

学士学位毕业设计

**基于私有协议的文件共享**

**平台设计与实现**

**姓 名** 柴奥

**学 号** 20183090219

**院 系** 信息工程学院

**专 业** 计算机科学与技术

**班 级** 计182

**指导教师**  刘雅军

**辅导教师** 刘雅军

2022 年 6 月 1 日

毕业设计原创性声明

本人所提交的毕业设计  **基于私有协议的文件共享平台设计与实现**  ，是在导师的指导下，独立进行研究工作所取得的原创性成果。除文中已经注明引用的内容外，本毕业设计不包含任何其他个人或集体已经发表或撰写过的研究成果。对本文的研究做出重要贡献的个人和集体，均已在文中标明。

本声明的法律后果由本人承担。

毕业设计作者（签名）： 指导教师确认（签名）：

年 月 日 年 月 日

毕业设计版权使用授权书

本毕业设计作者完全了解河北建筑工程学院有权保留并向国家有关部门或机构送交毕业设计的复印件和磁盘，允许毕业设计被查阅和借阅。本人授权河北建筑工程学院可以将毕业设计的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或其它复制手段保存、汇编毕业设计。

保密的毕业设计在\_\_\_\_\_\_\_年解密后适用本授权书。

毕业设计作者（签名）： 指导教师（签名）：

年 月 日 年 月 日

摘 要

当今社会，信息的传递与交换，越来越多地依靠因特网，现在已经有不少的互联网站都创建了资料库供广大用户访问，使用者只需通过文件传输系统，便可随时随地获取资料。至今，文件传输系统的高安全性、高效率、远距离访问等特性仍不能被取代。基于此本设计所实现的文件共享平台，可以把用户上传的数据存储在服务端磁盘里，让具有上传下载等权限的用户通过客户端进行数据的上传下载等基本操作。

通过对文件共享平台的需求分析设计出了基于私有应用层传输协议的文件共享平台的各功能模块，根据系统的功能模块设计并确定了文件共享平台数据库的结构。系统采用C/S架构，利用socket编程实现服务器和客户端的通信，并设置了应用层私有传输协议(包括文件名、用户名、文件的MD5值、文件大小、文件内容)，保证数据能够正常传输，并记录服务器操作日志，使用线程池和epoll多路复用响应多客户端的请求，利用文件MD5码值保证传输的差错控制，使用虚拟文件系统和mysql数据库来设计服务器的文件系统，节省数据库存储空间，不产生文件冗余。平台提供用户注册、登陆、查看、上传、下载、断点续传、秒传、创建子目录以及创建文件等功能。本文介绍并阐述了文件共享平台中的主要数据库表结构和系统整体流程图，其中分别包括服务端与客户端主线程与子线程的流程图。本论文在介绍系统整体流程图的基础之上再介绍系统相关功能模块的设计，通过vscode、navicat、vim等软件开发工具，完成了该文件共享平台的主要功能。

关键词：文件共享平台；断点续传；秒传

ABSTRACT

In today's society, the transmission and exchange of information, more and more rely on the Internet, and now there are many Internet stations have created a database for the majority of users to visit, users only through the file transmission system, they can obtain information anytime and anywhere. Up to now, the file transfer system of high security, high efficiency, remote access and other characteristics can not be replaced. Based on this design, the file sharing platform can store the data uploaded by users in the server disk, allowing users with upload and download permissions to upload and download data through the client and other basic operations.

Based on the analysis of the requirements of the file sharing platform, the functional modules of the file sharing platform based on the private application layer transfer protocol are designed and the database structure of the file sharing platform is determined according to the functional modules of the system. The system uses C/S architecture, uses socket programming to realize the communication between the server and the client, and sets the application layer private transmission protocol (including file name, user name, MD5 value of file, file size, file content) to ensure the normal transmission of data, and record the server operation log. Thread pool and epoll multiplexing are used to respond to requests from multiple clients, MD5 code values of files are used to ensure error control of transmission, and virtual file system and mysql database are used to design the file system of the server to save database storage space and avoid file redundancy. The platform provides user registration, login, view, upload, download, resumable, second transfer, create subdirectory and create files and other functions. This paper introduces and expounds the main database table structure and the overall flow chart of the system in the file sharing platform, including the flow chart of the main thread and sub-thread of the server and client respectively. This paper introduces the design of related functional modules of the system on the basis of the overall flow chart of the system. Through vscode, Navicat, ViM and other software development tools, the main functions of the file sharing platform are completed.

**Key words:** file sharing platform; Breakpoint continuation; A pass

目 录

[第1章 绪 论 1](#_Toc104725595)

[1.1 研究背景 1](#_Toc104725596)

[1.2 研究现状 1](#_Toc104725597)

[1.3 研究目的与意义 1](#_Toc104725598)

[1.4 设计目标 2](#_Toc104725599)

[1.5 论文章节安排 2](#_Toc104725600)

[第2章 相关技术简介 3](#_Toc104725601)

[2.1 Linux简介 3](#_Toc104725602)

[2.2 socket简介 3](#_Toc104725603)

[2.2.1 socket的创建 3](#_Toc104725604)

[2.2.2 socket常用函数 4](#_Toc104725605)

[2.2.3 socket工作过程 4](#_Toc104725606)

[2.3 线程池技术 5](#_Toc104725607)

[2.4 私有应用层传输协议 6](#_Toc104725608)

[2.5 本章小结 7](#_Toc104725609)

[第3章 系统需求与可行性分析 8](#_Toc104725610)

[3.1 需求分析 8](#_Toc104725611)

[3.1.1 用户需求分析 8](#_Toc104725612)

[3.1.2 服务端功能需求分析 8](#_Toc104725613)

[3.1.3 系统功能模块划分 8](#_Toc104725614)

[3.2 可行性分析 9](#_Toc104725615)

[3.2.1 技术可行性分析 9](#_Toc104725616)

[3.2.2 经济可行性分析 10](#_Toc104725617)

[3.2.3 实施可行性分析 10](#_Toc104725618)

[3.3 本章小结 10](#_Toc104725619)

[第4章 数据库设计及文件共享平台详细设计 11](#_Toc104725620)

[4.1 数据库设计 11](#_Toc104725621)

[4.1.1 数据库的概念结构 11](#_Toc104725622)

[4.1.2 数据库的逻辑结构 12](#_Toc104725623)

[4.2 系统整体流程设计 13](#_Toc104725624)

[4.2.1 服务端主线程流程 13](#_Toc104725625)

[4.2.2 服务端子线程流程 15](#_Toc104725626)

[4.2.3 客户端主线程流程 16](#_Toc104725627)

[4.2.4 客户端子线程流程 17](#_Toc104725628)

[4.3 各模块具体设计 18](#_Toc104725629)

[4.3.1 登录模块 18](#_Toc104725630)

[4.3.2 目录及文件操作模块 19](#_Toc104725631)

[4.3.3 文件传输模块 21](#_Toc104725632)

[4.4 本章小结 23](#_Toc104725633)

[第5章 系统测试 24](#_Toc104725634)

[5.1 系统功能模块测试 24](#_Toc104725635)

[5.1.1 登录模块测试用例 24](#_Toc104725636)

[5.1.2 文件及目录操作模块测试用例 24](#_Toc104725637)

[5.1.3 文件传输模块测试用例 25](#_Toc104725638)

