简易 ftp 服务器

为了方便在 UNIX 或 Linux 系统中传输文件，这个简易 ftp 服务器便产生了，您可以使用系统自带的 ftp 客户端程序进行登录。目前支持的命令有：

ascii、binary、bye、cd、delete、dir、get、mget、ls、mget、mkdir、mput、put、passive

pwd、quit、rmdir、size 等

为了从更直观的角度来了解这个服务器，先列出其主要的源文件：

```
root@localhost:~/code/unix_socket
File  Edit  View  Terminal  Tabs  Help
[root@localhost unix_socket]# ls
dxyh.h          dxyh_thread_lib.c  error.h  ftpd_main.c  record.h
dxyh_lib.c      edit1              ftpd.c   Makefile
dxyh_thread.h   error.c            ftpd.h   record.c
[root@localhost unix_socket]#
```

文件说明如下（查看源文件时设置 VI 中的 tabstop=4，shiftwidth=4 为最佳）：

dxyh.h —— 常用函数的包裹函数的声明头文件

dxyh_lib.c —— 常用包裹函数实现源文件

dxyh_thread.h —— 线程相关的包裹函数声明头文件

dxyh_thread_lib.c —— 线程相关的包裹函数实现源文件

record.h

record.c —— 日志系统

error.h

error.c —— 错误处理系统

ftpd.c —— ftp 服务器的主要实现源文件

ftpd_main.c —— 主函数文件

至于文件文件之间的相互联系，看下 Makefile 就一目了然了：

```
root@localhost:~/code/unix_socket
File  Edit  View  Terminal  Tabs  Help
SUBOPTS:=-Wall -c -o
OBJECTS:=dxyh_lib.o ftpd.o ftpd_main.o error.o record.o dxyh_thread_lib.o
SOURCES:=dxyh_lib.c ftpd.c ftpd_main.c error.c record.c dxyh_thread_lib.c
HEADERS:=dxyh.h dxyh_thread.h ftpd.h error.h record.h

ftpd_main: $(OBJECTS)
    $(CC) $(OPTIONS) $^ -o $@
ftpd_main.o: ftpd_main.c record.h ftpd.h dxyh_thread.h
    $(CC) ftpd_main.c $(SUBOPTS) $@
error.o: error.c error.h dxyh_thread.h
    $(CC) error.c $(SUBOPTS) $@
record.o: record.c record.h error.h dxyh.h dxyh_thread.h
    $(CC) record.c $(SUBOPTS) $@
dxyh_lib.o: dxyh_lib.c dxyh.h error.h
    $(CC) dxyh_lib.c $(SUBOPTS) $@
dxyh_thread_lib.o: dxyh_thread_lib.c dxyh_thread.h error.h dxyh_thread.h
    $(CC) dxyh_thread_lib.c $(SUBOPTS) $@
ftpd.o: ftpd.c error.h record.h ftpd.h dxyh.h
    $(CC) ftpd.c $(SUBOPTS) $@

.PHONY: clean
clean:
    rm -f *.o *.txt ftpd_main
"Makefile" 26L, 882C                                    26,2-5
```

下面根据程序的运行过程来进行说明：

运行环境为：Redhat 和 CentOS4.7，其中由 CentOS（以下简称【C】）运行这个服务器程序，而 Redhat（以下简称【R】）则运行系统自带的 ftp 客户端程序。

首先进行网址的配置，结果如下：

CentOS　10.6.173.225

Redhat　10.6.173.226

```
Applications  Actions
root@localhost:~
File  Edit  View  Terminal  Tabs  Help
[root@localhost ~]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:0C:29:06:24:3C
          inet addr:10.6.173.225  Bcast:10.255.255.255  Mask:255.0.0.0
          inet6 addr: fe80::20c:29ff:fe06:243c/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
```

```
Applications  Places  System
root@localhost:~
File  Edit  View  Terminal  Tabs  Help
[root@localhost ~]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:0C:29:15:0C:CE
          inet addr:10.6.173.226  Bcast:10.255.255.255  Mask:255.0.0.0
          inet6 addr: fe80::20c:29ff:fe15:cce/64 Scope:Link
```

