FTP SERVER & CLIENT

李胜涛 2017013618 **软件**72

实验说明：

实验环境：

```
系统：linux（manjaro发行版）
内核：linux 4.19.79
gcc版本：9.2.0
python版本（指客户端）：3.7.4
```

运行方法：

server：

```
[sudo] ./server [OPTION...]
-port, --port=port set the server port
-root, --root=root set the root directory
```

client:

```
python client.py
```

备注：

客户端与服务端只支持anonymous**登录。**

代码说明：

server

1. **实现命令：**

```
command_list[] = {
    {"ABOR", 4,handle_abor}, //断开当前所有数据传输
    {"APPE", 4, handle_appe},//断点续传（上传）
    {"CWD",  3, handle_cwd}, //进入文件夹
    {"DELE", 4, handle_dele},//删除文件
    {"LIST", 4, handle_list},//获取目录信息
    {"MKD",  3, handle_mkd}, //创建文件夹
    {"PASS", 4, handle_pass},//密码
```

```
    {"PASV", 4, handle_pasv},//使用pasv模式传输数据
    {"PORT", 4, handle_port},//使用port模式传输数据
    {"PWD",  3, handle_pwd}, //获取当前文件夹（以初始文件夹为根目录）
    {"QUIT", 4, handle_quit},//退出
    {"RETR", 4, handle_retr},//下载文件
    {"RMD",  3, handle_rmd}, //删除文件夹（空文件夹）
    {"RNFR,",4, handle_rnfr},//重命名（要重命名的文件夹）
    {"RNTO,",4, handle_rnto},//重命名（新名称）
    {"SIZE", 4, handle_size},//获取大小
    {"STOR", 4, handle_stor},//上传文件
    {"SYST", 4, handle_syst},//获取系统信息
    {"TYPE", 4, handle_type},//设定传输方式
    {"USER", 4, handle_user},//设置用户
    {"REST", 4, handle_rest} //断点续传（下载）
};
```

2. **架构说明：**

   server**采用多路复用**IO**的**epoll**完成，通过对大量的**file descriptor**统一管理实现多客户端的事件处理而不堵塞。** epoll**区别于**select**和**poll**，通过**

   ```
   int epoll_create(int size);
   int epoll_ctl(int epfd, int op, int fd, struct epoll_event *event);
   int epoll_wait(int epfd, struct epoll_event *events,int maxevents, int
   timeout);
   ```

   三个系统调用实现对大量file descriptor**的飞速查询和处理。**epoll**没有**select**对**fd**轮询以及描述符个数限制等缺点，同时和内核共享内存也节省了内存拷贝的时间。使得**eopll**对于大量**fd**的维护非常高效。**

   client**采用**pyqt4**完成，设计思路为使得用户像使用文件夹一般浏览文件。客户端下载采用了多线程的做法，将所有下载任务放到另一个进程中使其并不影响**UI**的更新。**

3.