[5.1.4 日志记录模块测试用例 26](#_Toc104725639)

[5.2 本章小结 26](#_Toc104725640)

[总 结 27](#_Toc104725642)

[参考文献 28](#_Toc104725643)

[致 谢 29](#_Toc104725644)

第1章 绪 论

1.1 研究背景

如今，人们对网络的依赖性越来越大，通过网络可以寻找、共享各种资料等。文件共享平台的出现能够很大程度的满足人们在共享资料和远程传输方面的需求。文件共享平台是人们认识世界、进行信息交换的一个重要途径，它可以使用户通过互联网连接到服务器，进行异地文件的上传、下载，方便用户共享数据文件，同时它具有高安全性和高效率性，是用户学习和生活中不可或缺的工具。不管是由于学习或是工作，又或是生活和娱乐需求，很多人都将自己的文件做一个存储备份，例如U盘、移动硬盘等。但是，U盘体积虽小，却容易遗失；移动硬盘体积较大，携带起来却十分麻烦，如此就必须有一种新型的替代产物，可以消除移动硬盘所带来的不便，基于此本设计所实现的文件共享平台能很好地克服这一缺点，通过将用户的数据上传到文件共享平台，用户登录互联网，可以轻松地在不同计算机之间进行文件的共享，所以文件共享平台的存在和发展将具有现实意义。

1.2 研究现状

目前，网络通讯技术已广泛地应用于各个领域，特别是网络文件系统、路由表管理、电子邮件、远程登录、文件传输、客户信息管理、服务台系统、流媒体传输、及时信息处理和社区聊天室等等。中国的发展日新月异，人们对网络服务的种类、服务的需求也在不断增加，因此，必须要有一套适合广大用户使用的网络服务器。当前，网络业务正朝着稳定、易于维护、易于管理、安全、多用户、高速度发展。因此，如何建立一个稳定、可靠、安全、性能优良的网络服务器平台，是当前网络技术研究的热点。所以，在未来的网络服务器通讯软件开发中，该系统的设计将会发挥很大的作用。不管是哪一种服务器，都需要能够和客户端建立端到端的数据通信，能够分析出客户端发来的指令，并且能够按照客户端的指令进行服务。

万物皆文件，在网络中传输的无非就是一些文件流数据，而提到文件在网络中进行传输，就不得不提到FTP文件传输协议，FTP协议是Internet最早提供的服务之一，截止到目前仍然被广大用户使用着。而本论文采用了私有的应用层传输协议，它和FTP协议同为应用层协议，其实都是为了处理特定的网络服务细节，解决某一类应用问题，规定应用进程在通信时所遵循的规则。

1.3 研究目的与意义

随着Linux操作系统的普及，基于Linux环境的文件共享平台的需求与日俱增，它在实际的生产中具有广阔的应用前景，它的出现对于促进互联网的快速发展具有重大的意义。通过对Linux环境下的文件共享平台的设计与开发，可以使读者更好地理解Linux操作系统，增强其实际操作的能力。Linux是一个开放源码的平台，人们可以方便的编写和加载自己的功能模块，真正做到人机的交互。另一方面是可以使人们对文件传输系统的框架有一个总体的认识。

随着现代信息技术的高速发展，网络通信技术在现代科技领域的应用日益深入，网络通信服务需求也越来越多，这样就更需要有一个良好的服务器平台来满足用户日益增长的网络服务需求。因此研究设计服务器相关通信软件已经成为网络发展的一个热门课题。本论文就是在此背景下研究并设计一个网络文件传输服务器平台。本课题系统地研究文件传输服务器整体设计流程，加深了对服务器通信软件设计的理解。对于今后研发和设计出功能更加强大的服务器具有一定的理论和实践意义。基于此，本论文详细讲述了一个开发基于私有协议的文件共享平台所应具备的主要功能以及设计与实现的过程。

1.4 设计目标

基于私有协议的文件共享平台采用C/S架构，其中服务端包含一个主线程和若干子线程，根据衡量连接客户端数量的能力来设定子线程的数量，主线程用来接收客户端请求并完成创建文件、删除文件、显示目前所处路径等一些瞬时响应需求，子线程用来完成上传下载这类需要耗费一定时间的需求，客户端同理。本论文所设计的文件共享平台，基于LINUX进行开发，除了实现用户查看、上传、下载、删除文件、创建文件、查看当前路径等一些基本操作外，还可以实现秒传、断点续传、零拷贝等技术，提高了下载效率，整个文件系统采用虚拟文件系统，方便集群间的同步，为分布式做准备。

1.5 论文章节安排

本文主要分为5个章节来阐述文件共享平台。

第1章介绍系统的研究背景、研究现状、研究目的与意义、设计目标。

第2章对Linux开发环境、socket网络编程、线程池技术以及私有的应用层传输协议进行了简要介绍。

第3章主要对文件共享平台在客户端以及服务端的需求进行了分析，简要介绍了各模块的功能，并对系统的开发进行了可行性分析。

第4章先对系统的数据库设计进行了介绍，分别从数据库的概念结构与逻辑结构进行了阐述，并对系统的整体设计流程和各功能模块的实现过程进行了介绍。

第5章主要对系统的各项功能进行了测试。

第2章 相关技术简介

2.1 Linux简介

Linux是一个类UNIX的自由和开放源码操作系统。Linux操作系统内容是由Linus Torvalds于1991年10月5号发行的。Linux严格意义上是单指操作系统内核，而现在Linux通常被用来指基于Linux的完整操作系统，而内核被称为Linux内核。

Linux起初只支持英特尔的x86架构的计算机。随着Linux的发展，现在的Linux被移植到了许多电脑的硬件平台上，远远超过了其它的操作系统。Linux能够在诸如大型计算机和超级计算机等其它大型平台上运行。超过90%的全球最高速的500台超级计算机都在运行Linux发行版或变种，其中速度最快的10台超级计算机运行的都是基于Linux内核的OS。Linux也被广泛地用于诸如手机，平板电脑，路由器，电视和电子游戏设备等的嵌入式系统。目前被人们在手机上广泛使用的安卓系统，就是建立在Linux内核上的。此外，苹果的MacOS和iOS都是基于开源的类UNIX系统Darwin进行开发的。

2.2 socket简介

socket通常也称作“套接字”，应用程序通常通过“套接字”向网络发出请求或者应答网络请求。Server Socket用于服务器端，当建立网络连接时，将会使用socket，当连接成功时，应用程序两端都会产生一个socket实例，操作此socket实例从而实现所需要的会话。在一个网络连接中，套接字是平等的，不会因服务端和客户端的原因而造成不同级别。socket能够提供数据的接收和发送，它定义了一种叫做socket的变量，在传输数据时，首先建立一个socket，然后通过send等方法对准某IP/端口来传输数据，而接收端也要建立一个socket，把这个socket绑定到一个IP/端口上，所有的数据都会像读取文件中的数据那样通过recv等函数读取。

2.2.1 socket的创建

socket数据的传输是一种特殊的I/O，socket拥有一个类似于打开文件的函数调用socket()，它会返回一个整型的socket描述符，随后的数据传输，建立连接等操作都是通过该socket实现的。

