

B 级达标测试 课程实验报告

实验名称 文件传输软件设计

2022 年 5 月 5 日

指导教师评语:

成 绩

测试教师:

____年____月____日

实验报告内容基本要求及参考格式

- 一、实验目的
- 二、实验原理
- 三、实验流程
- 四、实验数据记录（或仿真及软件设计）
- 五、实验结果分析及回答问题（或测试环境及测试结果）
- 六、心得与体会

一、实验目的

基于 http 协议,实现客户端与服务器文件互传软件系统,包括客户端不仅可以浏览自己本地的文件列表和服务器的文件列表,而且客户端可以将本地的文件上传到服务器及将从服务器上下载自己需要的文件.

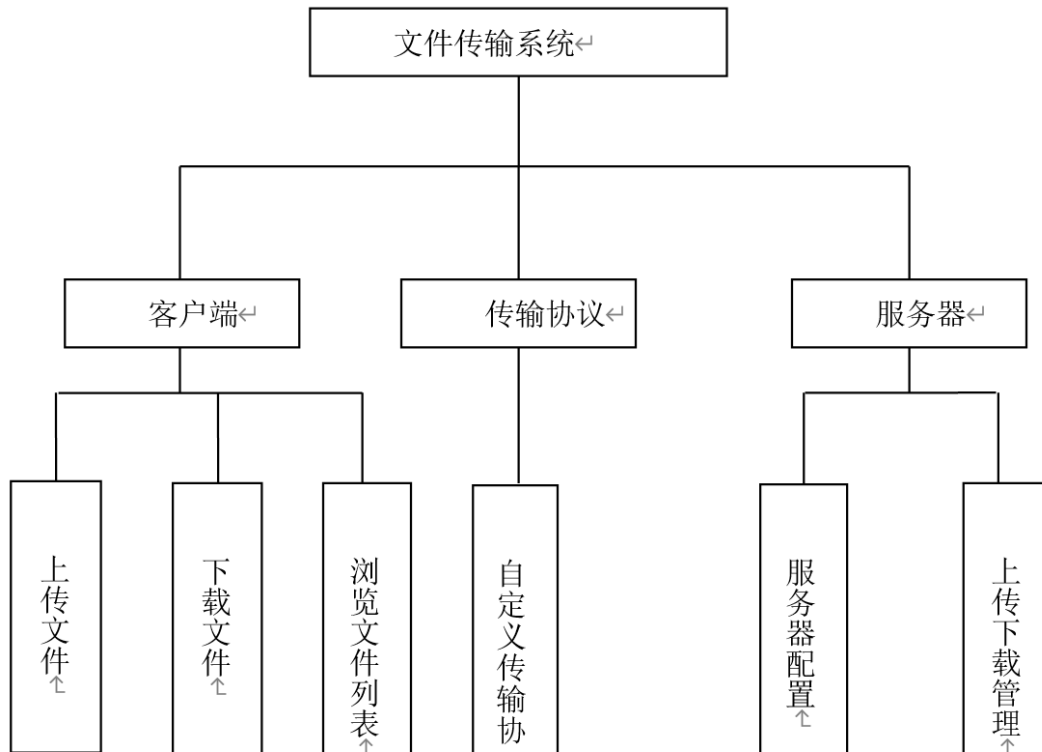


图 1 文件传输系统示意图

二、实验原理

HTTP 传输原理

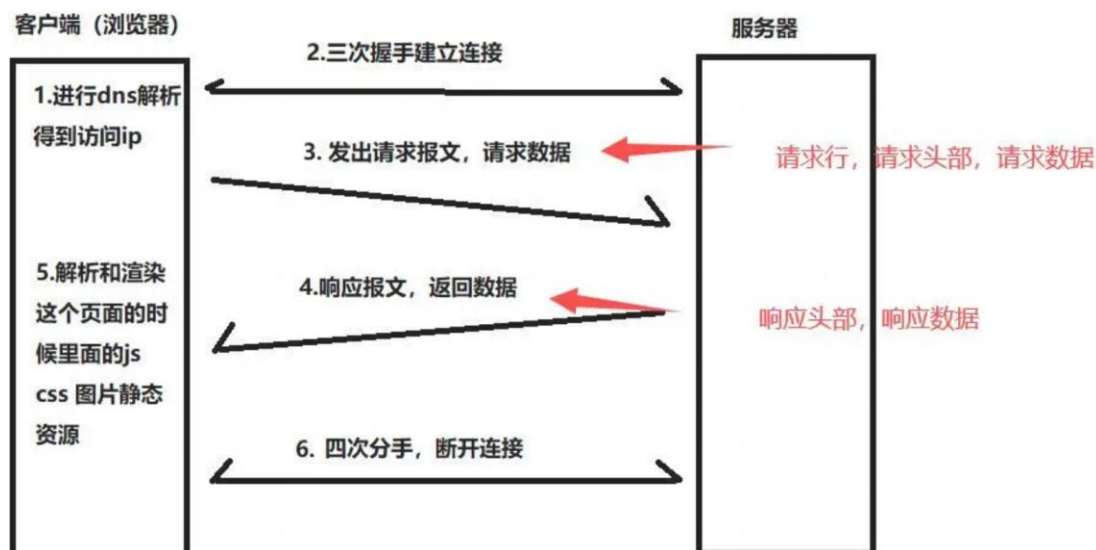
一. HTTP 介绍

HTTP 协议（HyperText Transfer Protocol，超文本传输协议）是 TCP/IP 协议的一个应用层协议，用于定义 WEB 浏览器与本地浏览器之间交换数据的过程。

它可以使浏览器更加高效，减少网络传输延时，具有正确性检验和文件压缩功能，还确定传输文档中的哪一部分，以及哪部分内容首先显示(如文本先于图形)等。

HTTP 由请求和响应构成，所有 HTTP 连接都被构造成一套请求和应答，是一个标准的客户端服务器模型（B/S）。