为了便于测试，现在【C】中和【R】中分别建立两个文件夹/try_C 和/try_R。

我们的可执行文件名为 ftpd_main，它可支持长短选项，具体支持哪些选项看以下程序片段就可知道：

```
struct option longopts[] = {
    { "port", required_argument, NULL, 'p' },
    { "debug", no_argument, NULL, 'd' },
    { "record", optional_argument, NULL, 'r' },
    { "verbose", no_argument, &do_verbose, 1 },
    { "help", no_argument, &do_help, 1 },
    { 0, 0, 0, 0 }
};

do_verbose = OFF;
do_help = OFF;
err_flg = 0;
while ((c = getopt_long(argc, argv, ":hr::vp:dW;", longopts, NULL)) != -1) {
```

具体情况如下运行所示：

```
root@localhost:~/code/unix_socket
File  Edit  View  Terminal  Tabs  Help
[root@localhost unix_socket]# ftpd_main -h
You can use any ftp client software to login.
Cmds as follows are supported:
user ls cd pwd get mget delete put mput
size port passive ascii binary dir mkdir
......
Well, you can add more cmds!
[root@localhost unix_socket]# ftpd_main --verbose
A simple ftp server.
It can support some of the mostly used cmds.
Author: jiangweifeng
Mail: jwf0713@gmail.com
Ftpd version: 1.1
[root@localhost unix_socket]# ftpd_main -u
ftpd_main: u: unknow option
usage: ftpd_main [-p <port#>] [-r["filenam"]][-v] [-d] [-h]
[root@localhost unix_socket]#
```

上面运行的-h/--help、-v/--verbose 选项分别表示显示帮助和版本信息，若选项错误（如最下面所所示）则会显示用法并退出。其他选项为：

-p 可以指定监听端口，默认是 9877（跟系统中的 ftp 服务器端口 21 相区别）

-r 可以指定日志文件名，默认为当前目录下的 logfile.txt

-d 是否打开调试模式，若不打开则在服务器端不会出现任何信息，除非当系统出现致命错误，日志系统中 EMERG 级别的信息才会显示出来。

下面就运行这个服务器：

```
root@localhost:~/code/unix_socket
File  Edit  View  Terminal  Tabs  Help
[root@localhost unix_socket]# ftpd_main -d -r
(ftpd.c:271:19618) Prepare to log in "logfile.txt" ok.
(ftpd.c:274:19618) serv port 9877
(ftpd.c:332:19618) create serv-listenfd ok 0.0.0.0:9877
(ftpd.c:346:19618) server is ready for a new  connection ...
```

说明：现在服务器处于监听状态，等待连接。

然后就在【R】中用系统自带的 ftp 客户端登录，用户名密码分别为：

dengxiayehu　　// 灯下野狐，呵呵

及

123456

具体登录情况如下所示：

```
root@localhost:/try_R
File  Edit  View  Terminal  Tabs  Help
[root@localhost try_R]# ftp 10.6.173.225 9877
Connected to 10.6.173.225.
220 Ftpd1.0.1 ready for new user.
530 Please login with USER and PASS.
530 Please login with USER and PASS.
KERBEROS_V4 rejected as an authentication type
Name (10.6.173.225:root): dengxiayehu
331 Please sepcify the password.
Password:
230 Login successful.
Remote system type is Linux.
ftp>
```