通常有两种socket:流式SOCK\_STREAM(SOCK\_STREAM)和数据报式SOCK\_DGRAM(SOCK\_DGRAM)。流式是一种面向连接的socket，对应于面向连接的TCP服务应用，数据报式socket是一种无连接的socket，对应于无连接的UDP服务应用。调用套接字函数时，它返回一个与文件描述符相似的句柄。

socket函数的原型是：int socket(int domain, int type, int protocol)；

socket函数用于创建一个socket设备。调用该函数时需要指定通信的协议域、套接字类型和协议类型。根据选择TCP或者UDP有着固定的写法，type参数指定socket的类型：SOCK\_STREAM或SOCK\_DGRAM，protocol一般被赋予0。socket函数的返回值是一个非负整数，就是指向内核socket设备的文件描述符。socket描述符是一种指向描述符内部数据结构的指针。实际上建立一个socket就意味着为一个socket数据结构分配存储空间。

2.2.2 socket常用函数

socket接口是TCP/IP网络的API，socket接口定义了许多函数或例程，可以用它们来开发TCP/IP网络上的应用程序。socket编程中常用到的函数有：

函数bind():用于给套接字赋予一个本地协议地址（即IP地址加端口号），函数定义为int bind(int sockfd, const struct sockaddr \*addr, socklen\_t addrlen)。

函数listen()：使TCP服务端开启监听，服务端在开启了listen之后，就可以开始接受客户端连接了。函数定义为int listen(int sockfd, int backlog)，一旦启用了listen之后，操作系统就知道该套接字是服务端的套接字，操作系统内核就不再启用其发送和接收缓冲区，转而在内核区维护两个队列结构：半连接队列和全连接队列。半连接队列用于管理第一次握手的连接，全连接队列用于管理已经完成三次握手的连接。backlog在有些操作系统中用来指明半连接队列和全连接队列的长度之和，一般填一个正数即可。如果队列已经满了，那么服务端受到任何再发起的连接都会直接丢弃（大部分操作系统中服务端不会回复RST，以方便客户端自动重传）。

函数connect():客户端使用connect来建立和TCP服务端的连接。函数定义为int connect(int sockfd, const struct sockaddr \*addr, socklen\_t addrlen)，客户端调用connect可以不使用bind来指定本地的端口信息，这时客户端就会随机选择一个临时端口号来作为源端口。调用connect预期是完成TCP建立连接的三次握手。如果服务端未开启对应的端口号或者未监听，则只能收到一个RST回复，并且报错返回的内容是"Connection refused"。

函数accept()：accept函数由服务端调用，用于从全连接队列中取出下一个已经完成的TCP连接。函数定义为int accept(int sockfd, struct sockaddr \*addr, socklen\_t \*addrlen)，如果全连接队列为空，那么accept会陷入阻塞。一旦全连接队列中到来新的连接，此时accept操作就会就绪，这种就绪是读操作就绪，所以可以使用select函数的读集合进行监听。当 accept执行完了之后，内核会创建一个新的套接字文件对象，该文件对象关联的文件描述符是accept的返回值，文件对象当中最重要的结构是一个发送缓冲区和接收缓冲区，可以用于服务端通过TCP连接发送和接收TCP段。

2.2.3 socket工作过程

关于socket的工作方式，首先从服务器端开始，服务器端先将socket初始化，然后与端口绑定(bind)，对端口进行监听(listen)，调用accept阻塞，等待客户端连接。此时，如果有客户端对socket初始化并连接服务器(connect)，客户端与服务器端的连接就建立了。

TCP通信流程图如图2.1所示：

socket

connect

send

bind

socket

recv

close

send

recv

accept

listen

close

图2.1 TCP通信流程图

2.3 线程池技术

在使用多线程技术时，难免会出现创建了多个线程的情况，线程过多会带来调度开销，进而影响缓存局部性和整体性能，线程执行任务时间较短，大部分时间都用在创建和销毁线程中，大大降低了系统的效率。线程池技术避免了在处理短时间任务时创建与销毁线程的代价，线程池维护着多个线程，等待着监督管理者分配可并发执行的任务，可以想象把所有使用的线程都放到一个容器中，池中的的线程可以重复利用，从而省去了频繁创建线程对象的操作，节省了大量的时间和资源，降低了资源消耗，提高了响应速度和线程的可管理性。

线程池的原理类似于操作系统中的缓冲区的概念。线程池中会先启动若干数量的线程，这些线程都处于睡眠状态。当客户端有一个新的请求时，就会唤醒线程池中的某一个睡眠的线程，让它来处理客户端的这个请求，当处理完这个请求之后，线程又处于睡眠的状态。线程池能很高地提升程序的性能。比如有一个省级数据大集中的银行网络中心，高峰期每秒的客户端请求并发数超过100，如果为每个客户端请求创建一个新的线程的话，那耗费的CPU时间和内存都是十分惊人的，如果采用一个拥有200个线程的线程池，那将会节约大量的系统资源，使得更多的CPU时间和内存用来处理实际的商业应用，而不是频繁的线程创建和销毁。

线程池的工作流程：服务器创建子线程，子线程去队列里取任务，由于一开始队列为空，所以所有的子线程都睡觉，睡在条件变量cond。客户端connect，主线程accept后将得到的描述符new\_fd放入队列，放之前需要加锁，放入队列后，执行signal函数，这样就会唤醒一个子线程，子线程被唤醒后，就去队列里拿描述符new\_fd，拿到后，给客户端发送文件，这是一个经典的生成者消费者模型，其中主线程是生成者，子线程是消费者。线程池总体流程图如图2.2所示：

主线程启动

定义数据结构

初始化线程池

加锁

子线程启动

启动线程池

主线程启动listen监听客户端请求

从队列中取出socket描述符，传输文件

子线程睡在条件变量中

将socket描述符插入到队列，唤醒子线程

判断队列是否为空

Y

Y

是否收到主线程通知

N

N

图2.2 线程池流程图

2.4 私有应用层传输协议

当发送多个文件给对端时，数据会被放到缓冲区里，所以对端需要具备对缓冲区中的数据进行分割与区分的功能，既需要对同一个文件的文件名与文件内容进行区分，又需要将不同文件的文件名文件内容区分开，又由于像音频视频在网络中传输时，它们的文件流中含有大量结束符，所以不可以用结束符来划分。

为了处理此类特定的网络服务细节，需规定应用进程在通信时所遵循的规则，基于此定义了一个结构体变量，它类似于变长结构体，如下所示：

//带有状态码的火车，模仿http协议中的状态码信息，用以辨别当前指令类型

typedef struct{

int data\_len;//存放buf，也就是火车车厢中数据长度

int state;//状态码，内容为下列枚举command中的数据

char buf[1000];//火车车厢长度

}train\_state\_t;

可以形象的比喻它为小火车，在这个火车里有火车头与火车车厢，其中火车头data\_len记录了火车车厢buf里边存了多少数据，所以当要发送数据给对端的时候是一列火车一列火车的发，这样对端会先接收长度，再按长度接收文件内容，从而实现控制存取。除了这些，在这个结构体中还会设置一个变量state，代表这列火车的功能是什么，是上传还是下载等操作，否则来回send、recv的次数会很多，不易阅读且代码量会增加。

枚举结构体如下所示：

enum command{

REGISTER,

LOGIN,

LS,

CD,

PWD,

REMOVE,

GETS,

PUTS,

TOKEN,

};

