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	实验 1：HTTP 代理服务器的设计与实现				
姓名	左镕畅		院系	未来技术学院	
班级	21R0361		学号	2021110788	
任课教师	刘亚维		指导教师	刘亚维	
实验地点	G002		实验时间	2024.04.16	
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：

1. 熟悉并掌握 Socket 网络编程的过程与技术；
2. 深入理解 HTTP 协议，掌握 HTTP 代理服务器的基本工作原理；
3. 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（添加 if-modified-since 头行），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）。

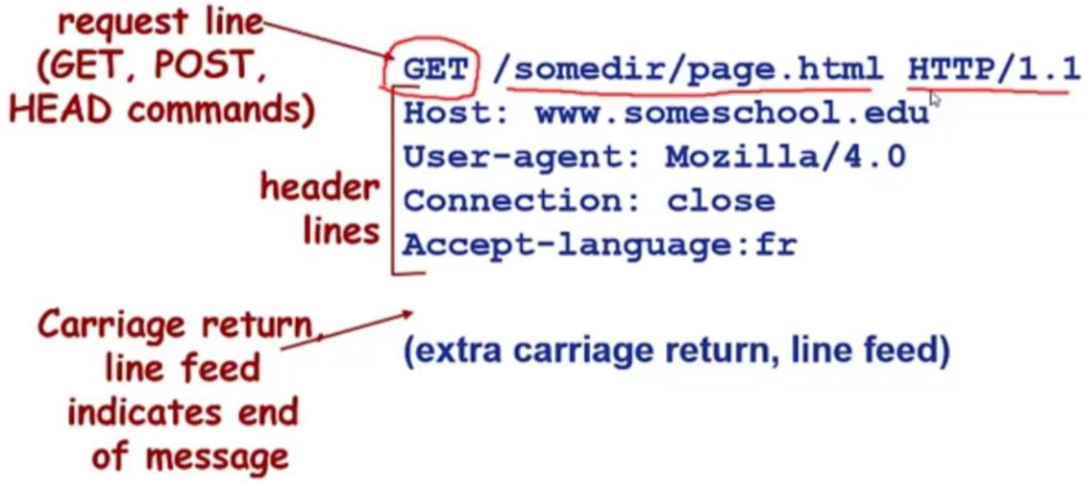
实验过程：

1. 首先分析指导书中提供的代码，并实现一个基本的HTTP代理服务器（由于代码均为指导书中代码，报告中没有给出截图）

a) 在main函数中，调用了InitSocket函数初始化了代理服务器的接收端Socket：使用bind函数绑定了接收端Socket的IP地址和端口，其中IP地址定为INADDR_ANY，即本机所有可用IP，端口定为ProxyPort，即监听端口（用于监听来自客户端的连接请求）。随后使用listen函数将端口设置为监听模式

b) 随后main函数在while循环中不断监听端口，并发处理客户端连接请求。使用了阻塞函数accept对每个连接请求建立连接，当连接的用户许可时（后续说明），建立连接，并创建新线程执行ProxyThread函数，用于处理客户端请求报文

c) 在ProxyThread函数中，首先使用recv函数接收客户端的请求报文。随后开始解析HTTP头部，HTTP头部如下所示，通过调用ParseHttpHead函数，可以抽取出请求方法（GET or POST）、URL、Host、Cookie。接着调用ConnectToserver函数创建于目标服务器连接的Socket，并与目标服务器连接。连接后，使用send函数向目标服务器发送接收到的报文，同时使用recv函数接收目标服务器的响应报文，最后再用send函数将响应报文发送给客户端。



d) 如果在ProxyThread函数的执行过程中出现异常，会跳转至error，关闭Socket（包括发送端和

接收端的Socket），并结束线程

2. 实现缓存以及缓存更新功能

a) 首先我们实现缓存功能。在代理服务器接收到目标服务器的响应报文时，我们将该缓存覆盖到已有的缓存文件中（如果已有缓存），或者新建并写入到一个新的缓存文件中（如果没有缓存）。具体实现方式如下：首先创建缓存文件的名称，由于URL的唯一性，选择提取URL作为缓存文件名，随后检查是否已存在该名称的缓存文件，并进行新建写入或覆写操作。以下为创建缓存名的makeCachename函数和新建或更新缓存文件的makeCache函数

```
/*构造缓存文件名*/
void makeCachename(char* url, char* cachename)
{
    int count = 0;
    while (*url != '\0') {
        if ((*url >= 'a' && *url <= 'z') || (*url >= 'A' && *url <= 'Z') || (*url >= '0' && *url <= '9'))
        {
            *cachename++ = *url;
            count++;
        }
        if(count >= 100) break;
        url++;
    }
    strcat(cachename, ".txt");
}
```

```
/*新建或更新缓存文件*/
void makeCache(char* buffer, char* url)
{
    char* p, * ptr, tempBuffer[MAXSIZE + 1];
    ZeroMemory(tempBuffer, MAXSIZE + 1);
    const char* delim = "\r\n";
    memcpy(tempBuffer, buffer, strlen(buffer));

    p = strtok_s(tempBuffer, delim, &ptr);  // 获取HTTP响应的第一行，包含状态码

    if (strstr(tempBuffer, "200") != NULL) // 检查第一行中是否包含状态码"200"
    {
        char cachename[105] = { 0 };
        makeCachename(url, cachename);

        FILE* out;
        fopen_s(&out, (cacheDir+cachename).c_str(), "w+");  // 使用fopen_s打开或创建文件，如果文件存在则清空
        fwrite(buffer, sizeof(char), strlen(buffer), out);  // 将buffer写入文件
        fclose(out);

        printf("新建或更新本地缓存成功，缓存名：%s\n", cachename);  // 打印缓存成功的消息
    }
}
```