HTTP 遵循请求(Request)/应答(Response)模型,是一个无状态的协议. 客户机（Web 浏览器）和服务器之间不需要建立持久的连接。客户机（浏览器）向服务器发送请求，服务器处理请求并返回适当的应答,随后连接被关闭,并且服务端不保留连接相关信息。



HTTP 工作流程

二. 格式

1. 请求报文格式:

请求行 — 通用信息头 — 请求头 — 实体头 — 报文主体

请求行以方法字段开始，后面分别是 URL 字段和 HTTP 协议版本字段，并以 CRLF 结尾。SP 是分隔符。除了在最后的 CRLF 序列中 CF 和 LF 是必需的之外，其他都可以不要。有关通用信息头，请求头和实体头方面的具体内容可以参照相关文件。

1. 应答报文格式如下:

状态行 — 通用信息头 — 响应头 — 实体头 — 报文主体

状态码元由 3 位数字组成，表示请求是否被理解或被满足。原因分析是对原文的状态码作简短的描述，状态码用来支持自动操作，而原因分析用来供用户使用。客户机无需用来检查或显示语法。有关通用信息头，响应头和实体头方面的具体内容可以参照相关文件。

关于 TCP

- TCP 提供一种面向连接的、可靠的字节流服务
- 在一个 TCP 连接中，仅有两方进行彼此通信。广播和多播不能用于 TCP
- TCP 使用校验和，确认和重传机制来保证可靠传输
- TCP 给数据分节进行排序，并使用累积确认保证数据的顺序不变和非重复

- TCP 使用滑动窗口机制来实现流量控制，通过动态改变窗口的大小进行拥塞控制

TCP 并不能保证数据一定会被对方接收到，因为这是不可能的。它不是 100%可靠的协议，它所能提供的是数据的可靠传递或故障的可靠通知。

TCP 建立连接(三次握手)

