哈爾濱工業大學
Harbin Institute of Technology

计算机网络
课程实验报告

实验名称	HTTP 代理服务器的设计与实现				
姓名	朱宸慷		院系	计算机科学与技术	
班级	2103103		学号	2021110908	
任课教师	聂兰顺		指导教师	聂兰顺	
实验地点	格物 207		实验时间	2023-10-21	
实验课表现	出勤、表现得分(10)		实验报告得分(40)		实验总分	
	操作结果得分(50)				
教师评语					

哈尔滨工业大学计算学部
FACULTY OF COMPUTING, HIT

实验目的：

本次实验的主要目的：熟悉并掌握 Socket 网络编程的过程与技术；深入理解 HTTP 协议，掌握 HTTP 代理服务器的基本工作原理；掌握 HTTP 代理服务器设计与编程实现的基本技能。

实验内容：

(1) 设计并实现一个基本 HTTP 代理服务器。要求在指定端口（例如 8080）接收来自客户的 HTTP 请求并且根据其中的 URL 地址访问该地址所指向的 HTTP 服务器（原服务器），接收 HTTP 服务器的响应报文，并将响应报文转发给对应的客户进行浏览。

(2) 设计并实现一个支持 Cache 功能的 HTTP 代理服务器。要求能缓存原服务器响应的对象，并能够通过修改请求报文，向原服务器确认缓存对象是否是最新版本。

(3) 扩展 HTTP 代理服务器，支持如下功能：

a) 网站过滤：允许/不允许访问某些网站；

b) 用户过滤：支持/不支持某些用户访问外部网站；

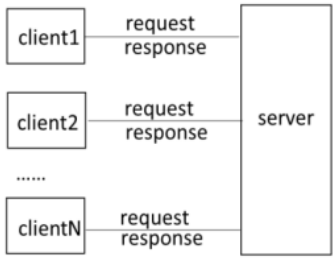
c) 网站引导：将用户对某个网站的访问引导至一个模拟网站（钓鱼）。

实验过程：

以文字描述、实验结果截图等形式阐述实验过程，必要时可附相应的代码截图或以附件形式提交。

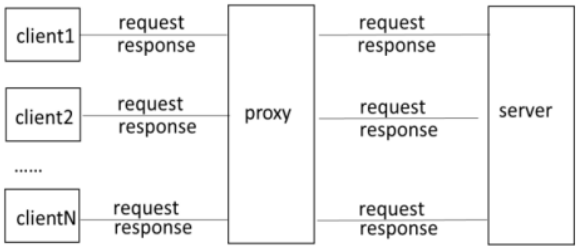
一、Socket 编程的客户端和服务器端主要步骤

代理服务器作为一个C/S架构的应用，需要分别从客户端和服务端进行工作的阐述。我们首先设想不存在这样一个代理服务器应用，于是，当我们想要使用浏览器访问页面时，浏览器进行了如下操作： 1. 将地址送往DNS服务器，获取目的服务器的IP地址和端口号后，创建一个套接字 2. 构造一个请求报文，通过套接字发往目的主机 3. 接收返回的报文，解析并使用这些报文 4. 传输完成后，关闭连接或者做其他操作。与之相对的，我们所请求的主机，需要完成以下操作： 1.创建套接字，绑定在本机地址上，等待连接加入 2. 与连接三次握手后，建立TCP连接 3. 接受请求报文 4. 返回响应报文。



一般B/S

此时我们再来看代理服务器，它所要做的其实就是一个C/S架构中的中间人——对于本地应用来说，代理就是目的服务器；对于目的服务器来说，代理就是客户。而在这个过程中，代理服务器不需要管报文实际上是干什么的。因此，我们首先只需要让代理服务器实现存储、转发报文的功能即可，而对于代理服务器的追加要求，也是建立在收发报文的基础上的：比如，当我们要实现cache时，实际上是向服务器发报询问更新日期，再决定转发本地报文还是服务器响应的报文；而过滤功能和钓鱼功能，则是在某些条件下决定是否返回报文和返回什么报文。

