**重庆大学课程项目报告**

项目名称： **简单的文件管理系统**

学 院： 计算机学院

专业班级： 计卓02

年 级： 2020

姓 名 1： 贾旺旺

学 号 1： 20204251

姓 名 2： 徐翎觉

学 号 2： 20204321

完成时间： 2022 年 12 月 4 日

指导教师： 李双庆

视频链接：https://www.bilibili.com/video/BV1JD4y1Y7tt/?vd\_source=ec93ae9e945374b83e05b6b6c93ba5ee

重庆大学教务处制

目录

**[重庆大学课程项目报告 1](#_Toc29246)**

**[一、 概述 2](#_Toc17743)**

**[二、 物理层 2](#_Toc9504)**

**[三、 数据链路层 3](#_Toc15439)**

**[四、 网络层 4](#_Toc19916)**

**[五、 运输层 6](#_Toc25449)**

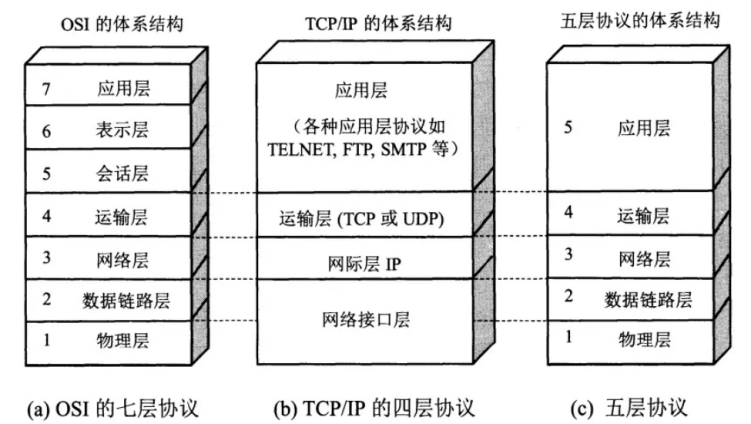
**[六、 应用层 8](#_Toc5836)**

**[七、 功能展示 11](#_Toc26787)**

**[八、 总结 14](#_Toc5387)**

# 概述

我们小组的计算机网络项目实现的是一个简单的文件管理系统，使用C语言代码编写，采用的是计算机网络的五层体系结构，如下图(c)所示。



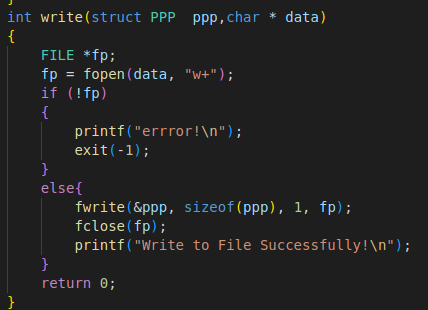
我们的文件管理系统可以实现文件的写入、读取、删除等功能，在各层之间主要采用结构体的形式进行数据传输。

下面是各层的实现功能及重要代码的讲解。

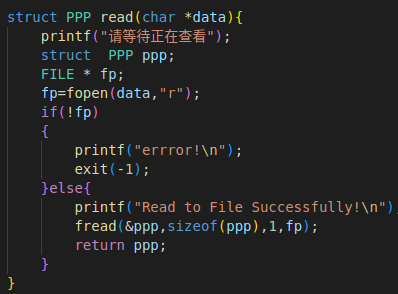
# 物理层

我们用文件的读写来模拟物理层比特流的传输。在封装过程中把数据链路层的数据以二进制形式写入一个文件中，在解封装过程中以二进制形式读取文件，将数据解析成数据链路层数据帧的格式。