2.5 本章小结

本章对系统开发的相关技术进行了简要介绍，包括系统的开发环境、socket网络编程、线程池技术以及私有的应用层传输协议，在系统开发之前，对涉及到的相关技术进行分析是十分必要的。

# 第3章 系统需求与可行性分析

3.1 需求分析

文件共享平台主要为用户提供一块网络上的存储空间，在这块空间中，用户可以像在本地计算机上操作自己硬盘空间中的文件一样随意操纵存储空间内的文件或目录。本论文所设计的系统使用C语言在Linux环境下进行开发，通过Xshell连接Ubuntu操作系统，使用Navicat连接Ubuntu的Mysql数据库，并使用Vim、Vscode等工具编写代码。

3.1.1 用户需求分析

用户通过客户端可以自己注册账户，自定义账号密码，输入正确的账号密码之后连接到服务端。对于每个用户来说，他们所看到的文件是独立的，用户可以查看当前所处的路径、创建目录、然后进入到相应目录以及在相应目录创建或删除文件等基本操作，用户可以看到文件及目录的类型，可以从本地上传文件到服务端，也可以从服务端下载文件到本地，还可以实现边上传边查看文件等功能。

3.1.2 服务端功能需求分析

服务端需要对不同用户的账号信息进行存储管理的操作，需要对同一用户及不同用户的文件进行有序的存储和管理，同时要避免出现不同用户的不同文件具有相同文件名的情况。对于服务端来说，所有用户的文件都存储到一个目录里，每一个用户的目录不会真正存在磁盘上，而是由数据库记录每一个用户的路径情况，每一个用户的文件或目录采用数据库的表进行存储。服务端需要实现上传下载与其它瞬时响应操作相分离，这样当用户在下载文件时，也能进行其它像查看文件、删除文件及显示当前所在路径这些基本操作，这是非常有必要的，不然的话用户在下载文件时就只能一直卡在那里，等待下载完成才能进行其它操作，这是非常糟糕的。服务端需要对用户的操作时间、连接服务端的时间及操作记录有一个日志记录功能，且服务端需具备同时连接大量客户端的能力。

3.1.3 系统功能模块划分

1. 登录模块

登录模块包含用户注册和密码验证功能，服务端核对客户端发来的用户名和密码，若有此用户则直接登录，若没有此用户则创建后再登录，采用sha512加密算法作为登录验证方法。

1. 目录操作模块

目录操作模块包含创建目录、删除目录、进入目录、列出相应目录文件以及显示目前所在路径等功能。

1. 文件操作模块

文件操作模块包含显示文件大小、显示文件类型和删除文件等功能。对于一名用户来说，查看文件大小、类型和删除文件这些基本操作来说是必不可少的。

1. 文件传输模块

文件传输模块包含上传文件、下载文件、秒传以及断点续传等功能。断点续传是大部分客户有用到的功能，所谓的断点续传就是可以在断点的地方继续传，这是为了解决TCP协议中的重传问题，因此极大地方便了用户上传，下载文件，对于一个大文件来说，若用户的文件没有下载完，那当他下一次下载这个文件时就只能重头开始下载，这会非常浪费时间，因此断点续传功能是很有必要的。另一方面，由于文件共享平台会容纳大量用户的文件，当某用户要上传其它用户已经上传过的文件时，就无需进行真正的上传操作，针对这一需求，秒传功能也是很有必要的。

1. 日志记录模块

日志记录模块记录客户端请求信息及客户端连接时间，记录客户端操作记录及操作时间，方便服务端管理。

文件共享平台设计模块图如图3.1所示：

文件共享平台

登录模块

目录操作模块

文件操作模块

文件传输模块

日志记录模块

用户登录

用户注册

创建目录

进入目录

列出目录下所含文件

删除文件

上传文件

下载文件

断点续传

显示目前所在路径

记录客户端请求信息

记录客户端操作记录

显示文件大小

秒传

图3.1 文件共享平台设计模块图

3.2 可行性分析

在系统开发之前，可行性分析是评价系统是否能够按时完成的一个关键环节。可行性分析不能一概而论，它是一个综合、系统的分析方法，可以为项目决策提供依据，总体上包括技术可行性、经济可行性、实施可行性。

3.2.1 技术可行性分析

此文件共享平台采用C/S架构，利用Mysql数据库进行数据存储，使用C语言进行开发。在该架构中，数据库应用的费用相对较低，操作简便，语言技术较为成熟，对于本系统而言，在技术上是可行的。

3.2.2 经济可行性分析

在开发一个系统前，应该充分考虑系统的开发投资和所产生的效益。本文所采用的 MySQL数据库和Linux操作系统均是开源的，C语言是一种易于操作的跨平台语言。同时，数据和信息的共享存在于生活和工作的各个方面，从经济性上来说，本系统是切实可行的。

3.2.3 实施可行性分析

文件共享平台在网络上建立之后，用户可以进行上传、下载等操作，并管理服务器端存储的目录与文件，同时具有成本低和使用维护简单的特点，在管理人员对平台的结构有一定程度上的了解的情况下，日后简单的管理是完全可以轻松达到的。所以本系统在实施上是可行的。

3.3 本章小结

本章对文件共享平台的系统功能进行定义，对客户端和服务端的需求进行了详尽的分析，阐述了各模块的功能，对系统的实现进行了可行性分析，为下一步的开发奠定了基础。

# 第4章 数据库设计及文件共享平台详细设计

4.1 数据库设计

每一个用户的目录不会真正存在磁盘上，而是由数据库记录每一个用户的路径情况，每一个用户的文件或目录采用数据库的表进行存储。每个文件的文件内容以其MD5码存在磁盘上，并以MD5码进行命名，所有用户的文件都存在一个目录里，各自用户只能看到自己的文件，不能看到其他人的文件。采用此种虚拟文件系统的好处是为分布式做准备，方便集群间的同步，因为数据库有主从同步，这些数据就可以很方便的同步到另一个数据库里。

此文件共享平台通过navicat远程连接ubuntu中的mysql数据库。共有4张表，各表的名字分别为：用户信息表，虚拟文件系统表，文件MD5表，日志记录表。

4.1.1 数据库的概念结构

根据第3章的系统功能需求分析，可简要地将需求实体抽取成如下关系，基本实体有用户实体、文件实体、MD5实体、日志实体。实体之间的关联关系如图4.1所示。

MD5

文件大小

MD5值

日志

操作内容

用户id

用户名

操作日期

文件

用户名

code

precode

文件大小

类型

用户id

用户

盐值

密文

用户名

用户id

操作

对应

记录

图4.1 文件共享平台E-R图

4.1.2 数据库的逻辑结构

文件共享平台数据库的逻辑结构根据上述概念结构的分析，共创建抽取出4张数据库表。

用户信息表记录了各用户的一些账号信息，包括用户id，用户账号名，盐值，以及通过盐值和密码生成的密文。用户信息表如下表4.1所示：

表4.1 用户信息表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段** | **类型** | **是否为空** | **主键** | **备注** |
| user\_id | int | 否 | 是 | 自动增长 |
| user\_name | varchar | 是 | 否 | 用户账号 |
| salt | varchar | 是 | 否 | 盐值 |
| passwd | varchar | 是 | 否 | 密文 |

