FTP-stage1文档

1.服务器

基于 `C` 语言的编程习惯，对于server中的多次重复操作都封装成了“宏”和“宏函数”，方便开发者调用。

a.日志记录

在实际工程运行中，任何程序难免会出现各种问题，需要输出这些错误信息来帮助修正程序。但根据要求，server程序不能直接在控制台上输出错误信息，因此建立了日志系统，其功能实现在 `log.h` 和 `log.c` 中。
程序会在 `/tmp` 文件夹下创建一个 `log.txt` 文件，将相关写操作封装成名为 `LOG(msg)(msg为一个字符串)` 的宏，当程序中的某处出现异常时，异常信息会以 `发生异常处的函数名:异常信息` 的格式输出到 `log.txt` 中。

b.服务器回复消息

由于 `rfc` 规定服务器的回复必须是一行一行的，且每一行的基本格式是"code + information",因此将每一行回复封装成一个结构体如下:

```C
#define MAX_WORDS_PER_LINE 128
struct FTP_Response{
    int code;
    char msg[MAX_WORDS_PER_LINE];
    struct FTP_Response *next;
};
```

server的全部回复就形成一个 `FTP_Response` 的链表，每次返回消息时就根据链表信息输出。服务器发送消息封装成名为 `REPLY(msg)(msg是FTP_Response链表的头节点)` 的宏。

c.服务器接收消息

`rfc` 规定客户端以 `verb paramaters \r\n` 的格式发送请求，据此定义结构体如下:

```C
#define MAX_REQUEST_WORDS 128
struct FTP_Request{
    char verb[MAX_REQUEST_WORDS];
    char param[MAX_REQUEST_WORDS];
};
```

当服务器接收到消息时，将消息经过解析返回上述结构体供后续程序使用。

d.程序架构

使用了 `select + 多线程` 的方式实现了多客户端访问服务器。主逻辑是单线程， `select` 返回时，不同的程序描述符变的可读，就执行相应的功能函数。当接受和发送文件时，创建新线程，在新的线程中使用 `recv` 和 `send` 函数，当 `recv` 和 `send` 完成后就关闭该线程，这样当 `send` 和 `recv` 发生阻塞时就不会对主逻辑造成影响。
对于不同请求命令，调用对应的 `handler` 函数来处理相应请求。

e.程序的ip参数

在 `PASV` 模式下，server需要发送本机的ip地址，因此server需要知道本机的ip地址。由于部署到不同的服务器上的ip地址是不同的，因此需要手动输入其ip地址，给 `./server` 程序新增了一个 `-ip xx.xx.xx.xx` 参数，表示server的实际ip地址，如果未给该参数，则ip地址默认为 `local host = 127.0.0.1`

f.FTP指令

实现了 `USER anonymous` (并为登录方式预留了接口), `PASS` , `PASV` , `PORT` , `LIST` (根据 `rfc` 规定实现了两种格式), `RNFR` , `RNTO` , `RETR` , `STOR` , `CWD` , `PWD` , `MKD` , `RMD` , `QUIT` , `ABOR` , `SYST` , `TYPE` 。

在实现文件路径指令时，服务器存储了每个客户端的根目录路径以及当前相对于根目录的路径，文件的实际路径就是上述两个路径的拼接。在实现 `CWD` 路径时，将 `CWD` 后的路径，按照 `/` 拆分成一个个小段，从当前路径开始，依次验证当前小段是否合法，从而实现了较复杂的 `CWD` 命令。(例如：根目录是 `/tmp` ，当前目录是 `/test/` ，现有 `CWD ../test2` 命令， `../test2` 拆分成 `..` 和 `test2` ，依次分析每个小段，先分析 `CWD ..` 回到上层文件夹，此时当前目录变为 `/` ，然后分析 `CWD test2` ，判断当前目录下是否有test2文件夹，返回结果)

2.客户端

使用了 `C++` 进行编写，使用了 `select` 来处理程序，可以在 `Mac OS` 和 `Linux` 系统上运行。客户端默认以 `client` 程序所在的文件夹为系统路径，所有下载的文件都会存储到 `client` 程序所在的文件夹。