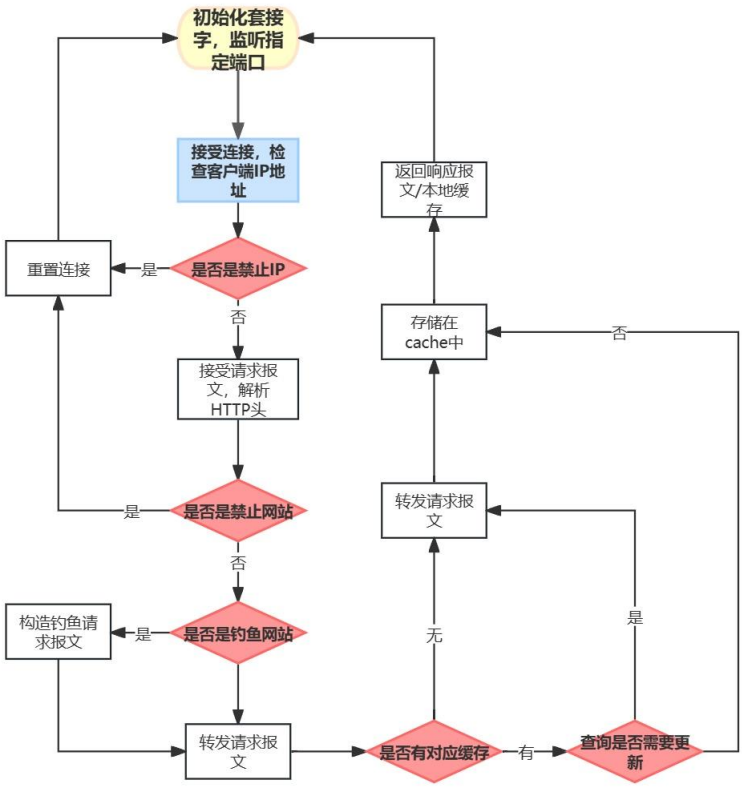


使用代理的B/S

二、HTTP 代理服务器的基本原理

代理服务器在指定端口（例如 8080）监听浏览器的访问请求，接收到浏览器对远程网站的浏览请求时，首先检查连接的IP是否属于被禁止的IP，若是，直接重置连接，不再进入后续的代理服务。当确认连接完成后，代理服务器开始检查HTTP头，确定目的主机是否为需要禁止的网站或钓鱼的网站，若是禁止网站，则关闭连接，若是钓鱼网站，则根据钓鱼信息构造新的HTTP报文。之后代理服务器开始在代理服务器的缓存中检索URL对应的对象（网页、 图像等对象），找到对象文件后，提取该对象文件的最新被修改时间；代理服务器程序在客户的请求报文首部插入的最新被修改时间>，并向原Web服务器转发修改后的请求报文。如果代理服务器没有该对象的缓存，则会直接向原服务器转发请求报文。并准备将原服务器返回的响应直接转发给客户端，同时将对象缓存到代理服务器中。代理服务器程序会根据缓存的时间、大小和提取记录等对缓存进行清理。

三、HTTP 代理服务器的程序流程图



四、实现 HTTP 代理服务器的关键技术及解决方案

（1） 初始化套接字，并不断等待接入

```
    printf("代理服务器正在启动\n");
    printf("初始化...\n");
    if (!InitSocket()) {
        printf("socket 初始化失败\n");
        return -1;
    }
    printf("代理服务器正在运行，监听端口 %d\n", ProxyPort);
```

（2） 屏蔽指定IP

首先将接受的网络二进制IP地址转换为点分十进制，通过字符串比较确定是否需要屏蔽该地址。如果需要屏蔽，直接continue当前循环，即重置连接。

```
        auto visit_ip = inet_ntoa(addr_in.sin_addr);
        //输出客户端IP地址
        printf("访问来自IP");
        printf(visit_ip);//将网络二进制的数字转换成网络地址
        printf("\n");

        // 检查是否在黑名单中
        if (strcmp(blocked_ip, visit_ip) == 0)//转换成网络地址
        {
            printf("%s用户禁止访问\n", visit_ip);
            continue;
        }
```

（3） 创建线程

```
        lpProxyParam->clientSocket = acceptSocket;//将客户端套接字赋值给线程参数
        hThread = (HANDLE)_beginthreadex(NULL, 0, &ProxyThread, (LPVOID)lpProxyParam, 0, 0);//创建线程
        CloseHandle(hThread);//关闭线程句柄
        Sleep(200);//延时200ms
```

（4） 解析HTTP头

提取报文中所含的信息，例如主机域名和cookie等

```
    //处理HTTP头部
    if (CacheBuffer)
    {
        ParseHttpHead(CacheBuffer, httpHeader);//将CacheBuffer中的内容解析到httpHeader中
    }
```