这是写入文件的函数，把上层的数据以二进制形式写入给定的文件。

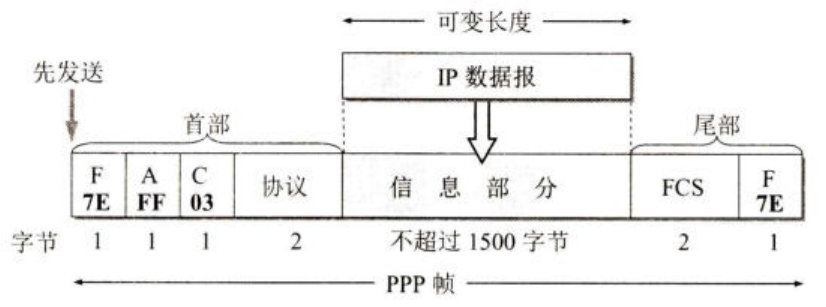


这是读取文件的函数，从给定的文件中读取二进制数据。



# 数据链路层

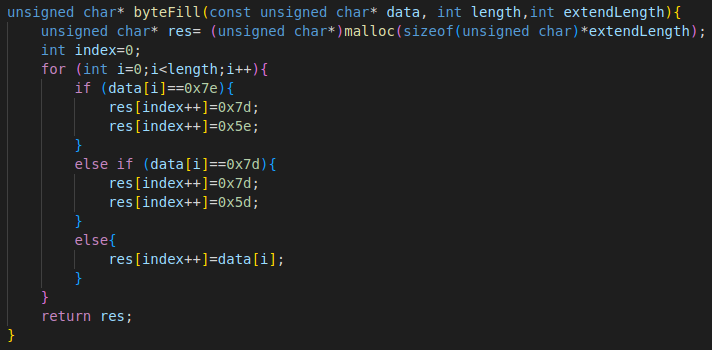
数据链路层我们采用的是PPP协议，数据帧的格式如下图所示。



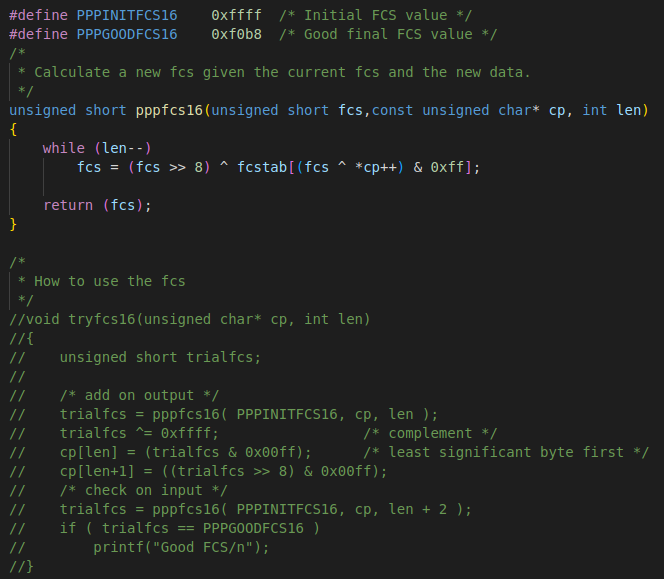
在代码中，我们把PPP数据帧封装成结构体。



在封装解封装过程中，一项比较重要的工作是字节填充和字节填充还原，下图展示的是字节填充函数。



计算FCS帧序列，我们采用的RFC1662文档附录提供的C语言实现，使用查表法，具体的代码和原理可以参考RFC1662文档。下图为文档里提供的函数（表略）。在封装和解封装过程中，我们采用了这个函数。