虚拟文件系统表中每一条记录代表一个目录或文件，code字段作为主键，precode字段代表当前目录或文件的上一级目录的code，例如precode=0代表当前文件在根目录下，precode=2代表当前文件的上一级目录的code是2，所以根据code和precode就可以实现一个无穷极的目录。虚拟文件系统表如下表4.2所示：

表4.2 虚拟文件系统表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段** | **类型** | **是否为空** | **主键** | **备注** |
| code | int | 否 | 是 | 自动增长 |
| precode | int | 是 | 否 | 等价于上级目录的code |
| file\_name | varchar | 是 | 否 | 该文件所属用户的用户名 |
| user\_id | int | 是 | 否 | 用户id |
| file\_type | varchar | 是 | 否 | 判断是目录还是文件 |
| file\_size | int | 是 | 否 | 文件大小 |

文件MD5表记录了每个文件的MD5值和文件的大小等信息，为后续的秒传功能服务。文件MD5表如下表4.3所示：

表4.3 文件MD5表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段** | **类型** | **是否为空** | **主键** | **备注** |
| md5sum | varchar | 否 | 是 | 文件的MD5值 |
| file\_size | int | 是 | 否 | 文件大小 |

日志记录表记录了每个用户的操作记录以及操作时间等信息。日志记录表如下表4.4所示：

表4.4 日志记录表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **字段** | **类型** | **是否为空** | **主键** | **备注** |
| user\_id | int | 否 | 否 | 用户id |
| user\_name | varchar | 是 | 否 | 用户名 |
| operate | varchar | 是 | 否 | 该用户的操作记录 |
| object | varchar | 是 | 否 |  |
| op\_time | datetime | 否 | 是 | 操作时间 |

4.2 系统整体流程设计

本系统采用C/S架构及socket函数库，实现文件的网络传输，服务端与和客户端分别建立套接字，并绑定到某个端口上，同时对socket进行初始化，服务端调用socket的accept方法，实现套接字的侦听，等待客户端的连接请求，客户端调用socket的connect方法建立连接。为了使用户可以边下载边查看目录文件，考虑瞬间响应命令由主线程负责，上传下载命令由子线程负责，客户端子线程通过TOKEN序列实现服务器认证。

4.2.1 服务端主线程流程

启动线程池

收到客户端请求，得到socket描述符

登录成功，维护客户端状态信息，初始化Client\_State\_t结构体，并发送TOKEN序列给客户端

通知客户端登录失败

接收客户端的请求，主线程执行相应操作

将客户端发来的用户名和密码采用sha512算法加密后存储到数据库中

将客户端发来的TOKEN序列与客户端状态信息队列结点中的TOKEN比较，并将客户端信息初始化为相匹配的队列结点信息后插入到队列中

判断所执行操作

判断密码是否正确

将socket描述符放到任务队列，交给子线程执行

注册

登录

TOKEN验证

是

否

图4.2 服务端主线程流程图

服务端主线程先将线程池初始化，然后启动线程池，此时会创建若干子线程，子线程一经创建就睡在条件变量中，等待被唤醒，同时主线程用epoll监听客户端，一旦有客户端连接，服务端就会得到一个socket描述符，根据火车中的state变量判断客户端要进行的操作，如果是注册则按照相应算法将账号信息存储在数据库中，如果是登录则使用结构体变量记录此客户端的状态信息，结构体如下所示：

//针对每个客户端需要有状态维护

typedef struct client\_State\_s{

int new\_fd;//对应客户端连接

char path[100];//路径，为cd与pwd服务

int code;//code为0代表是根

int code\_init;//多存一个根目录的code，在登录注册后这里就不变了

char token[100];//一旦登录，随机生成一个token并发送给客户端

struct client\_State\_s \*next;

}Client\_State\_t;

一旦客户端登录成功，则对上述结构体初始化，将path初始化为/,code初始化为0，随机生成一个字符串序列赋值给token，并发送这个字符串序列给客户端，目的是为接下来的客户端子线程实现服务器认证。初始化后将此结构体插入到客户端状态队列。此时客户端就可以发送需要执行的操作给服务端，服务端通过train\_state\_t结构体中的state状态码，也就是第二章说的小火车，看得到的是LS，PWD，REMOVE，CD中的哪一个，就走到对应函数，执行相应任务。结构体如下所示：

typedef struct{

int data\_len;//存放buf，也就是火车车厢中数据长度

int state;//状态码，内容为下列枚举command中的数据

char buf[1000];//火车车厢长度

}train\_state\_t;

state状态码的取值如下列枚举结构体所示：

enum command{

REGISTER,

LOGIN,

LS,

CD,

PWD,

REMOVE,

GETS,

PUTS,

TOKEN,

};

若服务端主线程接收到的状态码是TOKEN，则说明客户端要进行服务器认证，且需要执行上传下载操作，服务端得到的token序列一定是客户端子线程发给服务端的，这样两端的子线程来执行上传下载操作，因为服务端对于每一个登录的用户都会发送一个token序列给客户端并为之初始化结构体Client\_State\_t中的token字符串，所以客户端要执行上传下载操作时会将这个token序列传递给子线程，子线程再将此token发给服务端，服务端将客户端发来的token序列与Client\_State\_t客户端状态信息队列结点中的token比较，并将客户端信息初始化为与之相匹配的队列结点信息后插入到队列中，这样服务端就会知道此时的用户就是之前需要执行上传下载的用户。TOKEN验证之后，两端的子线程就可以执行上传下载操作了。

4.2.2 服务端子线程流程

子线程被创建

睡在条件变量中

主线程将任务插入到任务队列，通过signal函数唤醒子线程

子线程拿到任务

通知客户端上传成功

判断是上传还是下载

判断是否符合秒传

是

否

判断是否为断点续传

与客户端进行传输工作

根据客户端发来的文件大小执行相应的偏移再发送文件

发送文件到服务端

是

否

上传

下载

图4.3 服务端子线程流程图

子线程一经创建就立刻睡在条件变量中，等待主线程或其它完成任务的子线程通过signal函数唤醒。当主线程将任务放到队列时，就会唤醒一个子线程，因为只有上传下载这两种任务才会教给子线程来执行，所以子线程拿到任务以后只需判断是上传还是下载，如果是上传，则判断是否符合秒传条件，如果是下载，则判断是否符合断点续传。秒传和断点续传的具体实现方法在下文会详细介绍。

4.2.3 客户端主线程流程

客户端启动

读取用户输入的命令

将用户名和密码发送到服务端

判断是否为瞬时响应命令

注册

是

判断是注册还是登录

收到服务端发来的TOKEN序列，并开始读取用户输入的命令

是

否

将用户名和密码发送到服务端

将TOKEN序列发送给子线程实现服务器认证，子线程执行上传下载操作。

判断是否登录成功

主线程发送请求到客户端，通过recv函数等待接收数据

登录

否

图4.4 客户端主线程流程图

客户端启动后，会出现登录还是注册的选项，输入用户名和密码后，客户端会将其发给服务端，服务端根据相应算法到用户信息表中查看是否匹配，若匹配则通知客户端登录成功并发送一个token序列给客户端，客户端登录成功后，就可以进行一系列的基本操作，客户端输入要进行的操作后，如果是瞬时响应命令，则由客户端的主线程发送给服务端进行通信，如果是上传下载这种命令，则客户端主线程将token值发送给客户端的子线程，子线程根据token值与服务端建立连接再进行通信。