发现用规定的用户名及密码确实可以登录，再来看看服务器端，它会输出一大堆的调试信息，部分如下：

```
root@localhost:~/code/unix_socket
File  Edit  View  Terminal  Tabs  Help
(ftpd.c:274:19440) serv port 9877
(ftpd.c:332:19440) create serv-listenfd ok 0.0.0.0:9877
(ftpd.c:346:19440) server is ready for a new  connection ...
(ftpd.c:359:19440) accept a connection from 10.6.173.226:2369
(ftpd.c:512:19441) send resp: 220 Ftpd1.0.1 ready for new user.
(ftpd.c:346:19440) server is ready for a new  connection ...
(ftpd.c:575:19441) Got cmd = AUTH GSSAPI
(ftpd.c:537:19441) received a valid cmd AUTH GSSAPI
(ftpd.c:512:19441) send resp: 530 Please login with USER and PASS.
(ftpd.c:575:19441) Got cmd = AUTH KERBEROS_V4
(ftpd.c:537:19441) received a valid cmd AUTH KERBEROS_V4
(ftpd.c:512:19441) send resp: 530 Please login with USER and PASS.
(ftpd.c:575:19441) Got cmd = USER dengxiayehu
(ftpd.c:537:19441) received a valid cmd USER dengxiayehu
(ftpd.c:606:19441) certain user(dengxiayehu) is found
(ftpd.c:512:19441) send resp: 331 Please sepcify the password.
(ftpd.c:575:19441) Got cmd = PASS 123456
(ftpd.c:537:19441) received a valid cmd PASS 123456
(ftpd.c:632:19441) password for dengxiayehu ok
(ftpd.c:512:19441) send resp: 230 Login successful.
(ftpd.c:575:19441) Got cmd = SYST
(ftpd.c:537:19441) received a valid cmd SYST
(ftpd.c:512:19441) send resp: 215 Linux Type
```

至此，ftp 客户端就可以进行一系列命令操作了，所支持的命令见如下程序片段：

```
root@localhost:~/c
File  Edit  View  Terminal  Tabs  Help
root@localhost:~/code/unix_socket
const struct ftpd_cmd_st ftpd_cmds[] = {
    { "AUTH", ftpd_do_auth },
    { "USER", ftpd_do_user },
    { "PASS", ftpd_do_pass },
    { "PWD",  ftpd_do_pwd  },
    { "XPWD", ftpd_do_pwd  },
    { "CWD",  ftpd_do_cwd  },
    { "LIST", ftpd_do_list },
    { "MKD",  ftpd_do_mkd  },
    { "XMKD", ftpd_do_mkd  },
    { "SYST", ftpd_do_syst },
    { "SIZE", ftpd_do_size },
    { "DELE", ftpd_do_dele },
    { "RMD",  ftpd_do_rmd  },
    { "TYPE", ftpd_do_type },
    { "RETR", ftpd_do_retr },
    { "STOR", ftpd_do_stor },
    { "NLST", ftpd_do_nlst },
    { "PASV", ftpd_do_pasv },
    { "PORT", ftpd_do_port },
    { "QUIT", ftpd_do_quit },
    { NULL, NULL },
};
```