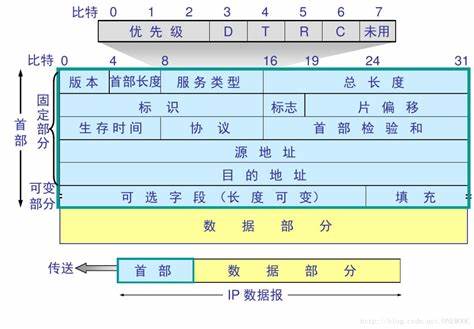


数据链路层对外的接口有打印、封装、解封装函数，下图为解封装函数（局部），首先进行帧校验，判断数据是否正确，再向上层传输。

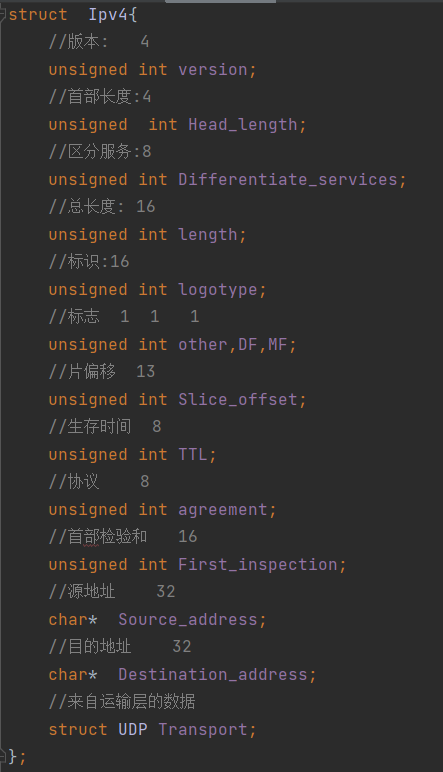


# 网络层

在网络层，我们主要采用的协议是IPV4协议。

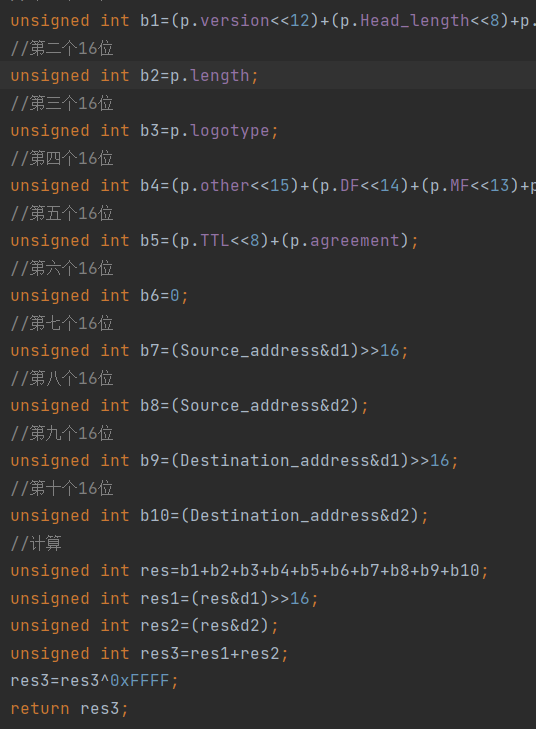


在代码中，我们把它封装为结构体，具体如下：

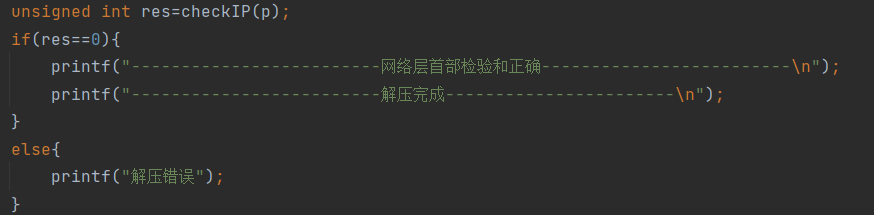


在网络层的封装解封装过程中，一项比较重要的工作是如何求首部检验和和如何验证首部检验和是否正确，下面我们将用两个来求取。

在求首部检验和中，我们调用的是getIP函数，我们将各个板块或者切分或者合并都凑成16位。然后按照数学公式去求解，具体就不再推导：



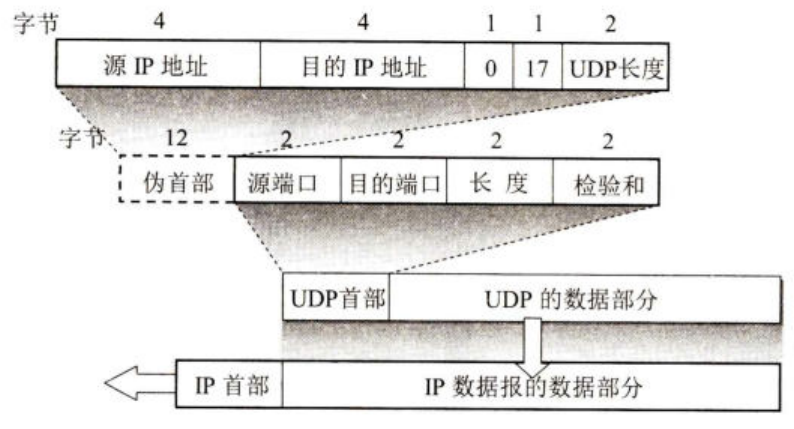
在检验首部检验和的过程中，我们调用的checkIP函数，我们将各个板块或者切分或者合并凑成16位，具体求解和上述过程相同，不过我们要把第六个16位变成getIP中求出的值。最终我们会得到res3==0，下面是我在解包时的判断条件。



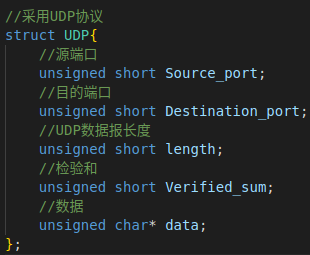
在此之外，我们还实现了封装（packIP）,解封装（unpackIP）,打印（printIP）三个函数。

