != 250 )  return -1;  else  return 0; }  //回到上一层目录 int ftp\_cdup( int c\_sock ) {  int re;  re = ftp\_sendcmd( c\_sock, "CDUP\r\n" );  if ( re != 250 )  return re;  else  return 0; }  //创建目录 int ftp\_mkd( int c\_sock, char \*path ) {  char buf[512];  int re;  sprintf( buf, "MKD %s\r\n", path );  re = ftp\_sendcmd( c\_sock, buf );  if ( re != 257 )  return re;  else  return 0; }  //列表 int ftp\_list( int c\_sock, char \*path, void \*\*data, unsigned long long \*data\_len) {  int d\_sock;  char buf[512];  int send\_re;  int result;  ssize\_t len,buf\_len,total\_len;    //连接到PASV接口  d\_sock = ftp\_pasv\_connect(c\_sock);  if (d\_sock == -1) {  return -1;  }    //发送LIST命令  bzero(buf, sizeof(buf));  sprintf( buf, "LIST %s\r\n", path);  send\_re = ftp\_sendcmd( c\_sock, buf );  if (send\_re >= 300 || send\_re == 0)  return send\_re;    len=total\_len = 0;  buf\_len = 512;  void \*re\_buf = malloc(buf\_len);  while ( (len = recv( d\_sock, buf, 512, 0 )) > 0 )  {  if (total\_len+len > buf\_len)  {  buf\_len \*= 2;  void \*re\_buf\_n = malloc(buf\_len);  memcpy(re\_buf\_n, re\_buf, total\_len);  free(re\_buf);  re\_buf = re\_buf\_n;  }  memcpy(re\_buf+total\_len, buf, len);  total\_len += len;  }  close( d\_sock );    //向服务器接收返回值  bzero(buf, sizeof(buf));  len = recv( c\_sock, buf, 512, 0 );  buf[len] = 0;  sscanf( buf, "%d", &result );  if ( result != 226 )  {  free(re\_buf);  return result;  }    \*data = re\_buf;  \*data\_len = total\_len;    return 0; }  //下载文件 int ftp\_retrfile( int c\_sock, char \*s, char \*d ,unsigned long long \*stor\_size, int \*stop) {  int d\_sock;  ssize\_t len,write\_len;  char buf[512];  int handle;  int result;    //打开本地文件  handle = open( d, O\_WRONLY|O\_CREAT|O\_TRUNC, S\_IREAD|S\_IWRITE );  if ( handle == -1 ) return -1;    //设置传输模式  ftp\_type(c\_sock, 'I');    //连接到PASV接口  d\_sock = ftp\_pasv\_connect(c\_sock);  if (d\_sock == -1)  {  close(handle);  return -1;  }    //发送STOR命令  bzero(buf, sizeof(buf));  sprintf( buf, "RETR %s\r\n", s );  result = ftp\_sendcmd( c\_sock, buf );  if (result >= 300 || result == 0)  {  close(handle);  return result;  }    //开始向PASV读取数据  bzero(buf, sizeof(buf));  while ( (len = recv( d\_sock, buf, 512, 0 )) > 0 ) {  write\_len = write( handle, buf, len );  if (write\_len != len || (stop != NULL && \*stop))  {  close( d\_sock );  close( handle );  return -1;  }    if (stor\_size != NULL)  {  \*stor\_size += write\_len;  }  }  close( d\_sock );  close( handle );    //向服务器接收返回值  bzero(buf, sizeof(buf));  len = recv( c\_sock, buf, 512, 0 );  buf[len] = 0;  sscanf( buf, "%d", &result );  if ( result >= 300 ) {  return result;  }  return 0; }  //上传文件 int ftp\_storfile( int c\_sock, char \*s, char \*d ,unsigned long long \*stor\_size, int \*stop) {  int d\_sock;  ssize\_t len,send\_len;  char buf[512];  int handle;  int send\_re;  int result;    //打开本地文件  handle = open( s, O\_RDONLY);  if ( handle == -1 ) return -1;    //设置传输模式  ftp\_type(c\_sock, 'I');    //连接到PASV接口  d\_sock = ftp\_pasv\_connect(c\_sock);  if (d\_sock == -1)  {  close(handle);  return -1;  }    //发送STOR命令  bzero(buf, sizeof(buf));  sprintf( buf, "STOR %s\r\n", d );  send\_re = ftp\_sendcmd( c\_sock, buf );  if (send\_re >= 300 || send\_re == 0)  {  close(handle);  return send\_re;  }    //开始向PASV通道写数据  bzero(buf, sizeof(buf));  while ( (len = read( handle, buf, 512)) > 0)  {  send\_len = send(d\_sock, buf, len, 0);  if (send\_len != len ||  (stop != NULL && \*stop))  {  close( d\_sock );  close( handle );  return -1;  }    if (stor\_size != NULL)  {  \*stor\_size += send\_len;  }  }  close( d\_sock );  close( handle );    //向服务器接收返回值  bzero(buf, sizeof(buf));  len = recv( c\_sock, buf, 512, 0 );  buf[len] = 0;  sscanf( buf, "%d", &result );  if ( result >= 300 ) {  return result;  }  return 0; }  //修改文件名&移动目录 int ftp\_renamefile( int c\_sock, char \*s, char \*d ) {  char buf[512];  int re;    sprintf( buf, "RNFR %s\r\n", s );  re = ftp\_sendcmd( c\_sock, buf );  if ( re != 350 ) return re;  sprintf( buf, "RNTO %s\r\n", d );  re = ftp\_sendcmd( c\_sock, buf );  if ( re != 250 ) return re;  return 0; }  //删除文件 int ftp\_deletefile( int c\_sock, char \*s ) {  char buf[512];  int re;    sprintf( buf, "DELE %s\r\n", s );  re = ftp\_sendcmd( c\_sock, buf );  if ( re != 250 ) return re;  return 0; }  //删除目录 int ftp\_deletefolder( int c\_sock, char \*s ) {  char buf[512];  int re;    sprintf( buf, "RMD %s\r\n", s );  re = ftp\_sendcmd( c\_sock, buf );  if ( re != 250 ) return re;  return 0; }  //链接服务器 int ftp\_connect( char \*host, int port, char \*user, char \*pwd ) {  int c\_sock;  c\_sock = connect\_server( host, port );  if ( c\_sock == -1 ) return -1;  if ( login\_server( c\_sock, user, pwd ) == -1 ) {  close( c\_sock );  return -1;  }  return c\_sock; }  //断开服务器 int ftp\_quit( int c\_sock) {  int re = 0;  re = ftp\_sendcmd( c\_sock, "QUIT\r\n" );  close( c\_sock );  return re; } |