由于命令较多，不可能个个都举例说明，以下只是部分例子：

```
root@localhost:/try_R
File  Edit  View  Terminal  Tabs  Help
ftp> ls
227 Entering Passive Mode (10,6,173,225,128,191).
150 Here comes the directory listing.
drwxr-xr-x 1 root root    12288 Oct 09 10:30 sbin
drwxr-xr-x 1 root root        0 Oct 09 17:49 sys
drwxr-xr-x 1 root root     4096 Jul 26 07:21 misc
drwxr-xr-x 1 root root     6660 Oct 09 09:58 dev
drwxr-xr-x 1 root root     4096 Oct 09 16:48 try_C
drwxr-xr-x 1 root root     4096 Oct 09 09:02 var
drwxr-xr-x 1 root root     4096 Feb 22 08:49 srv
drwxr-xr-x 1 root root     4096 Feb 22 08:49 opt
drwxr-xr-x 1 root root     4096 Oct 09 09:06 usr
drwxr-xr-x 1 root root     4096 Feb 22 08:49 initrd
drwxr-xr-x 1 root root     4096 Oct 09 09:59 media
drwxr-xr-x 1 root root     4096 Feb 22 08:49 home
drwxr-x--- 1 root root     4096 Oct 09 16:44 root
drwx------ 1 root root    16384 Oct 09 16:49 lost+found
dr-xr-xr-x 1 root root        0 Oct 09 17:49 proc
drwxr-xr-x 1 root root        0 Oct 09 17:49 selinux
drwxrwxrwx 1 root root     4096 Oct 09 16:05 tmp
drwxr-xr-x 1 root root     4096 Oct 09 10:20 lib
drwxr-xr-x 1 root root     4096 Oct 09 08:22 boot
drwxr-xr-x 1 root root     4096 Oct 09 09:26 tftpboot
drwxr-xr-x 1 root root     4096 Oct 09 10:24 bin
drwxr-xr-x 1 root root     4096 Feb 22 08:49 mnt
drwxr-xr-x 1 root root    12288 Oct 09 16:25 etc
drwxr-xr-x 1 root root     4096 Oct 09 09:44 windows_linux
226 Directory send OK.
ftp>
```

说明：当客户端连上之后，默认的当前目录是根目录，这里从【R】中也可以看到为了测试我们在【C】中建立的测试文件夹/try_C。这个显示模式和 shell 的 ls –l 显示模式类似(实际上我的初衷就是尽可能和其一致)，具体实现也就是打开一个文件夹，然后读取各个文件的属性，然后就是整理输出的事情。

下面就进入到【C】中的/try_C 目录：

```
ftp> cd /try_C
250 Directory successfully changed.
ftp> pwd
257 /try_C
ftp> ls
227 Entering Passive Mode (10,6,173,225,128,192).
150 Here comes the directory listing.
226 Directory send OK.
ftp>
```

看到进入该目录成功，并显示当前的目录确实是/try_C，此时用 ls 命令查看到该目录下没有任何文件。ftp 主要也就是用来传输文件嘛，故下面就测试它，可以支持 BIN 模式和 ASCII 模式，其实也就是对\n 字符的处理不同。若采用 ASCII 模式来传输非文本文件时会发现文件莫名变大了（\n 被解释为\r\n），当然那个文件也就毁了，慎重！

下面我们现在【C】中建立一个名为 hello.txt 的文件，内容为如下所示：

```
root@localhost:/try_C
File  Edit  View  Terminal  Tabs  Help
root@localhost:~/code/unix_socket   root@localhost:/try_C
[root@localhost try_C]# cat hello.txt
Hello client, this is ftpd 1.1.
Welcome to download or up-put, hope you enjoy it.

some meaningless words are as follows...

asdf ljl;jk   asdljf  asdf

Well, that's all.
[root@localhost try_C]#
```

下面转到【R】中来下载此文件，并比较文件内容是否和上面一致：

```
root@localhost:/try_
File  Edit  View  Terminal  Tabs  Help
ftp> ls
227 Entering Passive Mode (10,6,173,225,128,195).
150 Here comes the directory listing.
-rw-r--r-- 1 root root      171 Oct 09 16:47 hello.txt
226 Directory send OK.
ftp> ascii
200 Switching to ASCII mode.
ftp> get
(remote-file) hello.txt
(local-file) hello.txt
local: hello.txt remote: hello.txt
227 Entering Passive Mode (10,6,173,225,128,196).
150 File status OK, about to transfer.
226 Require file transferd OK.
179 bytes received in 0.00072 seconds (2.4e+02 Kbytes/s)
ftp> !cat hello.txt
Hello client, this is ftpd 1.1.
Welcome to download or up-put, hope you enjoy it.

some meaningless words are as follows...

asdf ljl;jk   asdljf  asdf

Well, that's all.
ftp>
```