# 运输层

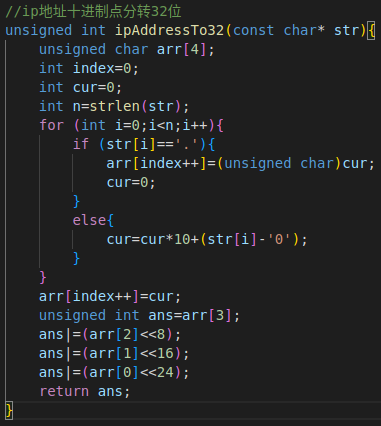
运输层我们采用的是UDP协议，UDP数据报格式如下图所示。



在代码中，我们把UDP数据报封装成结构体。

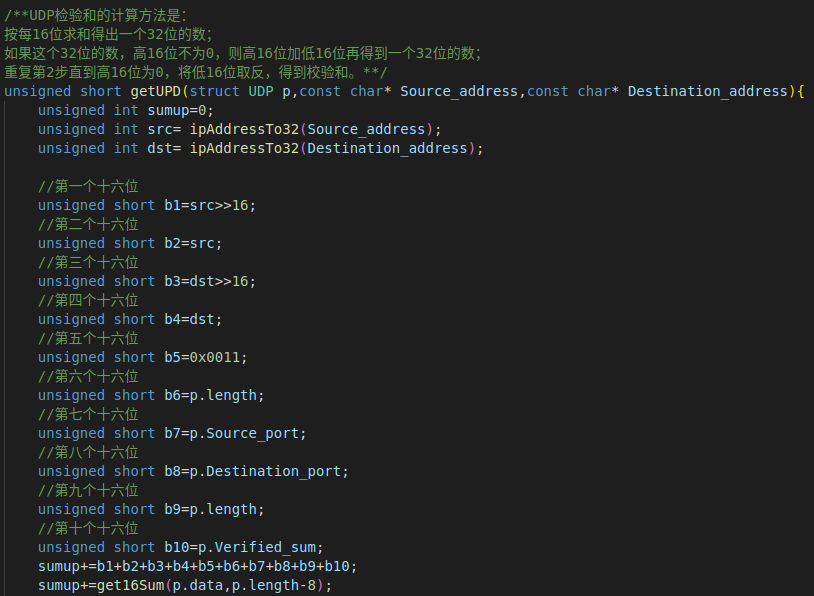


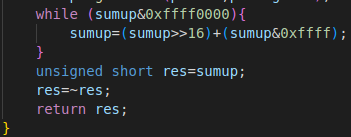
考虑到伪首部中含有IP地址，为方便用户输入，提供了IP地址十进制点分转32位的函数。