4.2.4 客户端子线程流程

收到主线程发来的TOKEN序列等信息

通过TOKEN实现服务器的认证

不符合秒传条件，执行上传操作

判断是上传还是下载

服务器通知秒传成功，则打印上传成功，无需进行真正的上传操作

获取本地文件大小，并将其发送给服务端，等待接收服务端文件内容

上传

下载

先不上传文件内容，而是发送文件的MD5值给服务端，等待服务器通知

图4.5 客户端子线程流程图

子线程只会执行上传下载操作，在此之前都会睡在条件变量中，当被唤醒时，子线程会拿到从主线程发来的信息，子线程会拿到一个指向结构体的指针，这个结构体是：

typedef struct{

char token[256];

int state;

int new\_fd;

char file\_name[64];//文件名

}Thread\_Info\_t;

子线程拿到其中的token值与描述符new\_fd之后与服务端建立连接，根据state判断是上传还是下载，如果是上传，则先将文件的MD5值发送给服务端，服务端根据相应的操作执行后判断是否符合秒传的条件，若符合则服务端直接通知客户端上传成功而无需进行真正的上传操作，若客户端要执行的操作是下载，则判断本地有没有这个文件，如果有这个文件则获取这个文件的大小再发送给服务端，然后等待接收数据。

4.3 各模块具体设计

4.3.1 登录模块

首先要介绍一下linux的登陆验证方法，它是将输入的密码进行相应编码加密后与/etc/shadow文件中对应的密码部分进行比较，如果相同则验证通过，如果不同则表明密码错误，但是问题是如何将用户输入的密码加密然后进行比较。为了提高安全性，Linux引入了salt，所谓salt即为一个随机数，引入的时候为一个12bit的数值，当用户设置密码时，会随机生成一个salt与用户的密码一起加密，得到一个加密的字符串(salt以明文形式包含在该字符中），存储到密码文件中。crypt函数可以将用户输入的密码和salt一起适应某种算法进行加密，该加密后的字符串就是需要与密码文件中密码对比的加密字符串。

基于此，在本论文所设计的文件共享平台中，就是采用的此种sha512加密算法作为登录验证的过程，不同的是，本项目通过数据库存储用户名，salt值(采用随机字符串生成)，密码(密文形式存储)，密码验证的方式是客户端发送用户名跟密码给服务端，服务端根据用户名从表中取出盐值，再将盐值与密码进行crypt生成密文，再去数据库的用户信息表中进行查询，若能找到与之相等的密文，则登录成功。登陆验证流程图如图4.6所示：

服务端接收到客户端发来的用户名和密码

服务端通过用户名从数据库的用户信息表中取出对应的盐值salt

通过crypt函数将salt与客户端发来的密码加密生成密文，再将此密文与用户信息表中的密文进行核对

判断核对是否成功

是

登录成功，初始化客户端状态信息

通知客户端登录失败

否

图4.6 登陆验证流程图

4.3.2 目录及文件操作模块

服务端针对每一个客户端都会有一个相应的状态维护信息，一旦有客户端连接，服务端就会对其状态进行初始化，并将其加入到队列中。客户端状态维护信息的结构体如下所示：

typedef struct client\_State\_s{//针对每个客户端需要有状态维护

int new\_fd;//对应客户端连接

char path[100];//路径，为cd与pwd服务

int code;//code为0代表此用户当前工作在根目录下

int code\_init;//多存一个根目录的code，在登录注册后这里就不变了

char token[100];//一旦登录，随机生成一个token并发送给客户端

struct client\_State\_s \*next;

}Client\_State\_t;

一旦有客户端登录，就会初始化上述结构体，并将其加入到队列中。

1. pwd显示当前所处路径功能

pwd功能通过上述结构体中的path字符串来实现，当客户端登陆时，其值初始化为/，一旦用户通过cd进入到某个目录中，服务端就会对path字符串进行相应的修改维护，这样当用户调用pwd来显示当前所在的一个路径时，只需将path字符串中的内容打印出来即可。

1. cd进入对应目录功能

cd功能通过上述结构体中的code来实现，因为本项目的文件系统采用的是虚拟文件系统，由数据库记录每一个用户的路径情况，每一个用户的文件及目录采用数据库的表进行存储，每个文件或目录都是数据库中虚拟文件系统表中的一条记录，例如，在这张数据库表中，字段precode若为0则代表此条目录或文件在根目录下，若precode为1则代表此目录或文件的上一级目录是code为1的那条目录。然而，在客户端状态维护信息结构体中，code为0则代表此用户当前处在根目录下，因此当cd到某个目录时，需要从数据库中将这个目录的precode的值取出然后赋值给状态维护信息结构体中的code，同时也要修改path字符串的值。之所以要这样做，也是为下面的ls显示当前所处目录下有哪些文件或目录来服务的。

cd分为3种情况：

1. 情况1，返回根目录

因为在上述结构体中，code\_init代表的是该客户端一开始登录时的code，这个变量在登录后就不再改变了，所以在我们要返回根目录时，我们只需要将当前客户端的code置为code\_init。代码如下所示：

if (0 == strcmp(buf, "/")) //情况1，返回根目录

{

rp->code = rp->code\_init; //回归初始

memset(rp->path, 0, sizeof(rp->path));

rp->path[0] = '/';

}

1. 情况2，返回上一级目录

从数据库中查询当前目录code的precode并返回，代码如下所示：

sprintf(buf,"select precode from virtual\_filesystem where code=%d;",code);

将当前客户端结构体中的code赋值为上述从数据库中返回的precode。同时修改path的值，也就是删除path中最后一个/之后的内容，代码如下所示：

for (len = strlen(rp->path)-1; len >= 0; len--) //去除倒数第一个斜杠后得内容

{

if (rp->path[len] != '/')

{

rp->path[len] = '\0';

}

else

{

if (len != 0)

{

rp->path[len] = '\0';

}

break;

}

}

1. 情况3，进入对应目录

从数据库查询将要进入的目录所对应的precode值，并将此precode值赋值给此客户端状态结构体中的变量code，同时修改字符串path的值，这样就改变了当前用户的工作目录，数据库查询代码如下所示：

sprintf(buf,"select code from virtual\_filesystem where file\_name='%s' and precode=%d;",file\_name,code);

1. ls列出相应目录文件

根据当前code，查找precode为当前code的所有文件，并逐行打印。代码如下：

char buf[256] = {0};

sprintf(buf, "select file\_type,file\_name,file\_size from virtual\_filesystem where precode=%d;", rp->code);

1. remove删除文件

因为我们采用的是虚拟文件系统，所以我们需要删除数据库中的一条代表文件的记录，代码如下所示：

sprintf(buf,"delete from virtual\_filesystem where file\_name='%s' and precode=%d;",file\_name,code);

至于其它的一些像显示文件类型大小的操作，同理，我们只需根据文件名和用户id去数据库中查询即可。

4.3.3 文件传输模块

服务端与客户端进行文件的传输，在发送多个文件给对端的时候，由于实际发送的文件中可能是字符串，可能是音频，可能是视频，所以发送时，对方要知道多少数据，我们必须采用控制数据，这也就是第二章提到的应用层传输协议设计，这里叫其小火车，其中火车头data\_len记录了火车buf中到底装载了多少数据发到对端，如下所示：

typedef struct{//不要交换data\_len和buf的定义顺序

int data\_len;//存放buf，也就是火车车厢中数据长度

char buf[1000];//火车车厢长度

}train\_t;