发现内容完全一致，表明传输正确，现在可以看看服务器的调试信息（片段）：

```
(ftpd.c:575:19441) Got cmd = LIST
(ftpd.c:537:19441) received a valid cmd LIST
(ftpd.c:512:19441) send resp: 150 Here comes the directory listing.
(ftpd.c:512:19441) send resp: 226 Directory send OK.
(ftpd.c:575:19441) Got cmd = TYPE A
(ftpd.c:537:19441) received a valid cmd TYPE A
(ftpd.c:512:19441) send resp: 200 Switching to ASCII mode.
(ftpd.c:575:19441) Got cmd = PASV
(ftpd.c:537:19441) received a valid cmd PASV
(ftpd.c:1000:19441) local bind: 10.6.173.225: 32964
(ftpd.c:512:19441) send resp: 227 Entering Passive Mode (10,6,173,225,128,196).
(ftpd.c:575:19441) Got cmd = RETR hello.txt
(ftpd.c:537:19441) received a valid cmd RETR hello.txt
(ftpd.c:512:19441) send resp: 150 File status OK, about to transfer.
#
Bytes transferred: 179
179 bytes send in 0.00688 secs(25 Kbytes/s)
(ftpd.c:1322:19441) RETR "hello.txt" successfully
(ftpd.c:512:19441) send resp: 226 Require file transferd OK.
```

说明：服务器端可以显示传输的字节大小及速率，并报告传输结果。

为了更能说明问题，下面就来用 BIN 模式上传一个大点的文件，这里为 arm-elf.tar.gz，大小为 30 几兆。

在【R】中首先要先使用 binary 命令将模式切换为 binary 模式，否则就会出现上面所述的问题，压缩包文件大小会变大，当然文件也就毁了。上传的命令是 put，当然这个程序也是支持多文件上传的，命令为 mput，传输时在服务器端会每接收 1K 个字节显示一个’#’来表示进度。

下面就将上传的具体过程贴出来：

（先是【R】中的客户端部分）

```
root@localhost:/try_R
File  Edit  View  Terminal  Tabs  Help
ftp> bin
200 Switching to Binary mode.
ftp> put
(local-file) arm-elf.tar.gz
(remote-file) arm-elf.tar.gz
local: arm-elf.tar.gz remote: arm-elf.tar.gz
227 Entering Passive Mode (10,6,173,225,128,197).
150 Ready to receive file.
226 File received OK.
36813092 bytes sent in 58 seconds (6.2e+02 Kbytes/s)
ftp> ls
227 Entering Passive Mode (10,6,173,225,128,198).
150 Here comes the directory listing.
-rw-r--r-- 1 root root      171 Oct 09 16:47 hello.txt
-rw-r----- 1 root root 36813092 Oct 09 17:35 arm-elf.tar.gz
226 Directory send OK.
ftp> !ls -l
total 36004
-rwxr--r-- 1 root root 36813092 Jan 31  2010 arm-elf.tar.gz
-rw-r--r-- 1 root root      171 Sep 23 15:41 hello.txt
ftp>
```

说明：现在本地看到文件大小为 36813092，可以使用 size 命令来查看上传到服务器的那个文件的大小，看是否一致：

```
ftp> size
(filename) arm-elf.tar.gz
213 Size of "arm-elf.tar.gz" is 36813092.
ftp>
```