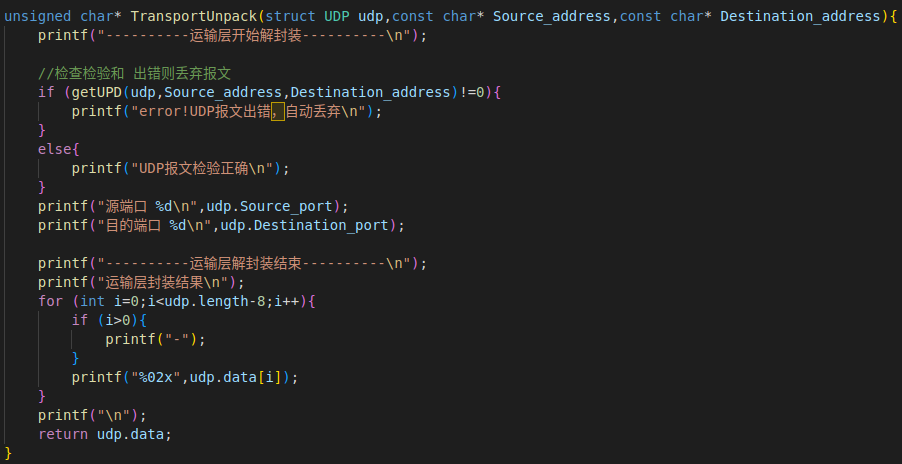
计算UDP检验和的函数如下图所示，方法参见函数上的注释。





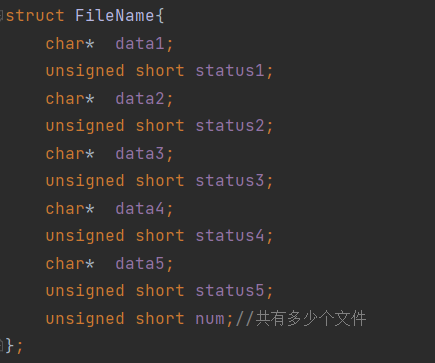


运输层的对外接口有打印UDP数据报、封装、解封装函数，下图为解封装函数，首先进行数据检验，输提示信息，然后将数据向上传递。

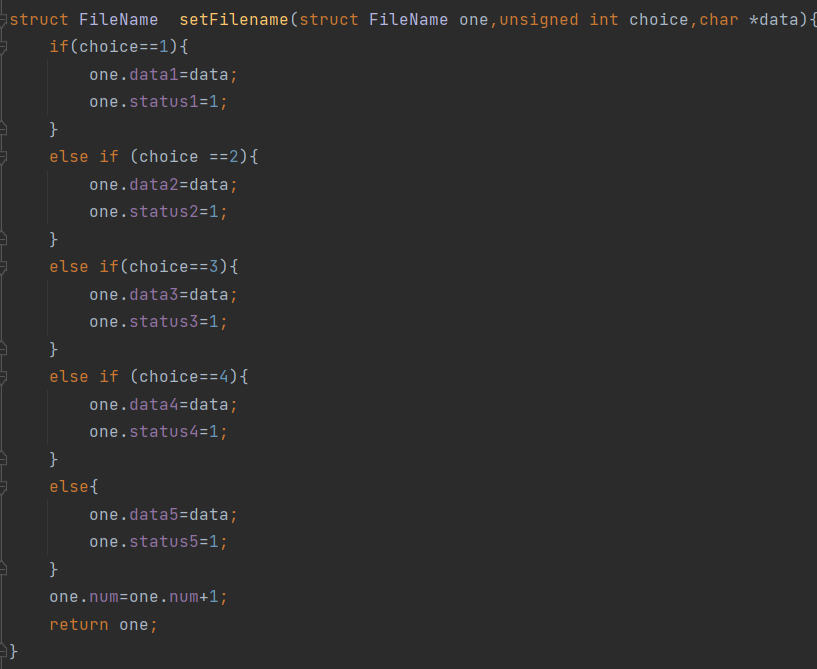


# 应用层

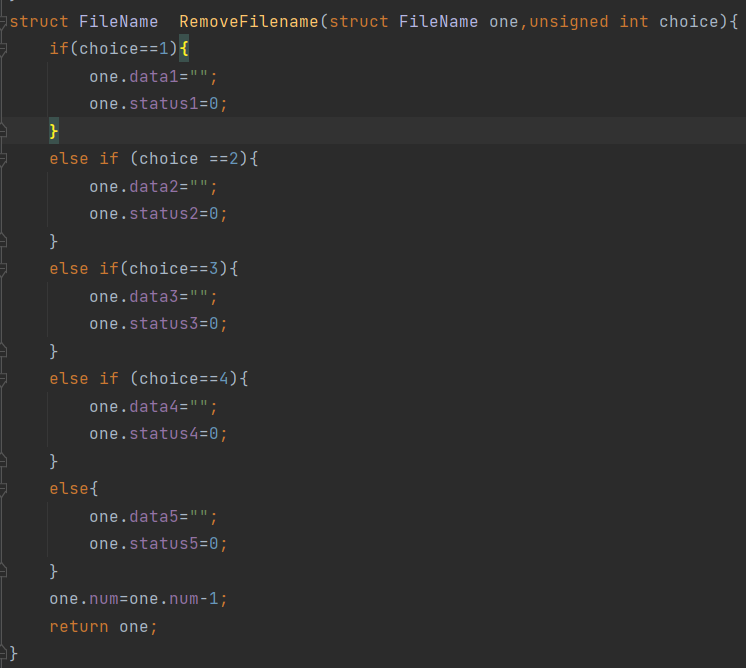
我们在应用层建立了一个文件管理系统。具体结构如下：



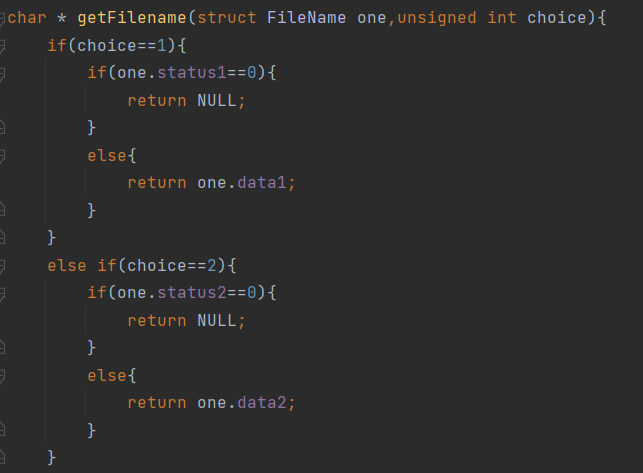
如上，我们目前的文件管理系统可以管理五个文件，我们以data1和status1为例，status1=0代表1号文件控制器尚未使用，此时data1无意义。当status1=1时代表1号文件控制器已经使用，此时data中存的就是1号文件控制器所控制的文件地址。我们可以通过此来对文件进行操作。注意一个文件管理器只能管理一个文件。



