06/02/19 08:49:27 C:\Users\rfbie\Desktop\FTP-Client\大作业\ftpcli.c

```c
/* header files */
#include <stdio.h>
#include <stdlib.h>
#include <netdb.h> /* getservbyname(), gethostbyname() */
#include <errno.h> /* for definition of errno */
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>
#include <sys/ioctl.h>
#include <sys/fcntl.h>
#include <termios.h>  /* Used to not display password directly, instead using *****   */
#include<sys/time.h> // used to limit the transmission rate, get current time
//int gettimeofday(struct  timeval*tv,struct  timezone *tz )
//int nanosleep(const struct timespec *req, struct timespec *rem);
/*
struct  timeval{
       long  tv_sec;
       long  tv_usec;
};

struct  timezone{
  int tz_minuteswest;
  int tz_dsttime;
};
*/

#define ECHOFLAGS (ECHO | ECHOE | ECHOK | ECHONL)


/* define macros*/
#define MAXBUF             1024
#define STDIN_FILENO     0
#define STDOUT_FILENO     1

/* define FTP reply code */
#define USERNAME     220
#define PASSWORD      331
#define LOGIN           230
#define PATHNAME      257
#define CLOSEDATA     226
#define ACTIONOK      250


/* DefinE global variables */
char *rbuf,*rbuf1,*wbuf,*wbuf1;           /* pointer that is malloc'ed */
char filename[100];
char newfilename[100];
char tmp[100];
char dirname[100];
char *host;                             /* hostname or dotted-decimal string */
struct sockaddr_in servaddr;


//int mygetch();
//int getpasswd(char *passwd, int size);
int set_disp_mode(int fd,int option);
int cliopen(char *host,int port);
int strtosrv(char *str);
void  ftp_list(int sockfd);
int ftp_get(int sck,char *pDownloadFileName);
int ftp_put(int sck,char *pUploadFileName_s);
void cmd_tcp(int sockfd);

//函数set_disp_mode用于控制是否开启输入回显功能
//如果option为0，则关闭回显，为1则打开回显
int set_disp_mode(int fd,int option)
{
   int err;
   struct termios term;
   if(tcgetattr(fd,&term)==-1){
     perror("Cannot get the attribution of the terminal");
     return 1;
   }
```

```
76      if(option)
77           term.c_lflag|=ECHOFLAGS;
78      else
79           term.c_lflag &=~ECHOFLAGS;
80      err=tcsetattr(fd,TCSAFLUSH,&term);
81      if(err==-1 && err==EINTR){
82           perror("Cannot set the attribution of the terminal");
83           return 1;
84      }
85      return 0;
86  }
87
88  int main(int argc,char *argv[])
89  {
90      int fd;
91      // Judge whether missing parameters
92      if(0 != argc -2)
93      {
94          printf("%s\n","missing <hostname>");
95          exit(0);
96      }
97
98      //define host and port
99      host = argv[1];
100     int port = 21;
101
102
103     /*****************************************************************
104     //1. code here: Allocate the read and write buffers before open().
105     // memory allocation for the read and writer buffers for maximum size
106     *****************************************************************/
107     rbuf = (char *)malloc(MAXBUF*sizeof(char));
108     rbuf1 = (char *)malloc(MAXBUF*sizeof(char));
109     wbuf = (char *)malloc(MAXBUF*sizeof(char));
110     wbuf1 = (char *)malloc(MAXBUF*sizeof(char));
111
112     //Initiate the socket to communication
113     fd = cliopen(host,port);
114     cmd_tcp(fd);
115     exit(0);
116 }
117
118 /* 如果不用替换* 暂时不用这个
119 int mygetch()
120 {
121     struct termios oldt, newt;
122     int ch;
123     tcgetattr(STDIN_FILENO, &oldt);
124     newt = oldt;
125     newt.c_lflag &= ~(ICANON | ECHO);
126     tcsetattr(STDIN_FILENO, TCSANOW, &newt);
127     ch = getchar();
128     tcsetattr(STDIN_FILENO, TCSANOW, &oldt);
129     return ch;
130 }
131
132 // password to * method
133 int getpasswd(char *passwd, int size)
134 {
135     int c, n = 0;
136     do
137     {
138         c = mygetch();
139         if (c != '\n' && c != 'r' && c != 127)
140         {
141             passwd[n] = c;
142             printf("*");
143             n++;
144         }
145         else if ((c != '\n' | c != '\r') && c == 127)//判断是否是回车或则退格
146         {
147             if (n > 0)
148             {
149                 n--;
150                 printf("\b \b");//输出退格
151             }
152         }
```

```
153     }while (c != '\n' && c != '\r' && n < (size - 1));
154     passwd[n] = '\0';//消除一个多余的回车
155     return n;
156 }
157 */
158 
159 /* Establish a TCP connection from client to server */
160 int cliopen(char *host,int port)
161 {
162     //control socket, in corresponding to data socket
163     int control_sock;
164     struct hostent *ht = NULL;
165 
166     control_sock = socket(AF_INET,SOCK_STREAM,0);
167     if(control_sock < 0)
168     {