根据这个类似于变长结构体的变量，我们就可以进行文件的传输。

服务器端发文件代码如下所示：

train\_t t;

//先发送文件名

t.data\_len=strlen(FILENAME);

strcpy(t.buf,FILENAME);

send(new\_fd,&t,4+t.data\_len);

//发送文件内容

int fd=open(FILENAME,O\_RDONLY);

ERROR\_CHECK(fd,-1,"open");

while(t.data\_len=read(fd,t.buf,sizeof(t.buf)))

{

send(new\_fd,&t,4+t.data\_len,0);

}

客户端接收文件代码如下所示：

//先接文件名

int data\_len;

recv(new\_fd,&data\_len,4);//每次先接 4 个字节，知道火车车厢的数据长度

char buf[1000]={0};

recv(new\_fd,buf,data\_len);

//接收文件内容

while(!0)

{

recv(sfd,&data\_len,4,0);//接文件内容长度

if(data\_len>0)

{

recv(sfd,buf,data\_len,0);//接文件内容

write(fd,buf,data\_len);//把文件内容写到文件里

}else{

break;

}

}

当接到火车头里边的值为零时，就知道文件内容接收完毕。

1. 秒传设计

当客户端要上传文件时，先不上传文件内容，而是将要上传到服务端的文件所对应的MD5值发送给服务端，服务端根据此MD5值去数据库中的MD5表中查询是否含有这个MD5值，若有则说明服务端本就存在这个文件（此文件可能是之前其它用户上传到服务端的），此时只需通知客户端秒传成功，然后再修改虚拟文件系统表中的数据即可，无需进行真正的上传操作。查询MD5表中是否有此MD5值的代码如下所示：

sprintf(buf,"select file\_size from file\_md5 where md5sum='%s';",md5);

秒传流程图如图4.7所示：

服务端接收到客户端发来的MD5值

从数据库MD5表中查询是否含有此MD5值

是

通知客户端秒传成功，在虚拟文件表中插入一条记录，无需进行真正的上传操作

开始接收文件内容，在虚拟文件表和MD5表中分别插入一条记录

否

图4.7 秒传流程图

1. 断点续传

在介绍断点续传之前，需要先对mmap映射此种零拷贝技术进行介绍，当向一个文件中读写数据时，需要调用read、write这类系统调用，它会先将用户态中的数据拷贝到内核态对应的文件对象的文件缓冲区中，这会导致用户态和内核态之间的切换，这是需要消耗时间的，特别当读写一个大文件时，需要频繁进行这种交换，所以传输效率就会降低，而mmap会在用户态中分配一块内存，它和内核态的文件缓冲区所映射在磁盘中的是同一块地址，这样就不会有用户态和内核态之间的切换，从而在传输大文件时节省了时间。mmap函数定义如下所示：

void \*mmap(void\* addr,size\_t len,int prot,int flags,int fildes,off\_t off);

mmap函数创建⼀个指向⼀段内存区域的指针，该内存区域与通过⼀个打开的⽂件描述符访问的⽂件的内容相关联。off⽤来改变⽂件中数据的起始偏移值，len指定可以访问的数据量，addr可以请求某个特定的内存地址，取值为零时则指针⾃动分配。prot参数⽤于设置内存段的访问权限，flags参数控制程序对该内存段所造成的影响

断点续传的具体实现方法：当客户端要从服务端下载文件时，先判断本地有没有这个文件，如果有，则获取本地这个文件的大小，然后将这个文件的大小发送给服务端，如发送的是gets file 1000,如果本地没有这个文件就发送gets file 0，服务端根据客户端发来的文件大小对文件内容偏移后再发送给客户端，只需将mmap的最后一个参数off改为客户端发来的文件大小即可实现从文件内容的第off个字节处开始传输数据。

4.4 本章小结

本章在第3章需求分析的基础上对该系统的数据库进行了设计，从数据库的概念结构和逻辑结构两方面进行了阐述，并详细介绍了该系统的整体设计流程，分别从服务端与客户端两个层面进行了细致的讲述，最后又对各功能模块的实现过程进行了详细的介绍。

# 第5章 系统测试

5.1 系统功能模块测试

在完成服务器软件设计与编码之后，接下来的工作就是对服务器进行系统测试，看服务器是否满足第3章所提出的设计要求，各功能模块需要完全通过测试用例，同时也需要选取覆盖率高的测试用例。在此使用了Xshell来创建多个会话，从而实现多个用户连接到服务端。先启动服务端，输入./server ip地址 端口号 子线程数，客户端输入./client ip地址 端口号来连接到服务端。

5.1.1 登录模块测试用例

1. 注册功能

客户端输入register打开注册界面，输入用户名和密码，服务端先去检查数据库中是否存在这个账户，若不存在则修改数据库中用户信息表，增添一行记录，告知客户端登录成功。

用户注册功能测试表如表5.1所示：

表5.1 用户注册功能测试表

|  |  |  |  |
| --- | --- | --- | --- |
| **测试功能** | **测试数据** | **预期结果** | **测试结果** |
| 用户注册 | 用户名：chaiao | 注册成功 | 用户chaiao注册成功 |
| 用户注册 | 用户名：abc | 注册成功 | 用户abc注册成功 |
| 用户注册 | 用户名：chaiao | 注册失败 | 用户chaiao已存在 |

2. 登录功能

客户端输入login打开登录界面，输入正确的用户名和密码后，若功能完善则服务端根据sha512加密算法验证后通知客户端用户登录成功，若用户不存在则通知客户端登陆失败。

用户注册功能测试表如表5.2所示：

表5.2 用户登录功能测试表

|  |  |  |  |
| --- | --- | --- | --- |
| **测试功能** | **测试数据** | **预期结果** | **测试结果** |
| 用户登录 | 用户名：chaiao | 登录成功 | 用户chaiao登录成功 |
| 用户登录 | 用户名：abc | 登录成功 | 用户abc登录成功 |
| 用户注册 | 用户名：123 | 登录失败 | 用户123未注册 |

5.1.2 文件及目录操作模块测试用例

用户通过客户端连接到服务器后，若功能完善则可以执行创建目录、进入目录，显示当前路径、查看目录下文件、删除文件等基本操作。

文件基本操作功能测试表如表5.3所示：

表5.3 文件基本操作功能测试表

|  |  |  |  |
| --- | --- | --- | --- |
| **测试功能** | **执行操作** | **预期结果** | **测试结果** |
| 创建目录 | 输入mkdir dir1 | 目录dir1创建成功 | 创建成功 |
| 进入目录 | cd dir1 | 进入dir1目录 | 进入成功 |
| 显示当前所在路径 | 输入pwd | 打印当前路径/dir1 | 打印成功 |
| 列出所处目录文件 | Ls | 打印当前目录所含文件的文件名、文件类型以及文件大小 | 打印成功 |
| 删除文件 | remove | 删除文件 | 删除成功 |