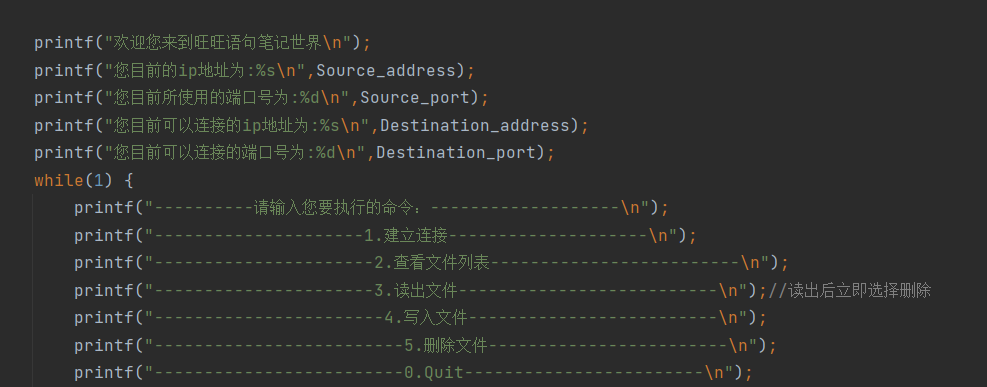
我们使用setFilename函数来在创建文件时将文件名存入对应的文件管理器和修改文件状态。



我们使用RemoveFilename函数来在删除文件时将文件名从文件管理器中删除，并将文件管理器的状态置为0；



我们使用getFilename函数来在查阅文件时返回文件路径（名字）。



最后如上是我们应用层实现的主要协议的具体实现吧。

应用层协议(application layer protocol)定义了运行在不同端系统上的应用程序进程如何相互传递报文。我们目前是模拟的在一台主机上用一个进程来模拟应用层的协议。实际上我们是想在一个操作系统上发布命令来操作另一个操作系统。所以也可以想象成一个简单的协议？

这里简单介绍一下我们的功能：

首先你必须建立连接，才可以控制对方系统的文件系统，如果没有建立连接则无法进行任何操作。

其次查看文件列表时会列出每一个文件控制器的状态。

再者读出文件只能读取已经建立的文件。

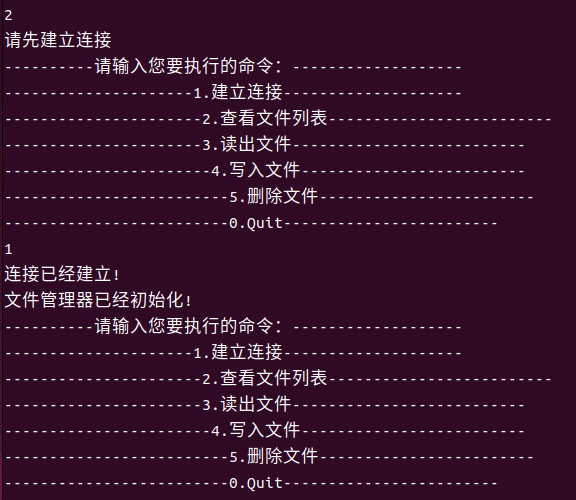
最后创建文件时必须申请文件管理器对其进行相应的管理，如果申请的文件管理器下面已经有存在的文件，则无法申请文件。