下载该代码文件:[THFTPAPI](http://storage.tanhao.me/2013/03/THFTPAPI.zip)

[#C](http://www.tanhao.me/tags/C/) [#FTP](http://www.tanhao.me/tags/FTP/) [#OSX](http://www.tanhao.me/tags/OSX/) [#Unix](http://www.tanhao.me/tags/Unix/) [#Windows](http://www.tanhao.me/tags/Windows/) [#代码](http://www.tanhao.me/tags/%E4%BB%A3%E7%A0%81/) [#源码](http://www.tanhao.me/tags/%E6%BA%90%E7%A0%81/)

[推荐两个免费的云计算平台:phpCloud和OpenShift](http://www.tanhao.me/talk/1323.html/)

[改键大师1.1.0小幅更新](http://www.tanhao.me/project/1308.html/)

* 文章目录
* 站点概览

由 [Hexo](http://hexo.io/" \t "_blank) 强力驱动

主题 - [NexT.Mist](https://github.com/iissnan/hexo-theme-next" \t "_blank)

©   2011 - 2015  TanHao   蜀ICP备12004580号

友情链接: [威言威语](http://www.weisay.com/blog/)  [路路库](http://www.llku.com/)  [FourFire](https://www.fourfire.cc/" \t "_blank)  [MacCocoa](http://maccocoa.com/" \t "_blank)  [CocoaChina](http://bbs.cocoachina.com/" \t "_blank)