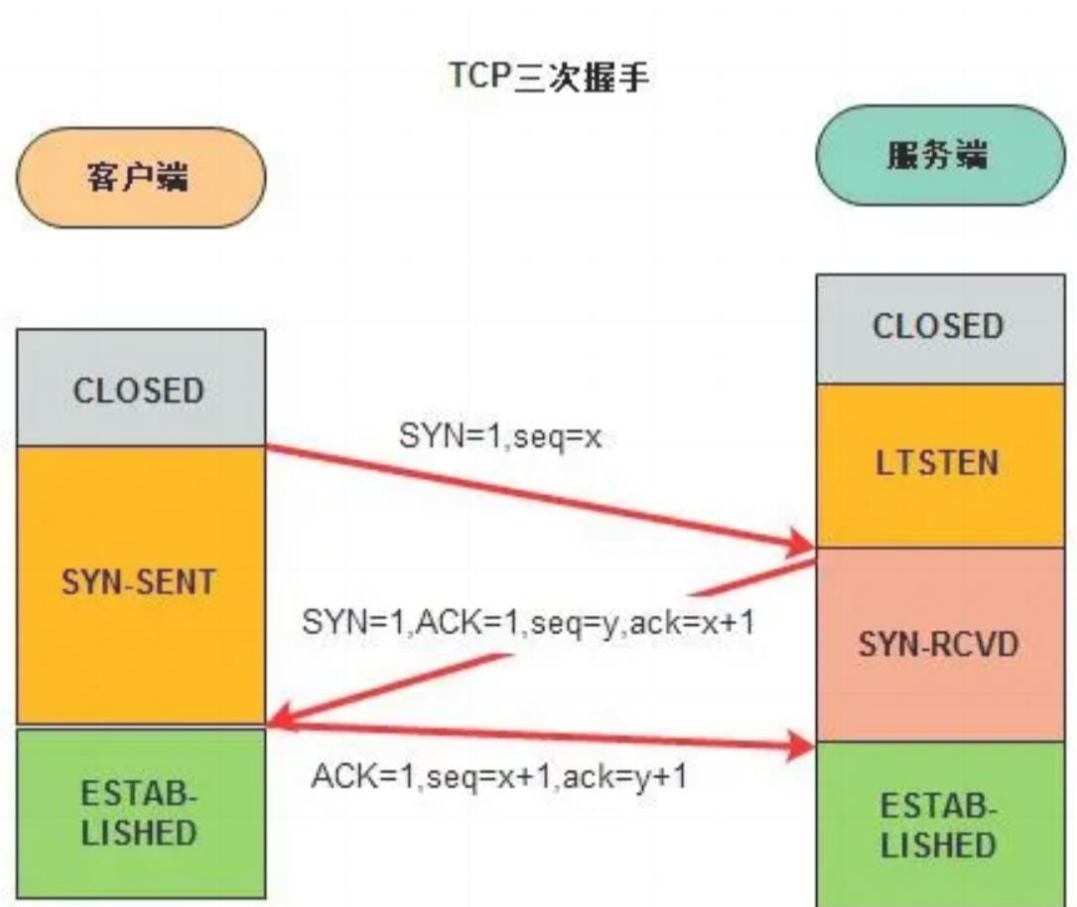
建立一个 TCP 连接时，需要客户端和服务端总共发送 3 个包。

目的是为了确保客户端和服务端发送的信息都能收到对方的应答。

A: 你好,能听到吗?(测试 A 发送)

B: 你也好,我说话你能听到吗?(B 应答, 测试 B 发送)

A: 能听到.(A 应答)



1. **第一次握手([SYN], Seq = x)** 客户端发送一个 SYN 标记的包，Seq 初始序列号 x，发送完成后客户端进入 SYN_SEND 状态。

2. **第二次握手**([SYN,ACK], Seq = y, ACK = x + 1) 服务器返回确认包(ACK)应答, 同时还要发送一个 SYN 包回去。ACK = x + 1, 表示确认收到(客户端发来的 Seq 值 + 1), Seq = y, 表示让客户端确认是否能收到。发送完成后服务端进入 SYN_RCVD 状态。
3. **第三次握手**([ACK], ACK = y + 1) 客户端再次发送确认包(ACK), ACK = y + 1, 表示确认收到服务器的包(服务端发来的 Seq 值 + 1)。客户端发送完毕后, 进入 ESTABLISHED 状态, 服务端接收到这个包, 也进入 ESTABLISHED 状态, TCP 握手结束。

TCP 关闭连接(四次挥手)