# 功能展示

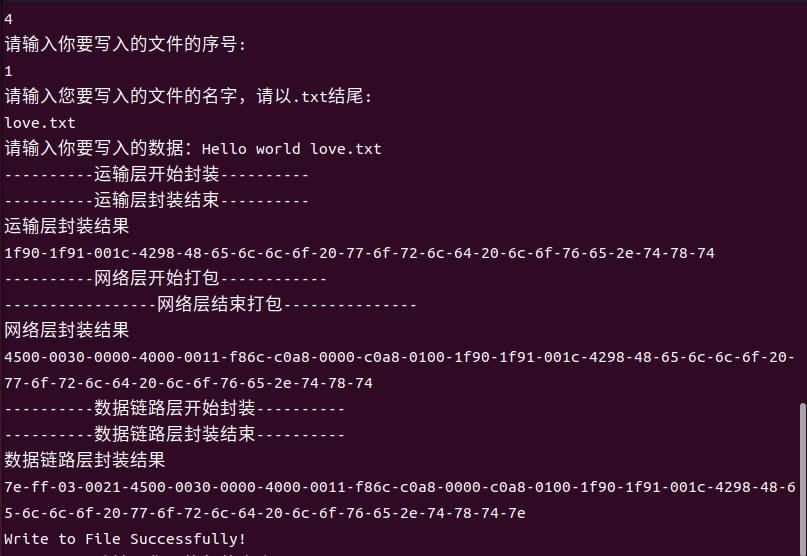
初始界面：



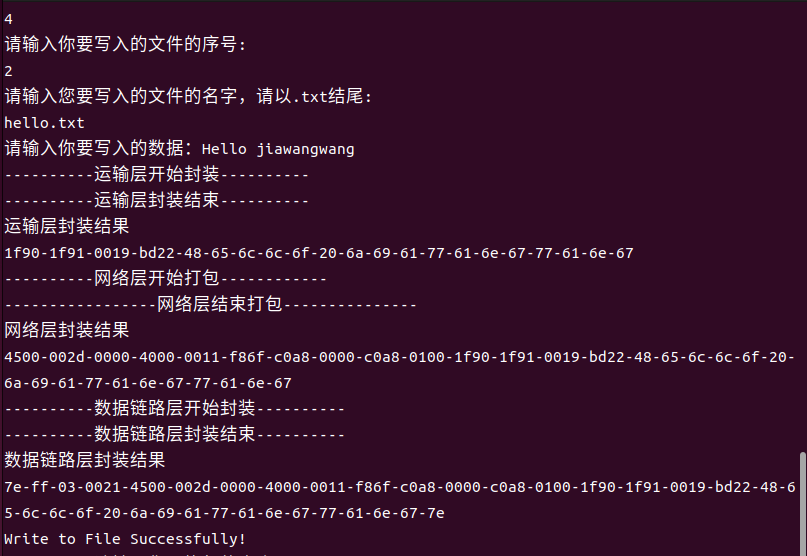
未建立连接和建立连接时的两种状态：



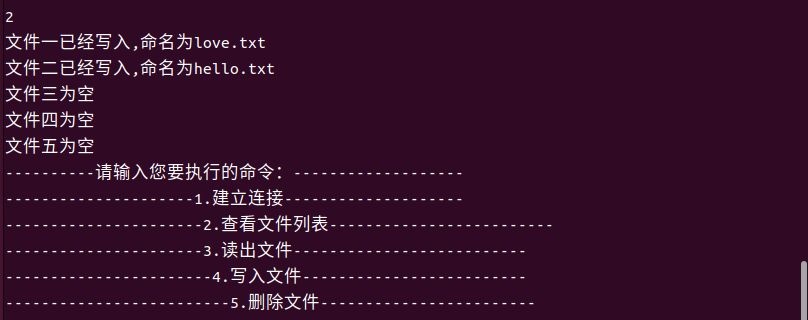
在1号控制器中创建love.txt，并在文件中写入“hello world love.txt”



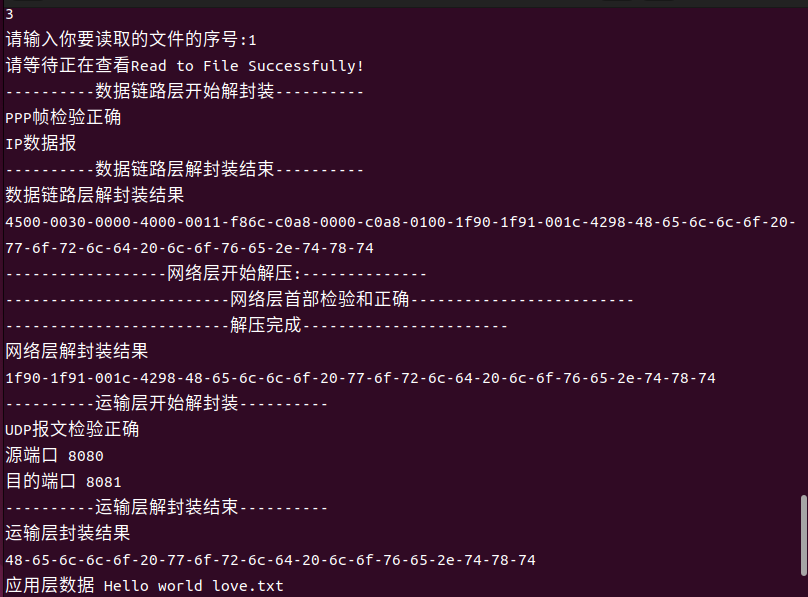
在2号控制器中创建hello.txt，并写入数据“Hello jiawangwang”。