（5） 处理禁止访问网站

比较HTTP报文中的目的主机，如果需要屏蔽，就关闭套接字

```
    if (strstr(httpHeader->url, blocked_web) != NULL)//如果在黑名单中
    {
        printf("\n========================================\n");
        printf("--------该网站已被屏蔽!----------\n");
        printf("关闭套接字\n");
        Sleep(200);
        closesocket(((ProxyParam*)lpParameter)->clientSocket);
        closesocket(((ProxyParam*)lpParameter)->serverSocket);
        delete lpParameter;
        _endthreadex(0);
        return 0;
    }
```

（6） 处理钓鱼网站

如果目的主机需要钓鱼，就更改HTTP请求报文的目的主机

```
    //处理钓鱼网站
    if (strstr(httpHeader->url, fishing_src) != NULL)
    {
        printf("\n========================================\n");
        printf("---已从源钓鱼网址: %s 跳转到 目的网址 : %s ---\n", fishing_src, fishing_dest);
        //修改HTTP报文
        memcpy(httpHeader->host, fishing_dest_host, strlen(fishing_dest_host) + 1);//修改host, +1是复制进去\0
        memcpy(httpHeader->url, fishing_dest, strlen(fishing_dest));//修改url
    }
```

（7） 缓存cache

每当代理服务器收到一个响应报文时，就在本地存储，并通过HTTP报文头来区分不同的报文。当接收到客户端请求报文时，代理服务器首先比较本地是否有该报文，再向目

的主机发报文询问是否需要更新报文，若不需要，则将本地报文返回给客户端，若需要，则转发请求报文并将服务器的响应报文发回给客户端。

实验结果：

采用演示截图、文字说明等方式，给出本次实验的实验结果。

相关数据设置如下

```
7   //代理相关参数
8   SOCKET ProxyServer;
9   sockaddr_in ProxyServerAddr;
10  const int ProxyPort = 10240;
11
12  //禁止访问网站
13  const char* blocked_web = { "http://computing.hit.edu.cn/" };
14  //限制访问用户
15  const char* blocked_ip = { "127.0.0.2" };
16  //钓鱼网站
17  const char* fishing_src = "http://today.hit.edu.cn";//钓鱼网站原网址
18  const char* fishing_dest = "http://jwts.hit.edu.cn";//钓鱼网站目标网址
19  const char* fishing_dest_host = "jwts.hit.edu.cn";//钓鱼目的地址主机名
```

（1） 屏蔽指定 IP

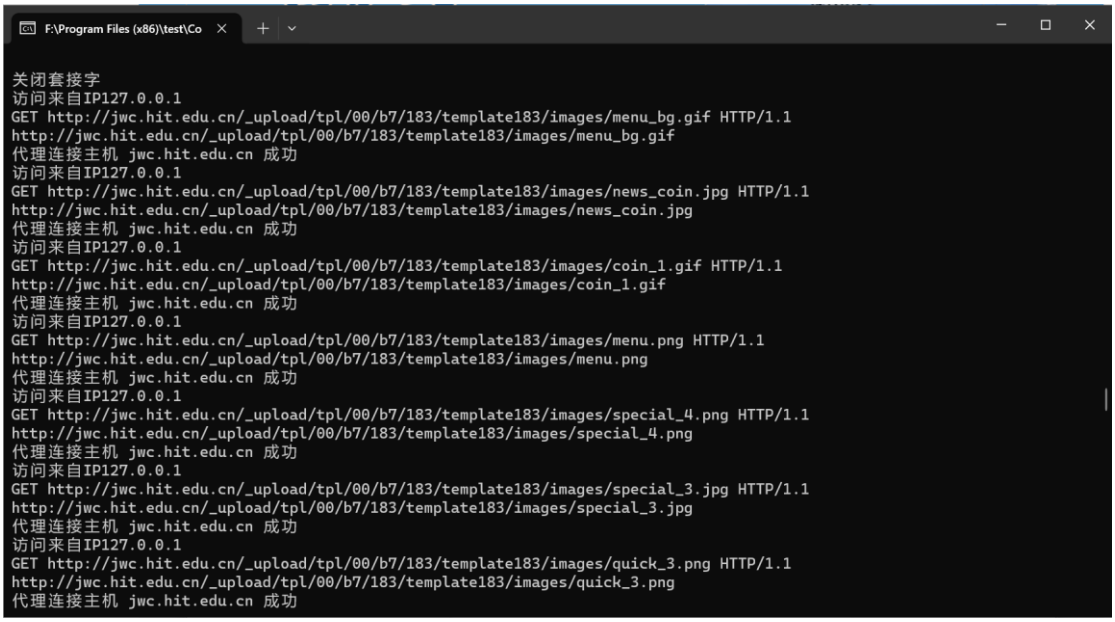
此时将头文件中指定的 blocked_ip 改为”127.0.0.1”，这是本地回环地址，理论上应该拦截所有连接。重新编译运行结果如下

```
F:\Program Files (x86)\test\Co
代理服务器正在启动
初始化...
代理服务器正在运行，监听端口 10240
访问来自IP127.0.0.1
127.0.0.1用户禁止访问
访问来自IP127.0.0.1
127.0.0.1用户禁止访问
访问来自IP127.0.0.1
127.0.0.1用户禁止访问
访问来自IP127.0.0.1
127.0.0.1用户禁止访问
访问来自IP127.0.0.1
127.0.0.1用户禁止访问
访问来自IP127.0.0.1
127.0.0.1用户禁止访问
访问来自IP127.0.0.1
127.0.0.1用户禁止访问
访问来自IP127.0.0.1
127.0.0.1用户禁止访问
访问来自IP127.0.0.1
127.0.0.1用户禁止访问
访问来自IP127.0.0.1
127.0.0.1用户禁止访问
访问来自IP127.0.0.1
127.0.0.1用户禁止访问
访问来自IP127.0.0.1
127.0.0.1用户禁止访问
访问来自IP127.0.0.1
127.0.0.1用户禁止访问
访问来自IP127.0.0.1
```