169        printf("socket error\n");
170        return -1;
171     }
172 
173     //将IP地址进行转换，变为FTP地址结构体
174     ht = gethostbyname(host);
175     if(!ht)
176     {
177         return -1;
178     }
179 
180    //connect to server, return socket
181     memset(&servaddr,0,sizeof(struct sockaddr_in));
182     memcpy(&servaddr.sin_addr.s_addr,ht->h_addr,ht->h_length);
183     servaddr.sin_family = AF_INET;
184     servaddr.sin_port = htons(port);
185 
186     if(connect(control_sock,(struct sockaddr*)&servaddr,sizeof(struct sockaddr)) == -1)
187     {
188         return -1;
189     }
190     return control_sock;
191 }
192 
193 //匹配下载的文件名
194 int s(char *str,char *s2)
195 {
196     //char s1[100];
197 
198     return sscanf(str," get %s",s2) == 1;
199 
200 }
201 
202 //匹配上传的文件名
203 int st(char *str,char *s1)
204 {
205     return sscanf(str," put %s",s1) == 1;
206 }
207 
208 
209 /*
210    Compute server's port by a pair of integers and store it in char *port
211    Get server's IP address and store it in char *host
212 */
213 int strtosrv(char *str)
214 {
215   /*************************************************************
216   //3. code here to compute the port number for data connection
217   EG:10,3,255,85,192,181  192*256+181 = 49333
218   *************************************************************/
219    int addr[6];
220    //divide the string in to different parts
221    sscanf(str,"%*[^(](%d,%d,%d,%d,%d,%d)",&addr[0],&addr[1],&addr[2],&addr[3],&addr[4],&addr[5]);
222    //clear the host
223    bzero(host,strlen(host));
224    sprintf(host,"%d.%d.%d.%d",addr[0],addr[1],addr[2],addr[3]);
225    int port = addr[4]*256 + addr[5];
226    return port;
227 }
228 
229 /* Read and write as data transfer connection */
```

```
230 void ftp_list(int sockfd)
231 {
232     int nread;
233     for( ; ; )
234     {
235         /* data to read from socket */
236         if((nread = recv(sockfd,rbuf1,MAXBUF,0)) < 0)
237         {
238             printf("recv error\n");
239         }
240         else if(nread == 0)
241         {
242             //printf("over\n");
243             break;
244         }
245         if(write(STDOUT_FILENO,rbuf1,nread) != nread)
246             printf("send error to stdout\n");
247         /*else
248             printf("read something\n");*/
249     }
250     if(close(sockfd) < 0)
251         printf("close error\n");
252 }
253
254 /*
255 void checkSpeed(){
256   int stopTime = 1;
257   if (curSpeed>SPEEDLIMIT){
258
259   }
260 }
261 */
262
263 /* download file from ftp server */
264 int ftp_get(int sck,char *pDownloadFileName)
265 {
266    int handle = open(pDownloadFileName,O_WRONLY | O_CREAT | O_TRUNC, S_IREAD| S_IWRITE);
267    int nread;
268    printf("%d\n",handle);
269    /*if(handle == -1)
270        return -1;*/
271    struct timeval start, end;
272
273    for(;;)
274    {
275
276        //checkSpeed
277        //gettimeofday( &start, NULL );
278
279        if((nread = recv(sck,rbuf1,MAXBUF,0)) < 0)
280        {
281           printf("receive error\n");
282        }
283        else if(nread == 0)
284        {
285           printf("over\n");
286           break;
287        }
288        //Limiting speed function
289        /*
290        printf("nread is %d\n",nread);//1024,760
291
292        gettimeofday( &end, NULL );
293        float timeuse = 1000000 * ( end.tv_sec - start.tv_sec ) + end.tv_usec - start.tv_usec;
294
295        //MB/s
296        float speed = nread / timeuse;
297        printf("current speed is %.2f" ,speed );
298
299        */
300     //   printf("%s\n",rbuf1);
301        if(write(handle,rbuf1,nread) != nread)
302            printf("receive error from server!");
303
304       /*  Not print file content to console
305        if(write(STDOUT_FILENO,rbuf1,nread) != nread)
306            printf("receive error from server!");
```

```
307         */
308     }
309         if(close(sck) < 0)
310             printf("close error\n");
311         return 0;
312 }
313
314 /* upload file to ftp server */
315 int ftp_put(int sck,char *pUploadFileName_s)
316 {
317     //int c_sock;
318     int handle = open(pUploadFileName_s,O_RDWR);
319     int nread;
320     //error open
321     if(handle == -1)
322         return -1;
323     //ftp_type(c_sock,"I");
324
325     for(;;)
326     {
327         if((nread = read(handle,rbuf1,MAXBUF)) < 0)
328         {
329             printf("read error!");
330         }
331         else if(nread == 0)
332            break;
333         if(write(STDOUT_FILENO,rbuf1,nread) != nread)
334             printf("send error!");