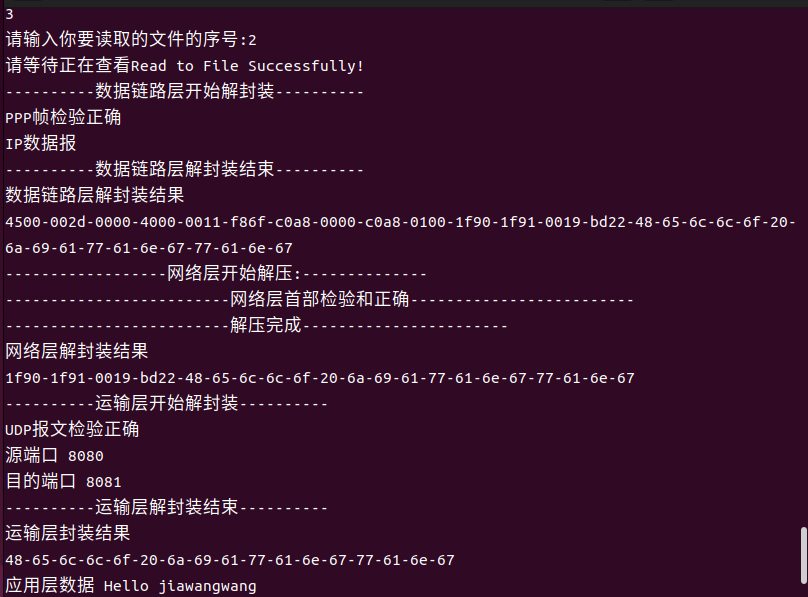
检查文件状态：



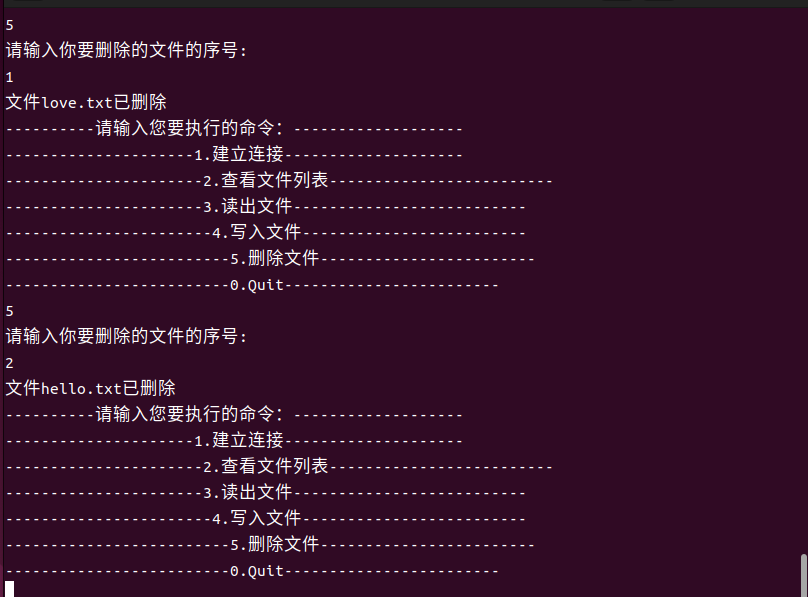
读取1号控制器下面的文件：



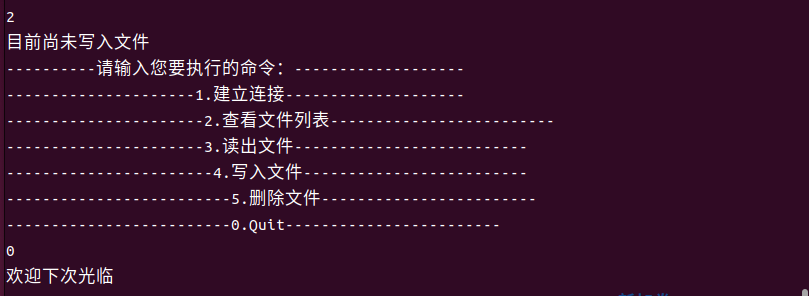
读取2号控制器下面的文件：



删除两个文件：



退出：



# 总结

通过本次项目的书写，我们对于计算机网络的五层架构有了较为深刻的理解。在不断地查阅资料中，我们一次又一次温习了计算机的五层架构，理解了数据如何在各个层次之间进行封装和解封装，在不断地学习过程中，我们的专业知识也掌握的更加牢靠，我觉得此次项目非常有意义。