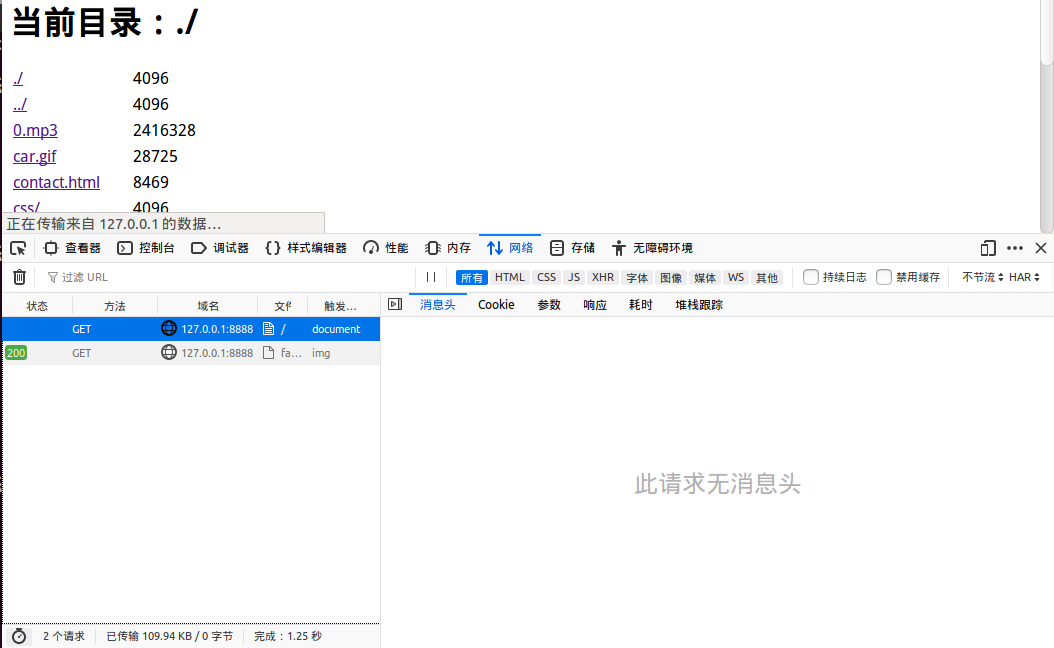
相比于socket14(静态服务器)中的v0.6版本，终版静态服务器WebServer01做了以下修改：

为加深印象，并且也为优化代码结构 提升代码质量

问题：

1. 一直都存在的问题是客户端存在一直在请求无法析构请求的情况



通过分析确定是处于一直请求的状态，因为当favion出现的时候，还没有出现正确的状态码；

解决问题：请求结束析构问题;（尝试当加入Nginx是否会避免发生这样的情况）

历经几个月的折磨，终于找到了问题的关键，由于我我们发送的是目录而不是文件，因此没有文件结束标志；但是偶然之间通过分析发现sendError发送错误页面时，发送完毕后；直接return ANALYSIS\_ERROR;因为我也可以在这个目录发送完后return ANALYSIS\_ERROR;但是将其改一个名字，这样使得我的程序可读性较高，这个名字叫return ANALYSIS\_DIR\_SUCCESS；

与先前版本不同的地方：（升级改造的地方）

仿照陈硕的书的muduo网络库，写了4个缓冲区的异步Log日志，没有区分日志的优先级，其他基本都具备了

该版本在测试过程中出现段错误

解决方式：

在启动myserver服务器时，应该使用root权限进行启动，这样才不会出现这样的问题

1. 准备加入访问默认目录：

该功能的目的是当我们在访问路径中未输入确定的访问文件时，直接默认访问所设置的文件目录中看所有文件；这里就会涉及到目录的展示，传输一个由目录组合好的网页信息；

首先解决的问题;

由于我们点击文件的时候，会把文件名字连接起来，放入地址栏中，因此这里就存在着中文字符的解码和编码的问题

/\*不管是否为中文文件，都对其进行解码，虽然解码，但是其实如果是英文字符我们不会对其进行处理\*/

char\* tmp = (char\*)fileName\_.*c\_str*();

decode\_str(tmp, tmp);

fileName\_ = (*string*)tmp;

其次我们就是要解决怎么访问和显示文件夹 或者目录的问题

if (S\_ISDIR(sbuf.*st\_mode*))