为了更能说明问题，下面就转到【C】中对那个压缩包进行解压，看有没有问题，具体如下：

```
root@localhost:/try_C
File  Edit  View  Terminal  Tabs  Help
root@localhost:~/code/unix_socket   root@localhost:/try_C
[root@localhost unix_socket]# cd /try_C
[root@localhost try_C]# ls
arm-elf.tar.gz  hello.txt
[root@localhost try_C]# tar -zxvf arm-elf.tar.gz
./usr/local/arm-elf/
./usr/local/arm-elf/bin/
./usr/local/arm-elf/bin/nm
./usr/local/arm-elf/bin/strip
./usr/local/arm-elf/bin/ar
./usr/local/arm-elf/bin/ranlib
./usr/local/arm-elf/bin/as
./usr/local/arm-elf/bin/ld
./usr/local/arm-elf/bin/elf2flt
./usr/local/arm-elf/bin/gcc
./usr/local/arm-elf/bin/ld.real
./usr/local/arm-elf/lib/
./usr/local/arm-elf/lib/ldscripts/
./usr/local/arm-elf/lib/ldscripts/armelf.x
./usr/local/arm-elf/lib/ldscripts/armelf.xbn
./usr/local/arm-elf/lib/ldscripts/armelf.xn
./usr/local/arm-elf/lib/ldscripts/armelf.xr
./usr/local/arm-elf/lib/ldscripts/armelf.xs
./usr/local/arm-elf/lib/ldscripts/armelf.xu
./usr/local/arm-elf/lib/elf2flt.ld
./usr/local/arm-elf/lib/libg.a
./usr/local/arm-elf/lib/mbig-endian/
```

说明：看到确实可以正确解压，就表明压缩包在传输过程中没有问题。

下面来看看服务器端，它会输出什么：

```
root@localhost:~/code/unix_socket
File  Edit  View  Terminal  Tabs  Help
root@localhost:~/code/unix_socket    root@localhost:/try_C
#######################################################################
#######################################################################
#######################################################################
#######################################################################
#######################################################################
#######################################################################
#######################################################################
#######################################################################
#######################################################################
#######################################################################
#######################################################
Bytes transferred: 36813092
36813092 bytes received in 18.1 secs(2e+03 Kbytes/s)
(ftpd.c:1470:19441) STOR (arm-elf.tar.gz) successfully
(ftpd.c:512:19441) send resp: 226 File received OK.
(ftpd.c:575:19441) Got cmd = TYPE A
(ftpd.c:537:19441) received a valid cmd TYPE A
(ftpd.c:512:19441) send resp: 200 Switching to ASCII mode.
(ftpd.c:575:19441) Got cmd = PASV
(ftpd.c:537:19441) received a valid cmd PASV
(ftpd.c:1000:19441) local bind: 10.6.173.225: 32966
(ftpd.c:512:19441) send resp: 227 Entering Passive Mode (10,6,173,225,128,198).
(ftpd.c:575:19441) Got cmd = LIST
(ftpd.c:537:19441) received a valid cmd LIST
(ftpd.c:512:19441) send resp: 150 Here comes the directory listing.
(ftpd.c:512:19441) send resp: 226 Directory send OK.
```

至此，一些主要的命令已介绍得差不多了，若需要关闭服务器，只需按下 CTRL+C 即可，我捕捉了这个信号，它会进行一系列处理之后安全退出程序。

```
(ftpd.c:575:19441) Got cmd = QUIT
(ftpd.c:537:19441) received a valid cmd QUIT
(ftpd.c:512:19441) send resp: 221 Goodbye.
(ftpd.c:582:19441) client exits normally
(ftpd.c:392:19440) child 19441 terminated normally
(ftpd.c:430:19440) user time = 0.094985, sys time = 2.97955
(ftpd.c:439:19440) ftpd interrupted by signal SIGINT!
(ftpd.c:1495:19440) Server is shutdown!
```