335         if(write(sck,rbuf1,nread) != nread)
336             printf("send error!");
337     }
338     if(close(sck) < 0)
339          printf("close error\n");
340      return 0;
341 }
342
343
344 /* Read and write as command connection */
345 void cmd_tcp(int sockfd)
346 {
347     int maxfdp1,nread,nwrite,fd,replycode,tag=0,data_sock;
348     int port;
349     int newfilenameLen;
350     char *pathname;
351     fd_set rset;
352     FD_ZERO(&rset);
353     maxfdp1 = sockfd + 1;        /* check descriptors [0..sockfd] */
354
355     for(;;)
356     {
357         //1.将  标准输入加入 可读文件描述符集合
358          FD_SET(STDIN_FILENO,&rset);
359         //2.将  命令套接字    加入可读文件描述符集合
360          FD_SET(sockfd,&rset);
361
362         //3.监听读事件
363          if(select(maxfdp1,&rset,NULL,NULL,NULL)<0)
364          {
365              printf("select error\n");
366          }
367          //4.判断标准输入是否有读事件
368          if(FD_ISSET(STDIN_FILENO,&rset))
369           // && replycode  != PASSWORD
370          {
371              //5.清空读缓冲区 和 写缓冲区
372               bzero(wbuf,MAXBUF);          //zero
373               bzero(rbuf1,MAXBUF);
374
375
376               //set_disp_mode(STDIN_FILENO,1);
377
378               if((nread = read(STDIN_FILENO,rbuf1,MAXBUF)) <0)
379                    printf("read error from stdin\n");
380               nwrite = nread + 5;
381
382
383             /* send username */
```

```
384                 if(replycode == USERNAME)
385                 {
386                     sprintf(wbuf,"USER %s",rbuf1);
387 
388                    if(write(sockfd,wbuf,nwrite) != nwrite)
389                    {
390                        printf("write error\n");
391                    }
392                    //printf("%s\n",wbuf);
393                    //memset(rbuf1,0,sizeof(rbuf1));
394                    //memset(wbuf,0,sizeof(wbuf));
395                    //printf("1:%s\n",wbuf);
396                 }
397                /************************************************************
398                //4. code here: send password
399                ************************************************************/
400 
401                 if(replycode == PASSWORD)
402                 {
403                      //printf("%s\n",rbuf1);
404                      sprintf(wbuf,"PASS %s",rbuf1);
405                      if(write(sockfd,wbuf,nwrite) != nwrite)
406                         printf("write error\n");
407                      //bzero(rbuf,sizeof(rbuf));
408                      //printf("%s\n",wbuf);
409                      //printf("2:%s\n",wbuf);
410                 }
411 
412                  /* send command */
413                 if(replycode == 550 || replycode == LOGIN || replycode == CLOSEDATA || replycode ==
    PATHNAME || replycode == ACTIONOK)
414                 {
415                   /* pwd -  print working directory */
416                   if(strncmp(rbuf1,"pwd",3) == 0)
417                   {
418                        //printf("%s\n",rbuf1);
419                        sprintf(wbuf,"%s","PWD\n");
420                        write(sockfd,wbuf,4);
421                        continue;
422                    }
423                    else if(strncmp(rbuf1,"quit",4) == 0)
424                    {
425                        sprintf(wbuf,"%s","QUIT\n");
426                        write(sockfd,wbuf,5);
427                        //close(sockfd);
428                       if(close(sockfd) <0)
429                          printf("close error\n");
430                       printf("221 Goodbye.");
431                       break;
432                    }
433                     /************************************************************
434                     // 5. code here: cd - change working directory/
435                     ************************************************************/
436                    else if(strncmp(rbuf1,"cd",2) == 0)
437                    {
438                        //sprintf(wbuf,"%s","PASV\n");
439                        sscanf(rbuf1,"%s %s", tmp, dirname);
440                        //printf("%s\n", dirname);
441                        int dirnameLen= strlen(dirname);
442                        //if not, the final character will be the \000
443                        dirname[dirnameLen] = '\n';
444                        sprintf(wbuf,"CWD %s",dirname);
445                        write(sockfd,wbuf,nread+1);
446 
447                        //sprintf(wbuf1,"%s","CWD\n");
448 
449                        continue;
450                    }
451 
452                    // mkdir function