验证结果，发现所有连接都被代理服务器所重置。将 blocked_ip 改为”127.0.0.2”，进行后续结果验证

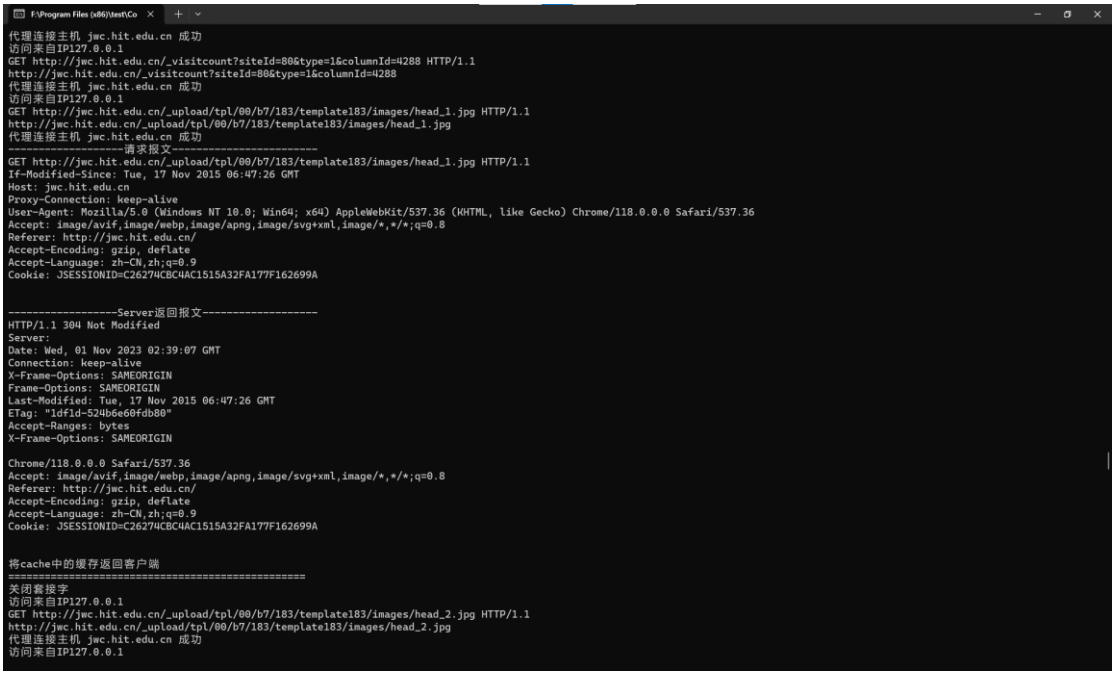
（2） 正常代理访问访问

经验证，http://jwc.hit.edu.cn/可以正常访问，代理服务器向主机 jwc.hit.edu.cn 请求资源并将报文转发给客户端



（3） 缓存功能验证