TCP 连接的断开需要发送四个包, 所以称为四次挥手。

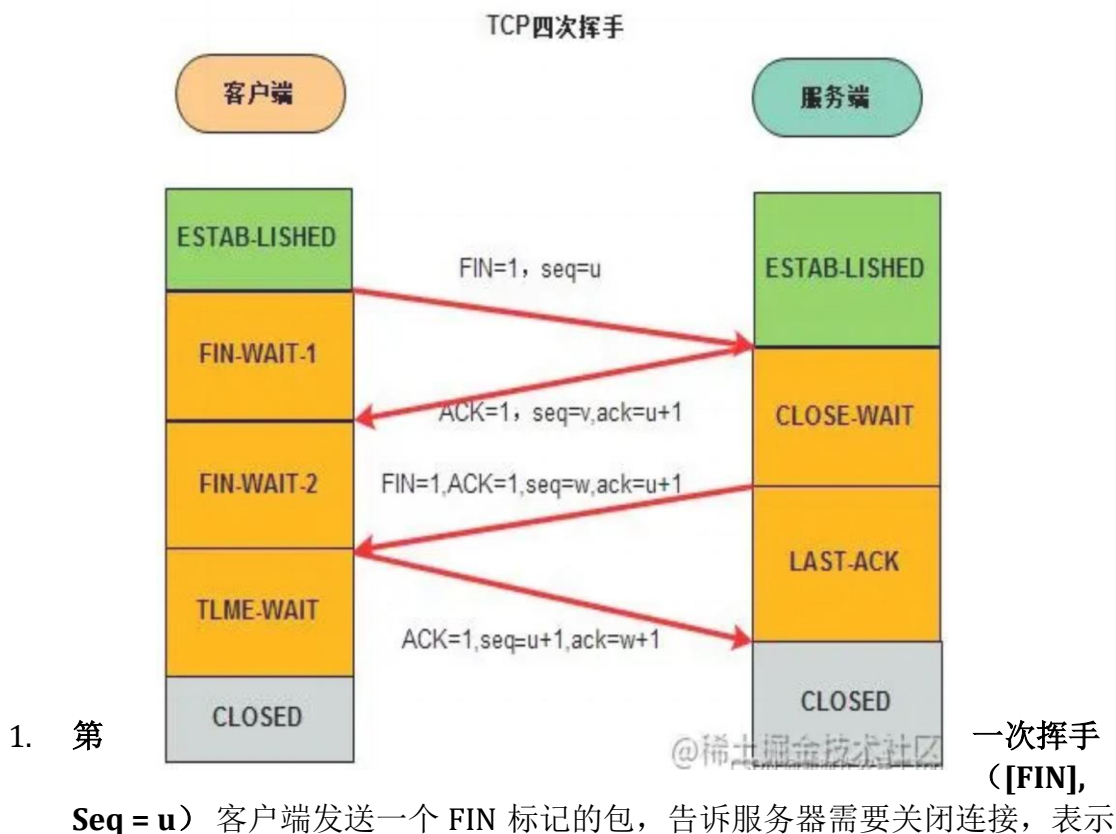
A: 我的发言到此结束, 你还有什么补充的吗?

B: 我核对一下....

B: 没有补充的了, 那我先下了?

A: 收到, 那下次再见.

(B 收到后下线, A 确认 B 下线后也下线.)



自己不用发送数据了，但是还可以接收数据。发送完成后，客户端进入 `FIN_WAIT_1` 状态。

2. **第二次挥手 ([ACK], ACK = u + 1, Seq = v)** 服务端发送一个 ACK 的确认包，告诉客户端接收到关闭的请求，但是还没有准备好。发送完成后，服务端进入 `CLOSE_WAIT` 状态，客户端收到这个包后，进入 `FIN_WAIT_2`，等待服务器关闭连接。
3. **第三次挥手 ([FIN], Seq = w, ACK = u + 1)** 服务端准备好关闭连接时，发送 FIN 标记的包，告诉客户端准备关闭了。发送完成后，服务端进入 `LAST_ACK` 状态，等待客户端确认。
4. **第四次挥手 ([ACK], ACK = w + 1)** 客户端接收到服务端的关闭请求，再发送 ACK 标记的确认包，进入 `TIME_WAIT` 状态，等待服务端可能请求重传的 ACK 包。服务端接收到 ACK 包后，关闭连接，进入 `CLOSED` 状态。客户端在等待固定时间(两个最大段生命周期)后，没有接收到服务的 ACK 包，认为服务器已关闭连接，自己也关闭连接，进入 `CLOSED` 状态。