5.1.3 文件传输模块测试用例

如文件传输功能完善，则用户可以从服务端下载文件到本地，也可以从本地上传文件到服务器。

文件传输功能测试表如表5.4所示：

表5.4 文件传输功能测试表

|  |  |  |  |
| --- | --- | --- | --- |
| **测试功能** | **执行操作** | **预期结果** | **测试结果** |
| 上传文件 | 用户chaiao输入puts file1 | 文件file1上传至平台 | 上传成功 |
| 下载文件 | 用户chaiao将本地文件file1删除，然后输入gets file1 | 文件file1下载到用户chaiao的本地 | 下载成功 |

服务器已经存储了用户chaiao的文件file1，并使用MD5值为其命名，此时用户abc登录并上传文件file1，文件file1在服务器中已经被用户chaiao上传过，若秒传功能完善则会瞬间上传成功。

秒传功能测试表如表5.5所示：

表5.5 秒传功能测试表

|  |  |  |  |
| --- | --- | --- | --- |
| **测试功能** | **执行操作** | **预期结果** | **测试结果** |
| 秒传 | 用户abc输入puts file1 | 瞬间上传成功 | 秒传成功 |

用户abc从服务端下载文件file2，在下载过程中与服务端断开连接，当用户abc重新连接服务端并下载文件file2时，若断点续传功能完善则会从断点的地方继续下载这个文件。

断点续传功能测试表如表5.6所示：

表5.6 断点续传功能测试表

|  |  |  |  |
| --- | --- | --- | --- |
| **测试功能** | **执行操作** | **预期结果** | **测试结果** |
| 断点续传 | 用户abc输入gets file2，输入quit断开连接，然后重新连接服务端输入getsfile2 | 用户abc的本地先得到file2.temp，重新下载后得到file2文件 | 断点续传成功 |

5.1.4 日志记录模块测试用例

日志记录功能测试表如表5.7所示：

表5.7 日志记录功能测试表

|  |  |  |  |
| --- | --- | --- | --- |
| **测试功能** | **执行操作** | **预期结果** | **测试结果** |
| 记录用户连接时间 | 用户登录 | 显示用户连接时间 | 记录成功 |
| 记录用户操作日期 | Ls、cd、remove等操作 | 显示用户操作日期 | 记录成功 |

5.2 本章小结

本章节主要对系统的全部功能模块进行了测试，并全部成功通过测试用例，且覆盖率较高，说明系统整体的功能完善。

# 总 结

文件共享平台基于私有的应用层协议进行设计，在linux操作系统下进行开发，编程语言使用c语言，实现数据共享。可以完成类似于百度网盘的一系列操作，如用户查看，上传下载，显示当前工作路径，查看对应目录下的文件，显示文件类型，文件大小，通过token序列实现边下载边查看目录文件，除此之外，为了提高文件共享平台的效率，又实现了秒传，断点续传功能，采用了零拷贝mmap映射技术，缩短了响应时间，同时，整个文件系统采用虚拟文件系统，为分布式做准备，方便集群间的同步。在实现此项目的过程中，也遇到了很多问题，比如由发送和接收速度不匹配造成的大文件发送异常，经过一系列的查阅相关文献和测试，找到了其中的问题，最后采用循环发送和循环接收来解决了这种协议异常。后面考虑到服务器总是要升级的，所以配备了有序的退出机制。

关于此文件共享平台，我有以下展望：实现多点下载功能，多点下载是指客户端通过多个socket描述符连接不同的服务器，将一份大文件拆分为几段，不同段从不同的服务器进行下载。在多点下载过程中，如果客户端断开，那么下次重新下载时，需要再次进行多点下载，所以客户端本地需要使用数据库存储多点下载每个位置下载到什么地方。

参考文献

[1]爽.Linux系统下计算机C语言的编程技巧探讨[J]. 电脑编程技巧与维护,2022,(03):40-42.

[2]朱澳,陈丽琼.基于Linux的FTP服务器设计[J]. 无线互联科技,2022,19(03):49-50.

[3]瞿小淦,唐孝国.基于C语言的计算机编程技术[J]. 计算机与网络,2021,47(16):39-40.

[4]于鑫.C语言下的计算机软件编程探讨[J]. 电子元器件与信息技术,2021,5(08):119-120.

[5]邓飞,蔡波.基于Linux平台的文件系统设计研究[J]. 信息与电脑(理论版),2020,32(20):98-99.

[6]仇正兰.Linux下单线程多连接网络编程的一种实现[J]. 工业控制计算机,2020,33(05):50-51.

[7]陈娟.基于Linux的文件传输系统的设计[J]. 信息与电脑(理论版),2019,31(22):69-70+73.

[8]李妍.消息队列在Linux线程或进程间通信中的应用[J]. 电子世界,2019,(17):146-147.

[9]董伟梁.Linux下使用SOCKET编程接口实现C/S通信[J]. 数码世界,2019,(05):40.

[10]陈亮平,罗南超.Linux下基于Epoll+线程池简单Web服务器实现[J]. 福建电脑,2019,35(04):24-27.

[11]王举辉.基于Linux内核的文件服务器模型的研究与构建[J]. 科技视界,2018,(28):154-155.

[12]周继宇.基于HTTP协议的多线程断点续传软件设计与实现[J]. 计算机产品与流通,2018,(09):129-130.

[13]王天伟.一种Linux系统下的线程通信方法[J]. 计算机应用与软件,2017,34(10):330-333.

[14]黄隽.Linux网络编程的研究[J]. 电脑编程技巧与维护,2017,(06):21-22.

[15]宋泽瑞.基于SOCKET编程接口的网络通信[J]. 中国新通信,2017,19(05):29.

# 致 谢

时光荏苒，岁月如梭，大学四年时光转瞬即逝，在这四年里，我学到了很多计算机的相关理论知识，并积累了一定的实践经验，这对我的未来职场有着莫大的帮助，我非常庆幸选择了计算机科学与技术专业，这使我结识了信工学院的老师们，我很热爱这个专业，并且学院的学习氛围也非常浓厚，老师们也很和蔼，所以在学习的过程中也非常轻松愉快。

目前，我也即将毕业了，经过这段时间的努力，毕业论文也是终于完成了。毕业论文的完成过程也是非常坎坷的，遇到了很多技术上的问题，如果没有老师的耐心教导和同学们的细心帮助，我的论文完成过程会更加的艰难。所以首先，我要感谢我的毕设指导教师刘雅军老师，在我完成毕设的过程中提供了很大的帮助，每当迷茫时，遇到困难时，刘老师都会给予指导性的意见，且在完成毕设论文之前给予了一定意见使我有了明确的写作思路，随着不断的修改与完善，最终定稿。其次，我也要感谢我的同学和宿舍舍友们，积极向上的学习氛围和安静的学习环境对于我的毕业论文写作也是必不可少的，同学们积极完成毕业论文的模样也给了我很大的动力。最后我要感谢我的母校，为我们提供了良好的学习环境和教学资源，每当遇到问题时，学院的老师们也是都会耐心的为我们讲解，在这样一种环境下，每当我因为一些挫折而失落时，也都会迅速振作起来打起精神，积极的克服遇到的各种困难。

在耗时两三个月的毕业论文写作的过程之中，我深刻的感受到了知识的匮乏，也感受到了毕业论文的写作是非常不容易的，希望今后自己能够更加努力，继续学习计算机的相关理论知识。