{

//判断如果是目录，需要将目录中的信息拼接成一个网页发给客户端

/\*由于重写发送机制因此我们只需要将信息放入outBuffer\_即可，不需要重复发送--接下来---拼接一个html压面发给客户端\*/

char buf[4094] = { 0 };

int i;

*sprintf*(buf, "<html><head><title>目录名：%s</title></head>", fileName\_.*c\_str*());

*sprintf*(buf + *strlen*(buf), "<body><h1>当前目录：%s</h1><table>", fileName\_.*c\_str*());

char enstr[1024] = { 0 };

char path[1024] = { 0 };

//目录项二级指针

struct dirent\*\* ptr;

int num = scandir(fileName\_.*c\_str*(), &ptr, *NULL*, alphasort);

//遍历

for (i = 0; i < num; ++i)

{

char\* name = ptr[i]->d\_name;

//拼接文件的完整路径

*sprintf*(path, "%s/%s", fileName\_.*c\_str*(), name);

//打印太多了因此把这段代码进行注释

//printf("path = %s ==================\n", path);

struct *stat* st;

*stat*(path, &st);

//编码生成 %E5 %A7 将中文字字符进行编码 但是在进行路径文件分析时需要解码 编码和解码相对应

//void encode\_str(char \*to, int tosize, const char\* from)

encode\_str(enstr, sizeof(enstr), name);//将地址写入到内存 并且进行编码

/\*

memcpy(enstr, name, strlen(name));

//a使用字串形式打印

enstr[strlen(name)] = '\0';\*/

//如果是文件

if (S\_ISREG(st.*st\_mode*))

{

*sprintf*(buf + *strlen*(buf), "<tr><td><a href=\"%s\">%s</a></td><td>%ld</td></tr>", enstr, name, (long)st.*st\_size*);

}

else if (S\_ISDIR(st.*st\_mode*))

{

*sprintf*(buf + *strlen*(buf), "<tr><td><a href=\"%s/\">%s/</a></td><td>%ld</td></tr>", enstr, name, (long)st.*st\_size*);

}

}

*sprintf*(buf + *strlen*(buf), "</table></body></html>");

//将buf转换成string 放入outBuffer\_中

outBuffer\_ += (*string*)buf;

//这里我想打印当前请求目录的时间，方便随时查看是否服务器还在运行

//获取系统时间

*time\_t* now\_time = *time*(*NULL*);

//获取本地时间

*tm*\* t\_tm = *localtime*(&now\_time);

*printf*("Request Dir: %s\n", *asctime*(t\_tm));

return ANALYSIS\_SUCCESS;

}

1. 切换工作路径：

当进入程序中，首先切换工作目录到我们制定的系统目录中，在指定目录下去访问相关资源，方便了程序对当前路径下，当前文件的读取。

这里存在一个问题：由于我们在此处设置了logfile日志文件，因此当我们切换时，这时如果日志目录没有切换的话就会出现段核心错误，正确的操作是将dir目录放在当前目录下，将log文件夹建立在dir目录下

函数：int ret = chdir(PATH);

1. 加入Nginx

Nginx是高性能的http和反向代理服务器；Nginx的作用包括反向代理；负载均衡和动静结合；这里我们主要使用其反向代理的特性，因为我们这个网站是静态网站，暂时没有存在动态访问的问题；

1. 配置Nginx服务器

这里我使用的是Nginx的1.16.1版本；我会将该文件上传到我的当前git仓库中，可以直接下载进行配置；

upstream myserver{

#server 192.168.242.129:8888;

server localhost:8888;

#server 192.168.242.129:9999;

server localhost:9999;

}

location / {

root html;

# 2019年11月12日21:29:31 修改 实现反向代理多个服务器，实现负载均衡 myserver在上面已经写好

proxy\_pass http://myserver;

index index.html index.htm;

}

（2）启动Nginx服务器

/usr/local/nginx/sbin/nginx -c /usr/local/nginx/conf/nginx.conf

（3）解决传回目录的问题，访问文件是完全没有问题的，并且可以做到负载均衡；

后端能够收到请求，但是后端无法将数据完全传输完毕，由于我们直接访问开启的服务器时，可以直接显示网页内容，但是其实请求一直没有结束，数据一直处于传输当中；现在加上nginx服务器之后，如果还是这样的情况，nginx服务器无法收到所有的数据，没有结束传输，因此绝不会在前端进行显示，这是一个需要解决的问题；

1. 将title标签中的中文字符进行

加入meta标签

"<html><head><meta http-equiv=Content-Type content=\"text/html; charset=utf-8\"><title>哎~出错了</title></head>";