453                    else if(strncmp(rbuf1,"mkdir",5) == 0)
454                    {
455                        //sprintf(wbuf,"%s","PASV\n");
456                        sscanf(rbuf1,"%s %s", tmp, dirname);
457                        //printf("%s\n", dirname);
458                        int dirnameLen= strlen(dirname);
459                        //if not, the final character will be the \000
```

```
460                     dirname[dirnameLen] = '\n';
461                     sprintf(wbuf,"MKD %s",dirname);
462                     write(sockfd,wbuf,nread+1);
463                     //sprintf(wbuf1,"%s","CWD\n");
464 
465                     continue;
466                 }
467 
468                 else if(strncmp(rbuf1,"delete",6) == 0)
469                 {
470                     tag = 4; //删除文件标识符
471                     //printf("%s\n",rbuf1);
472                     sprintf(wbuf,"%s","PASV\n");
473                     //printf("%s\n",wbuf);
474                     write(sockfd,wbuf,5);
475                     //read
476                     //sprintf(wbuf1,"%s","LIST -al\n");
477                     nwrite = 0;
478                     //write(sockfd,wbuf1,nwrite);
479                     //ftp_list(sockfd);
480                     continue;
481                 }
482 
483 
484                 /* ls - list files and directories*/
485                 else if(strncmp(rbuf1,"ls",2) == 0)
486                 {
487                     tag = 2;            //显示文件 标识符
488                     //printf("%s\n",rbuf1);
489                     sprintf(wbuf,"%s","PASV\n");
490                     //printf("%s\n",wbuf);
491                     write(sockfd,wbuf,5);
492                     //read
493                     //sprintf(wbuf1,"%s","LIST -al\n");
494                     nwrite = 0;
495                     //write(sockfd,wbuf1,nwrite);
496                     //ftp_list(sockfd);
497                     continue;
498                 }
499                  /************************************************************
500                  // 7. code here: get - get file from ftp server
501                  ************************************************************/
502                 else if(strncmp(rbuf1,"get",3) == 0)
503                 {
504                     tag = 1;            //下载文件标识符
505 
506                     //被动传输模式
507                     sprintf(wbuf,"%s","PASV\n");
508                     //printf("%s\n",s(rbuf1));
509                     //char filename[100];
510                     s(rbuf1,filename);
511 
512                     //printf("%s\n",filename);
513                     write(sockfd,wbuf,5);
514                     continue;
515                 }
516                  /************************************************************
517                  // 8. code here: put -  put file upto ftp server
518                  ************************************************************/
519                 else if(strncmp(rbuf1,"put",3) == 0)
520                 {
521                     tag = 3;            //上传文件标识符
522                     sprintf(wbuf,"%s","PASV\n");
523 
524                    //把内容赋值给  读缓冲区
525                     st(rbuf1,filename);
526                     //printf("%s\n",filename);
527                     write(sockfd,wbuf,5);
528                     continue;
529                 }
530 
531                 //binary mode and ascii mode
532                else if(strncmp(rbuf1,"binary",6) == 0)
533                 {
534                     sprintf(wbuf,"TYPE %s","I\n");
535 
536                    //把内容赋值给  读缓冲区
```

```
537                     st(rbuf1,filename);
538                     printf("%s\n",filename);
539                     write(sockfd,wbuf,7);
540                     continue;
541                 }
542                 else if(strncmp(rbuf1,"ascii",5) == 0)
543                 {
544                     sprintf(wbuf,"TYPE %s","A\n");
545 
546                    //把内容赋值给  读缓冲区
547                     st(rbuf1,filename);
548                     printf("%s\n",filename);
549                     write(sockfd,wbuf,7);
550                     continue;
551                 }
552 
553                 else if(strncmp(rbuf1,"rename",6) == 0)
554                 {
555                     //sprintf(wbuf,"%s","PASV\n");
556                     sscanf(rbuf1,"%s %s %s", tmp, filename,newfilename);
557                     //printf("%s\n", dirname);
558                     int filenameLen= strlen(filename);
559                     newfilenameLen = strlen(newfilename);
