计算机网络第一次实验报告

软 22 邱泓钧 2011013259

目录

编译环境

操作系统	Ubuntu 14.04.1
实现语言	C 语言
执行方式	通过终端进入对应文件 make 编译并执行

UDP 实现

修改了 c 的源代码，为了方便 make 将客户端和服务器端源文件放在两个不同文件夹下对应不同的 make 文件。开启服务器再开启客户端后，客户端会向服务器端发送 50 个具有编号的消息，服务器接收到后则按收到顺序给消息附上第二组编号，打印并发回客户端。如此便能在客户端与服务器上，看到消息发送与返回的编号一致。

FTP 实现

简要介绍

FTP 文件传输协议是基于 TCP 上的应用层协议，他通过两条 TCP 套接字实现文件传输，第一个是“控制连接”，第二个是“数据连接”。“控制连接”在通信过程中是保证连接的，而“数据连接”只有在文件传输过程中才会打开，文件传输完毕则会关闭。

在通信过程中，服务器会先监听客户端，一旦与客户端建立连接，便开始会话，且一个服务器端可以接受多个客户端访问，且多个客户端之间会话互不影响。客户端需要连接时，需要在 21 端口初始化对应服务器的 TCP“控制连接”，并发送用户身份密码至服务器端，通过认证后可以实现其他会话。

默认端口号 21 作为 FTP 的 TCP 控制连接端口号。

实现情况

通过 filezilla 客户端软件可以成功访问个人服务器端并实现主要操作。

通过个人客户端可以成功访问科协的 FTP（168.111.80.5）并实现主要操作。

个人客户端可以成功访问个人服务器端并实现主要操作。

实现命令

以下命令客户端都可以发送，且服务器端都可以接收。

客户端可以发送 OPEN, USER, PASS, RETR, STOR, QUIT, SYST, TYPE, PORT, PASV, CWD, CDUP, DELE, LIST, MKD, PWD, RMD, RNFR, RNTO 等命令。为了避免输入切换麻烦，该命令对于大小写不敏感，即输入 open, user, pass, ……, rnto 也可以。此外，输入错误的命令除了提示错误外，还会自动列出可以使用的命令及其所需参数。

基础命令		
命令	**参数**	**简介**
USER	用户名	只有 anonymous 符合
PASS	密码	输入邮箱作为密码
RETR	文件名	从服务器下载文件
STOR	文件名	上次文件到服务器
QUIT	无	退出
SYST	无	获取操作系统类型
TYPE	传输类型	只有 I 符合
PORT	IP 与端口	主动模式
PASV	无	被动模式

扩展命令		
命令	**参数**	**简介**
CWD	目录	指定工作目录
CDUP	无	指定父目录为工作目录
DELE	文件名	删除指定文件
LIST	文件/目录	查看指定文件或目录文件信息
MKD	目录	创建目录
PWD	无	查看工作目录
RMD	目录	删除目录
RNFR	文件/目录	需要重命名的文件或目录

RNTO	文件/目录	设置新的文件/目录名

其他命令		
命令	**参数**	**简介**
OPEN	IP，端口号	客户端最初建立连接

服务器端

服务器监听客户端的连接，一旦有连接连接上服务器，便开始与客户端通信。接收到用户发送命令，判断是否能够执行，并对应执行不同的操作。

多线程部分采用 fork()实现多进程，保证服务器端可以与多个客户端同时会话。运行 fork()后，系统优先分配给新进程资源，将原进程的所以数据都复制到新进程，即克隆了一个新的代码。创建新进程成功后，新的子进程 fork 返回值为 0，原先的父进程 fork 返回新建子进程的进程 ID。如此服务器便能同时接受多个客户端发送信息且不发生堵塞。设定最大连接数 CLIENTNUM，通过 listen 监听，客户端数量超出则拒绝连接。

客户端

开启一个 FTP 会话时，用户需要执行 OPEN 命令，客户端先在端口 21 初始化一个控制连接，并发送用户身份密码命令到服务器端，服务器通过验证后，通过控制链接客户端可以实现以上的主要命令。

当用户输入对应命令时，会将其解析成 cmd 和 arg 部分，前者为命令后者为对应参数，通过 sendMessage 将其发送给服务器端，并通过 recvMessage 获取服务器端返回的信息打印出来。

心得感想

UDP 部分较为简单，却很好地帮助 FTP 的上手。

FTP 有很多细节需要注意，需要耐心细心去一步一步完成。实现 STOR、RETR、PORT 和 PASV 等花费了较多的精力。服务器端与客户端之间通信时 read 和 write 部分，客户端发送一条消息需要伴随读取服务器返回的信息。

通过这次实验，对老师课上所讲的知识点有了进一步深入的理解，也提高了本人对于 socket 编程的能力。

此外，在 linux 下使用 c 语言调试遇到了很大的困难，几乎都是靠 printf 实现。此外难忘的是犯了一个很简单却很坑的错误，printf 输出的字符没有\n，导致有些字符存在缓冲区没有输出，对于 debug 造成了严重的误判。