一、数据结构的定义

1. 类型

FTP协议有控制和数据两个连接，因此在发送报文时也区分控制报文&数据报文。Msg\_Data\_Type为报文类型，包括Control和Data两种类型

enum Msg\_Data\_Type{

    Control,

    Data

};

此外，还需要为不同的指令设置不同的指令类型Syntax\_Cmd。除了8种指令外，还另外设置了一个FTP\_error用来表示报文是否出错。

typedef enum Syntax\_Cmd{

    FTP\_get,

    FTP\_put,

    FTP\_delete,

    FTP\_ls,

    FTP\_cd,

    FTP\_mkdir,

    FTP\_pwd,

    FTP\_quit,

    FTP\_error

}Syntax\_Cmd;

2. 消息类型

首先需要包含上述定义的两个类型，即消息类型和指令类型。

此外，需要有储存数据的data段，用于在client和server之间的数据传输；对应的需要data\_size字段确定储存数据的长度。

由于data字段的长度有限，而在传输文件（get/put）内容时，文件内容可能大于data段的最大长度，因此需要用多个报文段传输数据。为了表示文件内容全部读取完成，需要一个last字段。

最后，为了表示在报文传输，或server段执行时可能出现的错误，增加一个error字段表示错误。

#pragma pack(1) //1字节对齐

typedef struct MsgHeader

{

    Syntax\_Cmd s\_cmd;

    enum Msg\_Data\_Type MsgType;

    int data\_size;

    char data[CACHE\_SIZE];

    bool last;

    bool error;

}MsgHeader;

#pragma pack()

特别的是，由于struct在设置时会自动进行对齐，为了节省内存空间，减小传输报文的长度，用#pragma pack()规定报文字段1字节对齐。

二、控制连接

客户端输入的所有指令通过控制信息传递给服务器，将此过程封装在函数Control\_Connection中。

1. 客户端发送控制信息

首先确定控制信息ControlMsg各个数据项的意义：

|  |  |
| --- | --- |
| Struct MsgHeader | ControlMsg |
| MsgType | 控制类型的信息——Control |
| Cmdtype | 根据输入的指令确定各自的指令类型 |
| Data\_size |  |
| data | <get put delete cd mkdir> filepath  <ls pwd quit>空 |
| last | 控制连接只需要一个Msg，因此client可以直接将last设置为1 | |
| Error | 用于反馈，此处为0 |

初始化各个数据项后，通过send发送给服务器。

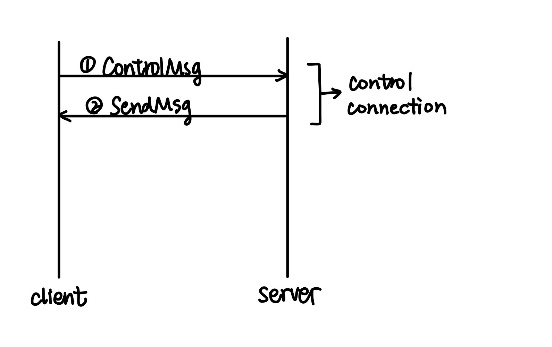
1. 服务器端接收控制信息

通过recv将收到的信息保存在ControlMsg中

通过ControlMsg->Cmdtype导向不同的处理函数

对于不需要数据传输的指令（delete cd mkdir quit），在反馈信息中向客户端发送指令执行的情况：如果error=false，则成功执行；如果error=true，则在data中包含错误信息

而对于需要进行数据传输的指令，则服务器端先发送反馈信息表示已经成功收到控制信息，再进行后续的数据传输。

三、不需要数据传输的指令的执行过程

在收到ControlMsg后执行对应指令，并以SendMsg为反馈信息发送给client，同时由于这类指令不需要数据传输，因此将执行情况捎带在控制信息的ACK/NAK中

如果成功执行，则将SendMsg的error设置为false；如果执行失败，则把错误提示信息储存在SendMsg的data中，并相应修改data\_size

对于这类指令，client的操作相同。即发送ControlMsg，等待recv\_msg，如果error为0则输出Done，否则输出data中携带的错误信息。将整个过程封装在Control\_Connection中，方便需要数据传输的指令进行控制传输。

【delete】

【cd】

【mkdir】

<server>

首先需要确定设立新文件夹的地址。在server端curpath储存当前路径，而ControlMsg中的data储存新文件夹名，即新文件夹路径应为curpath+”/”+ControlMsg.data。通过strncpy将此路径储存在block.filepath中。

然后通过调用\_access()函数判断在该路径下是否存在同名文件

如果不存在即返回值为-1，则调用mkdir()新建文件夹。如果mkdir返回值为-1，说明新建文件夹失败，则需要在SendMsg中向client发送信息表示新建失败，包括error=true以及在data中添加说明”fail to make a new directory”；如果返回值非-1，说明新建成功。

如果已经存在文件夹，则修改SendMsg信息表示“the directory exists“

最后调用send向client发送控制信息即可。

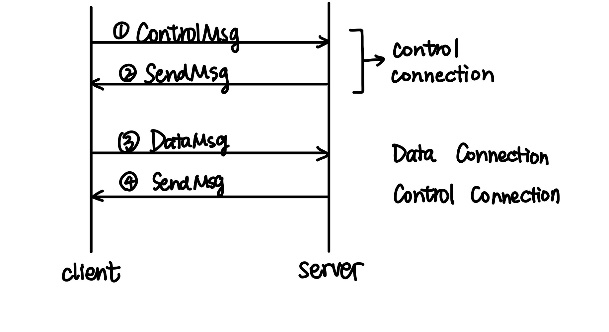
【quit】

四、需要数据传输的指令

【get】

【put】

<client>

 首先控制连接将文件名传给server，server通过curpath和ControlMsg.data得到文件路径。

然后client通过数据链路向server发送数据。通过循环重复执行“读取数据-封装成报文-发送-等待回复“，其中以256byte为单位划分报文，并且每次需要确定已经从client接收到ACK（即error==0）时才能继续下次读取数据，否则重复发送当前报文，直到server正确收到并正确反馈ACK。

对于文件不存在的情况，在进行控制连接之前，先判断是否存在该文件。利用file\_type函数查找文件，如果返回值为NOT\_FOUND，则说明文件不存在，此时不再进行后续的控制&数据报文传输，直接在client端输出”No such file”后进入下一次的指令循环。

<server>

从控制链路获得的ControlMsg中得到文件名后，构造文件路径到block.filepath

然后循环执行“数据链路接收报文-存入文件-返回信息“，如果接收到的报文error==1，则发送NAK表示收到的报文有错，希望client重新发送报文，直到收到正确报文。

【ls】