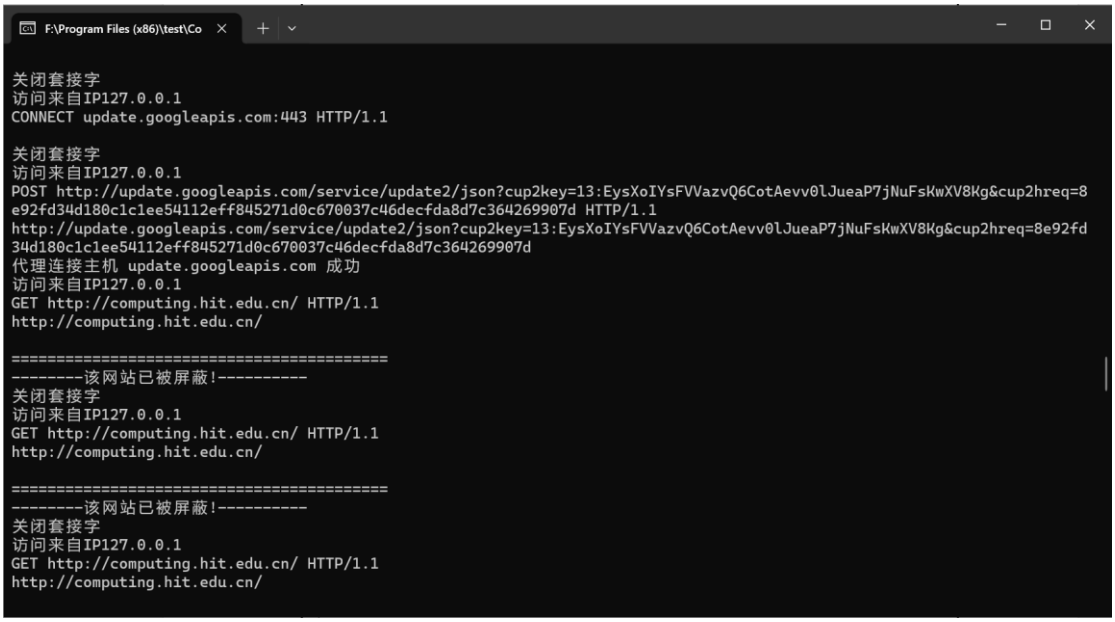
刷新 http://jwc.hit.edu.cn/，可以发现代理服务器向主机发送报文查询是否需要更新，并返回了本地缓存中的报文



（4） 屏蔽指定网页

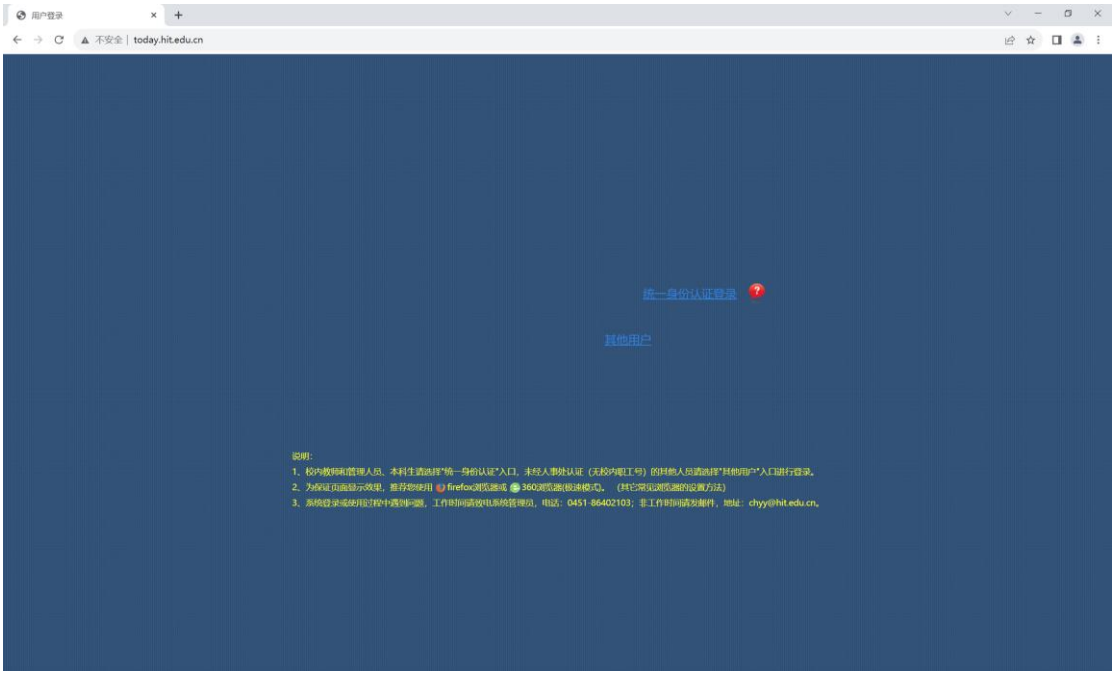
chrome 中显示，http://computing.hit.edu.cn/没有发送任何数据，这是因为代理服务器没有将报文转发给客户端。