哈希文件

哈希算法的不可逆特性使其在以下领域使用广泛, 在这里主要做文件传输正确性的判断

1. 密码，我们日常使用的各种电子密码本质上都是基于 hash 的，你不用担心支付宝的工作人员会把你的密码泄漏给第三方，因为你的登录密码是先经过 hash+各种复杂算法得出密文后 再存进支付宝的数据库里的
2. 文件完整性校验，通过对文件进行 hash，得出一段 hash 值，这样文件内容以后被修改了，hash 值就会变。MD5 Hash 算法的“数字指纹”特性，使它成为应用最广泛的一种文件完整性校验和(Checksum)算法，不少 Unix 系统有提供计算 md5 checksum 的命令。
3. 数字签名，数字签名技术是将摘要信息用发送者的私钥加密，与原文一起传送给接收者。接收者只有用发送者的公钥才能解密被加密的摘要信息，然后用 HASH 函数对收到的原文产生一个摘要信息，与解密的摘要信息对比。如果相同，则说明收到的信息是完整的，在传输过程中没有被修改，否则说明信息被修改过，因此数字签名能够验证信息的完整性。

//哈希

```
type Hasher struct {
    io.Writer
    io.Reader
    hash.Hash
    Size uint64
}
func (h *Hasher) Write(p []byte) (n int, err error) {
```

```

        n, err = h.Writer.Write(p)
        h.Hash.Write(p)
        h.Size += uint64(n)
        return
    }
    func (h *Hasher) Read(p []byte) (n int, err error) {
        n, err = h.Reader.Read(p)
        h.Hash.Write(p[:n])
        return
    }
    func (h *Hasher) Sum() string {
        return hex.EncodeToString(h.Hash.Sum(nil))
    }
}

```

三、实验结果

.\client.exe -help

```

PS C:\GoProjects\src\ftp\client> .\client.exe -help
Usage of C:\GoProjects\src\ftp\client\client.exe:
  -action string
        upload, download, list, history, or resume
  -downDir string
        下载路径, 默认当前目录 (default "download")
  -downfile string
        下载文件名
  -ip string
        服务器地址, 默认本机 (default "127.0.0.1")
  -port int
        服务器端口, 默认10808 (default 10808)
  -upfile string
        上传文件路径, 多个文件路径用空格相隔

```

#向服务器上传本地文件

```

.\client.exe --action upload -upfile "C:\Users\lenovo\Desktop\test1.txt"
"
.\client.exe --action upload -upfile "C:\Users\lenovo\Desktop\test2.txt"
"

```

```

PS C:\GoProjects\src\ftp\client> .\client.exe --action upload -upfile "C:\Users\lenovo\Desktop\test1.txt"
32d10c7b8cf96570ca04ce37f2a19d84240d3a89
上传文件test1.txt成功
PS C:\GoProjects\src\ftp\client> .\client.exe --action upload -upfile "C:\Users\lenovo\Desktop\test2.txt"
01b307acba4f54f55aafc33bb06bbbf6ca803e9a

```

查看服务器文件



.\client.exe --action list

```
PS C:\GoProjects\src\ftp\client> .\client.exe --action list
+-----+-----+
|  name  |  size  |
+-----+-----+
| test1.txt | 26 |
| test2.txt | 10 |
+-----+-----+
```

从服务器下载文件

```
.\client.exe --action download -downfile test1.txt
.\client.exe --action download -downfile test2.txt
```

indows (C:) > GoProjects > src > ftp > client > download

| <input type="checkbox"/> 名称 | 修改日期 | 类型 | 大小 |
|---|----------------|------|------|
|  test1.txt | 2022/5/9 18:24 | 文本文档 | 1 KB |
|  test2.txt | 2022/5/9 18:25 | 文本文档 | 1 KB |

下载历史

```
.\client.exe --action history
```

```
PS C:\GoProjects\src\ftp\client> .\client.exe --action history
+-----+-----+-----+
|  name  |  size  | status |
+-----+-----+-----+
| test1.txt | 26 | complete |
| test2.txt | 10 | complete |
+-----+-----+-----+
```