560                     //if not, the final character will be the \000
561                     filename[filenameLen] = '\n';
562                     newfilename[newfilenameLen] = '\n';
563                     sprintf(wbuf,"RNFR %s",filename);
564                     write(sockfd,wbuf,filenameLen+6);
565 
566 
567 
568                     nwrite = 0;
569                     //sprintf(wbuf1,"%s","CWD\n");
570 
571                     continue;
572                 }
573                 else{
574                  write(sockfd,rbuf1,strlen(rbuf1));
575                  continue;
576                 }
577 
578 
579 
580 
581              }
582                    /*if(close(sockfd) <0)
583                       printf("close error\n");*/
584             }
585             if(FD_ISSET(sockfd,&rset))
586             {
587             //9.清空读缓冲区 和 写缓冲区
588                 bzero(rbuf,strlen(rbuf));
589             //10.读套接字中的内容
590 
591                 if((nread = recv(sockfd,rbuf,MAXBUF,0)) <0)
592                      printf("recv error\n");
593                 else if(nread == 0)
594                   break;
595 
596                 //printf("%s",rbuf);
597 
598                 if(strncmp(rbuf,"220",3) ==0 || strncmp(rbuf,"530",3)==0)
599                 {
600                    /*if(write(STDOUT_FILENO,rbuf,nread) != nread)
601                        printf("write error to stdout\n");*/
602 
603                     //链接字符串
604 
605 
606                     //printf("your name:");
607                     //getpasswd(rbuf, 20);
608                     strcat(rbuf,"your name:");
609 
610                     nread += 12;
611                     replycode = USERNAME;
612                 }
613                 if(strncmp(rbuf,"331",3) == 0)
```

```
614                 {
615                    /*if(write(STDOUT_FILENO,rbuf,nread) != nread)
616                        printf("write error to stdout\n")*/;
617
618                    strcat(rbuf,"your password:\n");
619                    nread += 16;
620                    /*if(write(STDOUT_FILENO,rbuf,nread) != nread)
621                        printf("write error to stdout\n");*/
622                    replycode = PASSWORD;
623                 }
624                 if(strncmp(rbuf,"230",3) == 0)
625                 {
626                    /*if(write(STDOUT_FILENO,rbuf,nread) != nread)
627                        printf("write error to stdout\n");*/
628                    replycode = LOGIN;
629                 }
630                 if(strncmp(rbuf,"257",3) == 0)
631                 {
632                    /*if(write(STDOUT_FILENO,rbuf,nread) != nread)
633                        printf("write error to stdout\n");*/
634                    replycode = PATHNAME;
635                 }
636                 if(strncmp(rbuf,"226",3) == 0)
637                 {
638                    /*if(write(STDOUT_FILENO,rbuf,nread) != nread)
639                        printf("write error to stdout\n");*/
640                    replycode = CLOSEDATA;
641                 }
642                 if(strncmp(rbuf,"250",3) == 0)
643                 {
644                    /*if(write(STDOUT_FILENO,rbuf,nread) != nread)
645                        printf("write error to stdout\n");*/
646                    replycode = ACTIONOK;
647                 }
648                 if(strncmp(rbuf,"550",3) == 0)
649                 {
650                    replycode = 550;
651                 }
652
653                 // Ready to rename
654                 if(strncmp(rbuf,"350",3) == 0)
655                 {
656                    bzero(wbuf,strlen(wbuf));
657                    sprintf(wbuf,"RNTO %s",newfilename);
658                    write(sockfd,wbuf,newfilenameLen+6);
659                    replycode = 350;
660                 }
661
662                 /*if(strncmp(rbuf,"150",3) == 0)
663                 {
664                    if(write(STDOUT_FILENO,rbuf,nread) != nread)
665                        printf("write error to stdout\n");
666                 }*/
667                 //fprintf(stderr,"%d\n",1);
668                 if(strncmp(rbuf,"227",3) == 0)
669                 {
670                    //printf("%d\n",1);
671                    /*if(write(STDOUT_FILENO,rbuf,nread) != nread)
672                       printf("write error to stdout\n");*/
673
674                    //获取服务器返回的 接收数据的端口，和地址
675                    int port1 = strtosrv(rbuf);
676                    printf("%d\n",port1);
677                    printf("%s\n",host);
678
679                    //创建新的传输数据的套接字?