观察代理服务器，可以发现该网站确实被正常屏蔽

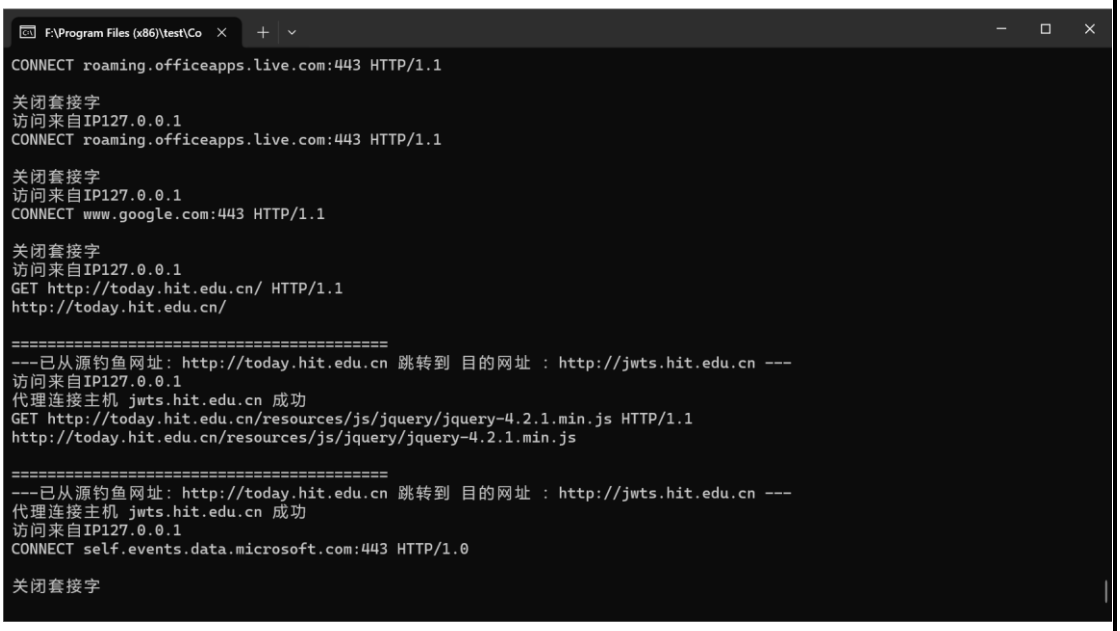


（5） 钓鱼网站

尝试访问 http://today.hit.edu.cn，但是实际上访问的是 http://jwts.hit.edu.cn



代理服务器检测到试图访问钓鱼网站 http://today.hit.edu.cn，并将连接重定向到 http://jwts.hit.edu.cn 。代理服务器在更改了客户端的请求报文头，将其导向钓鱼目的主机后转发。再将从目的主机返回的响应报文转发给客户端。



问题讨论：

对实验过程中的思考问题进行讨论或回答。

（1） 参考代码中使用了大量的goto语句，这在现代c++中是会造成问题的。编译器因为不能确定goto语句之后到goto目标之间的代码段是否可以被访问，而这些代码段中又定义了变量，因此vs认为这是一种危险的未定义行为，不能正常地编译链接cpp文件。在实验过程中可以通过将goto目标直接转移到原goto语句处来解决。

（2） 在使用代理服务器访问时其实是远慢于正常访问的，并且图片与css经常不能够正常加载出来。经过排查，应该不是网速的问题，猜测应该是代理服务器的运行类似于停等，每一个资源都需要等待一段时间传输，对于有大量资源的网页接受速度慢，而正常情况下代理服务器应该是使用流水线方式的，速度较快。

心得体会：

结合实验过程和结果给出实验的体会和收获。

1. 学习到了代理服务器的运行原理
2. 对于HTTP协议的理解进一步的加深了
3. 对于socket编程有了些基本的概念