恢复下载

在服务器端设置大文件

```
PS C:\GoProjects\src\ftp\client> .\client.exe --action list
+-----+-----+
|  name  |  size  |
+-----+-----+
| bigguy.apk | 2070259761 |
| test1.txt | 26 |
| test2.txt | 10 |
+-----+-----+
```

image-20220509184734447

下载大文件

```
.\client.exe --action download -downfile bigguy.apk
```


断点续传

1. ctrl+c 或者 p + 回车

```
PS C:\GoProjects\src\ftp\client> .\client.exe --action download -downfile bigguy.apk
Downloading... 490 MB complete
Downloading... 602 MB complete
pause

P
start
Downloading... 715 MB complete
Downloading... 2.1 GB complete
```

image-20220509190345438




| | | | |
|---|----------------|--------------------|--------------|
|  bigguy.apk | 2022/5/9 19:24 | Android App Pac... | 2,021,739... |
|  bigguy.apk.2downloading | 2022/5/9 19:24 | 2DOWNLOADIN... | 1 KB |
|  bigguy.apk.downloading | 2022/5/9 19:04 | DOWNLOADING... | 1 KB |

image-20220509192514348

临时文件:.downloading 文件: json 版

 bigguy.apk.downloading 2022/5/9 19:04 DOWNLOADING... 1 KB

bigguy.apk downloading - 记事本

文件 编辑 查看

```
{ "name": "bigguy.apk", "size": 2070259761, "offset": 0, "status": true, "host": "127.0.0.1:10808" }
```

image-20220509192724505

.2downloading 文件: 存储 offset

n.go bigguy.apk.2downloading bigguy.apk.2downloading.hexdump x downloader.go

Offset: 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000: 00 00 70 35 00 00 00 00 ..p5....

image-20220509192938170

主动恢复下载全部文件

.\client.exe --action resume

```
PS C:\GoProjects\src\ftp\client> .\client.exe --action resume
```

```
PS C:\GoProjects\src\ftp\client> .\client.exe --action history
```

| name | size | status |
|------------|------------|----------|
| bigguy.apk | 2070259761 | complete |
| test1.txt | 26 | complete |
| test2.txt | 10 | complete |

image-20220509185210710

查看状态

.\client.exe --action history

```
PS C:\GoProjects\src\ftp\client> .\client.exe --action history
```

| name | size | status |
|------------|------------|-------------|
| test1.txt | 26 | complete |
| test2.txt | 10 | complete |
| bigguy.apk | 2070259761 | downloading |

image-20220509185146088

读取 client 端 download 目录下所有文件信息, 按照格式打印出文件信息, 状态为 complete. 对于未完成下载的文件, 忽略临时文件, 显示状态为 downloading.

四. 功能实现与代码解析

配置与辅助功能

- 定义路由与传输大小(config.go)

```
// 服务器地址
var BaseUrl string
var MaxPart int64 = 1024 * 1024
var MaxChan int = 1024
```

- 文件结构 (file.go)

```

type FileInfo struct {
    Filename string `json:"name"`
    Filesize int64 `json:"size"`
    Offset    int64 `json:"offset"`
    Status    bool  `json:"status"`
    Host      string `json:"host"`
}
// 文件列表
type ListFileInfos struct {
    Files []FileInfo
}

```

- 下载进度输出(common.go)

```

type Reader struct {
    io.Reader
    Name      string
    Total     uint64
    Current   uint64
}
func (r *Reader) Read(p []byte) (n int, err error) {
    n, err = r.Reader.Read(p)
    r.Current += uint64(n)
    fmt.Printf("\r%s\t 进度 %.2f%%", r.Name, float64(r.Current*10000/r.Total)/100)
    return
}

```

- 日志输出

```

// 日志输出到info.log
file, err := os.OpenFile("info.log", os.O_CREATE|os.O_APPEND|os.O_WRONLY, 0644)
if err != nil {
    log.Fatal("cant open log: ", err)
}
defer file.Close()
log.SetOutput(file)
log.SetPrefix("server main(): ")

```

- 服务器端端口定义(config.json)

```

// ServiceConfig 配置文件结构
type ServiceConfig struct {
    Port      int
    Address   string
    StoreDir  string
}
// ./etc/config.json
{

```

```

        "Port": 10808,
        "Address": "0.0.0.0", //全网
        "StoreDir": "upload" //存储文件名
    }

    //-----server main.go-----
    var configPath = flag.String("configPath", "./etc/config.json", "
    服务配置文件")
    var confs = &common.ServiceConfig{}
    // 加载配置文件
    func loadConfig(configPath string) {
        if !common.IsFile(configPath) {
            log.Panicf("config file %s is not exist", configPath)
        }
        buf, err := ioutil.ReadFile(configPath)
        if err != nil {
            log.Panicf("load config conf %s failed, err: %s\n", c
            onfigPath, err)
        }
        err = json.Unmarshal(buf, confs) //配置读入 conf 中
        if err != nil {
            log.Panicf("decode config file %s failed, err: %s\n",
            configPath, err)
        }
    }
    // -----main 函数中主动加载服务器配置-----
    loadConfig(*configPath)