顺带看一下日志文件中的内容，具体如下：

```
root@localhost:~/code/unix_socket
File  Edit  View  Terminal  Tabs  Help
Sat Oct  9 17:55:58 [info](PID:19721) Prepare to log in "logfile.txt" ok.
Sat Oct  9 17:55:58 [info](PID:19721) create serv-listenfd ok 0.0.0.0:9877
Sat Oct  9 18:02:07 [info](PID:19722) local bind: 10.6.173.225: 32967
Sat Oct  9 18:02:54 [info](PID:19722) local bind: 10.6.173.225: 32968
Sat Oct  9 18:03:20 [info](PID:19722) local bind: 10.6.173.225: 32969
Sat Oct  9 18:06:02 [info](PID:19722) local bind: 10.6.173.225: 32970
Sat Oct  9 18:06:12 [info](PID:19722) local bind: 10.6.173.225: 32971
Sat Oct  9 18:07:11 [info](PID:19722) local bind: 10.6.173.225: 32972
Sat Oct  9 18:12:17 [info](PID:19722) local bind: 10.6.173.225: 32973
Sat Oct  9 18:12:26 [info](PID:19722) local bind: 10.6.173.225: 32974
Sat Oct  9 18:19:28 [info](PID:19722) client exits normally
Sat Oct  9 18:19:28 [info](PID:19721) user time = 0.109982, sys time = 5.99709
Sat Oct  9 18:19:37 [error](PID:19721) ftpd interrupted by signal SIGINT!
Sat Oct  9 18:19:37 [info](PID:19721) Server is shutdown!
~
~
```

日志内容为可裁剪型，具体级别有如下几种：debug、info、warn、error、emerg，其中 emerg 级别的信息

会自动在屏幕上显示出来，表明服务器出现了较为致命的错误。

对于 ftp 而言，采用传统的服务器为每个客户分配一个处理进程就足够了，但对于一个处理繁忙的服务器而言，采用这种服务器模型就欠妥，一般是采用分配线程处理的方式，这里又要涉及到是主线程 accept 还是每个子线程都 accept（要对各 个子线程的 accept 上锁）、主线程采用普通阻塞型 I/O 还是非阻塞型 I/O（很复杂，“性价比”不高）还是 select（可进行轮询）等等，这些都要根据实际情况来选择，但不管怎么样，由于现在很多电脑都是多核处理器，似乎采用线程是一个不错的选择。

这个程序采用分配进程的方式，同时主进程记录下所分配的各个子进程，在主进程接受到 SIGINT 信号时，依次先对各个子进程发送 SIGQUIT 消息，然后再退出。

限于篇幅，这里不准备进入到代码的分析之中，但有一点值得说明的是：对于传输部分的处理，若按如下方式，那么就会奇慢无比：

```
//////////////////////////////////////
// so slow the below is
//////////////////////////////////////
// while ((c = fgetc(fp)) != EOF) {
//      if ('\n' == c) {
//              c = '\r';
//              Writen(connfd, &c, 1);
//              c = '\n';
//      }
//      Writen(connfd, &c, 1);
// }
//////////////////////////////////////
```

改进后的代码如下所示，比上面的要快得多得多（输出‘#’等功能是附带的）：

```
char    tmpbuff[BUFSIZ];
register int i, k;
volatile int sz;
FILE    *fp;

while ((sz = fread(tmpbuff, 1, sizeof(tmpbuff)/2, fp)) > 0) {
    for (i = k = 0; i < sz; ++i) {
        if ('\n' == tmpbuff[i]) {
            if (ftpd_hash_print) {
                while (bytes >= hashbytes) {
                    putchar('#');
                    hashbytes += HASHBYTES;
                }
                fflush(stdout);
            }
            if (ftpd_tick_print && (bytes >= hashbytes)) {
                printf("\rBytes transferred: %ld", bytes);
                fflush(stdout);
                while (bytes >= hashbytes)
                    hashbytes += TICKBYTES;
            }
            bytes++;
            buff[k++] = '\r';
        }
        buff[k++] = tmpbuff[i];
        bytes++;
    }
    Writen(connfd, buff, k);
}
```

说明：策略就是先在本地将传输格式处理好之后一次性发送，而不是一个一个字节发送。当然还有其他的很多地方没有说明，各方考虑，暂且到此。

希望您在看后能够将之完善，使其更有用……

灯 下 野 狐

2 0 1 0 年 1 月

mail：dengxiayehu@yeah.net