680                    //1. 猜测 ====================应该是将 ssl 接口放在这里，用来传输数据
681                    data_sock = cliopen(host,port1);
682
683
684
685  //bzero(rbuf,sizeof(rbuf));
686                    //printf("%d\n",fd);
687                    //if(strncmp(rbuf1,"ls",2) == 0)
688                    if(tag == 2)
689                    {
690                       write(sockfd,"list\n",strlen("list\n"));
```

```
691                     ftp_list(data_sock);
692                     /*if(write(STDOUT_FILENO,rbuf,nread) != nread)
693                         printf("write error to stdout\n");*/
694
695                 }
696                 //else if(strncmp(rbuf1,"get",3) == 0)
697                 else if(tag == 1)
698                 {
699                     //sprintf(wbuf,"%s","RETR\n");
700                     //printf("%s\n",wbuf);
701                     //int str = strlen(filename);
702                     //printf("%d\n",str);
703                     sprintf(wbuf,"RETR %s\n",filename);
704
705                     //printf("%s\n",wbuf);
706                     //int p = 5 + str + 1;
707
708                     //命令套接字中写入  下载文件命令
709                     write(sockfd,wbuf,strlen(wbuf));
710
711                     //9.清空读缓冲区 和 写缓冲区
712                     bzero(rbuf,strlen(rbuf));
713                       //10.读套接字中的内容
714
715                     if((nread = recv(sockfd,rbuf,MAXBUF,0)) <0)
716                         printf("recv error\n");
717
718                     //printf("%s",rbuf);
719                     if(strncmp(rbuf,"550",3) == 0)
720                     {
721                       replycode = 550;
722                     }
723                     // 如果有这个文件，就下载
724                     else{
725                       ftp_get(data_sock,filename);
726                     }
727                     //printf("%d\n",write(sockfd,wbuf,strlen(wbuf)));
728                     //printf("%d\n",p);
729
730                     //下载文件
731
732                 }
733                 else if(tag == 3)
734                 {
735                     // 上传文件
736                     sprintf(wbuf,"STOR %s\n",filename);
737                     //printf("%s",wbuf);
738                     int handle = open(filename,O_RDWR);
739                     //printf("%s",wbuf);
740
741                     //Handle file not exist error
742                     if (handle != -1)
743                     {
744                       close(handle);
745                       write(sockfd,wbuf,strlen(wbuf));
746                       ftp_put(data_sock,filename);
747                     }
748                     else{
749                       bzero(wbuf,strlen(wbuf));
750                       printf("File not exist\n");
751                       printf("%s",wbuf);
752                     }
753                        //int c_sock;
754
755
756
757                 }
758                 else if (tag == 4){
759                    //sprintf(wbuf,"%s","PASV\n");
760                      sscanf(rbuf1,"%s %s", tmp, filename);
761                      //printf("%s\n", dirname);
762                      int filenameLen= strlen(filename);
763                      //if not, the final character will be the \000
764                      filename[filenameLen] = '\n';
765                      sprintf(wbuf,"DELE %s",filename);
766                      write(sockfd,wbuf,nread+1);
767                      //sprintf(wbuf1,"%s","CWD\n");
```

```
768
769                     continue;
770                 }
771                 nwrite = 0;
772             }
773             /*if(strncmp(rbuf,"150",3) == 0)
774             {
775                 if(write(STDOUT_FILENO,rbuf,nread) != nread)
776                     printf("write error to stdout\n");
777             }*/
778             //printf("%s\n",rbuf);
779             if(write(STDOUT_FILENO,rbuf,nread) != nread)
780                 printf("write error to stdout\n");
781
782
783              //如果该输密码了，让输入不可见
784              if (replycode == PASSWORD)
785              {
786                set_disp_mode(STDIN_FILENO,0);
787              }
788              else{
789                set_disp_mode(STDIN_FILENO,1);
790              }
791
792             /*else
793                 printf("%d\n",-1);*/
794         }
795     }
796 }
```