```

上传与下载(以上传为例)

一. 客户端(client)

1. 根据路径判断文件是否存在

```

targetUrl := common.BaseUrl + "upload"
if !common.IsFile(filePath) {
    fmt.Printf("filePath:%s is not exist", filePath)
    return errors.New(filePath + "文件不存在")
}

```

2. 将打开的文件写入 multipart

```

filename := filepath.Base(filePath)
bodyBuf := &bytes.Buffer{} //http body
bodyWriter := multipart.NewWriter(bodyBuf)
fileWriter, err := bodyWriter.CreateFormFile("filename", filename)
if err != nil {
    fmt.Println("error writing to buffer")
    return err
}

```

//打开文件句柄操(文件句柄对于打开的文件是唯一的识别依据)

```

fh, err := os.Open(filePath)
if err != nil {
    fmt.Printf("error opening filePath: %s\n", filePath)
    return err
}
hasher := &common.Hasher{//指针
    Reader: fh,    //文件名
    Hash:  sha1.New(),
    Size:  0,
}
_, err = io.Copy(fileWriter, hasher)//写http body
if err != nil {
    return err
}
contentType := bodyWriter.FormDataContentType()
bodyWriter.Close()

```

3. 创建 http 请求

```

request, err := http.NewRequest(http.MethodPost, targetUrl, bodyBuf)
if err != nil {
    return err
}
request.Header.Set("Content-Type", contentType)
request.Header.Add("file-md5", hasher.Sum())
fmt.Println(hasher.Sum())
resp, err := http.DefaultClient.Do(request) // enter 键
if err != nil {
    return err
}
defer resp.Body.Close()
if resp.StatusCode != http.StatusOK {
    fmt.Printf("%s 文件上传失败\n", filename)
    return errors.New("上传文件失败")
}
fmt.Printf("上传文件%s 成功\n", filename)

```

二. 服务器(server)

一. 打开 http 文件体

```

file, handler, err := r.FormFile("filename")
if err != nil {
    fmt.Println(err)
    return
}
defer file.Close()

```

二. 下载到设置文件中规定的目录

```

hasher := &common.Hasher{
    Reader: file,

```

```

        Hash:  sha1.New(),
        Size:  0,
    }
    //存储到本地
    f, err := os.OpenFile(path.Join(confs.StoreDir, handler.Filename),
os.O_WRONLY|os.O_CREATE, 0666)
    if err != nil {
        fmt.Println(err)
        return
    }
    defer f.Close()
    io.Copy(f, hasher)

```

三. 传输校验

```

    if r.Header.Get("file-md5") != "" && r.Header.Get("file-md5") !=
hasher.Sum() {
        http.Error(w, "md5 错误", http.StatusBadRequest)
        return
    }

```

文件列表浏览

客户端向服务器发送 http 请求, 服务器获取请求后, 读取并生成文件信息, 以 json 形式返回给客户端.

```

//路由设置
http.HandleFunc("/list", listFiles)
http.HandleFunc("/upload", upload)
http.HandleFunc("/download", download)
http.HandleFunc("/info", info)

```

客户端收到信息后, 将 json 解码为结构体. 并以表格的形式打印出来

```
// 文件元数据
type FileInfo struct {
    Filename string `json:"name"`
    Filesize int64 `json:"size"`
    Offset    int64 `json:"offset"`
    Status    bool  `json:"status"`
    Host      string `json:"host"`
}

// 文件列表
type ListFileInfos struct {
    Files []FileInfo
}
```

五、心得与体会

这次实验,增进了我们小组协作开发的能力.在开发文件传输系统的过程中,增进了我们对 go 语言编程和 IO 模块的理解,加深了我们对 HTTP 和 TCP 协议的了解.在开发过程中,我们遇到了很多困难,最终通过查阅资料解决.并且我们原实验要求的基础上,增加断点续传功能.在协作过程中,我们切身体会了写代码不写注释的人的可恶,明白了写文档的重要性.

程序还有很多待改进的地方,比如,日志输出不够规范,并且数据库模块也没有开发完毕,多线程下载部分开发也遇到困难.....我们之后会继续完善.