b) 接下来需要实现缓存文件的更新问题。在代理服务器向目标服务器转发客户端的请求报文时，如果我们已经存在了Cache文件，那么我们需要将Cache文件的日期添加到请求报文的“if-modified-since”中，再发送给目标服务器，以确认Cache文件是否是最新版本。如果是最新版本则无需更新。

通过查找缓存文件中的Date字段我们可以获取缓存文件的日期，获取日期的getCachedate函数如下

```
/*获取缓存文件中的日期*/
void getCachedate(FILE* in, char* date)
{
    // 逐行寻找，直到包含 "Date" 字段的行
    char target[5] = "Date";
    char *p, *ptr;
    char buffer[MAXSIZE];
    ZeroMemory(buffer, MAXSIZE);
    fread(buffer, sizeof(char), MAXSIZE, in);
    const char* delim = "\r\n"; //换行符
    p = strtok_s(buffer, delim, &ptr);  // 提取一行
    int len = strlen(target) + 2;   // 只需要“Date: ”后的
    while (p)
    {
        if(strstr(p, target) != NULL)   // 是否有匹配的
        {
            memcpy(date, &p[len], strlen(p) - len);
            return;
        }
        p = strtok_s(NULL, delim, &ptr);
    }
}
```

通过addDate函数，我们可以在即将转发给目标服务器的请求报文的HTTP头中添加包含缓存日期信息的“if-modified-since”字段

```
/*为HTTP请求报文添加字段*/
void addDate(char* buffer, char* date)
{
    const char* field = "Host";
    const char* newfield = "if-modified-since: ";  // 定义将要插入的新字段 "if-modified-since:"
    //const char *delim = "\r\n";

    char temp[MAXSIZE];  // 临时数组，用于存储原始数据的副本
    ZeroMemory(temp, MAXSIZE);

    char* pos = strstr(buffer, field);  // 找到"Host"字段的位置

    for(int i = 0; i < strlen(pos); i++) temp[i] = pos[i];  // 备份
    *pos = '\0';  // 在"Host"的位置插入字符串结束符，截断原始buffer

    while(*newfield != '\0') *pos++ = *newfield++;      // 插入"if-modified-since: "

    while(*date != '\0') *pos++ = *date++;  // 插入日期
    *pos++ = '\r';  // 在日期后添加回车符
    *pos++ = '\n';  // 添加换行符

    for(int i = 0; i < strlen(temp); i++)  *pos++ = temp[i];  // 将temp中的数据复制回buffer
}
```

与此同时，我们还需要使用两个bool变量来维护是否需要新建、更新缓存。当提取URL后，我们检查是否存在缓存文件，如果存在，将ifHave变量设置为true表示存在缓存；当接收到目标服务器的响应报文后，如果ifHave为true，那么调用useCache函数获取响应中的状态码，如果状态码为304，则说明页面未被修改，可以将ifMake设置为false，表示不需要新建或更新缓存。如果状态码为200，表示缓存被修改，则将ifMake设置为true，并调用makeCache函数新建或更新缓存。以下为useCache函数

```
/*判断是否使用本地的缓存文件*/
boolean useCache(char* buffer, char* cachename)
{
    char *p, *ptr, tempBuffer[MAXSIZE + 1];
    ZeroMemory(tempBuffer, MAXSIZE + 1);
    const char* delim = "\r\n";  // 分隔符
    memcpy(tempBuffer, buffer, strlen(buffer));

    p = strtok_s(tempBuffer, delim, &ptr);  // 找到并返回第一行，通常是状态行

    // 主机返回的报文中的状态码为304时返回已缓存的内容
    if (strstr(p, "304") != NULL)  // 检查是否包含"304"状态码
    {
        printf("使用本地缓存\n");
        ZeroMemory(buffer, strlen(buffer));  // 清空原始buffer
        FILE* in = NULL;
        if (fopen_s(&in, (cacheDir+cachename).c_str(), "r") == 0)
        {
            fread(buffer, sizeof(char), MAXSIZE, in);  // 从缓存中读取数据到buffer
            fclose(in);
        }
        return false;  // 返回false，不需要更新缓存
    }
    printf("需要更新本地缓存\n");
    return true;    // 需要更新缓存

}
```

3. 网站过滤功能：不允许访问指定网站

在ProxyThread函数中得到目的服务器的URL后，可以检查该URL是否与预设的需屏蔽的网站相同，如果相同，则不向该服务器发送数据，而是跳转到error结束线程

```
// 网站过滤
if(strstr(httpHeader->url, banedWeb) != NULL)
{
    printf("禁止访问 %s\n", banedWeb);
    goto error;
}
```

4. 用户过滤功能：不允许特定用户访问外部网站

在main函数监听客户端连接部分，当连接的用户的IP是被ban的用户的IP时，忽略该用户的连接请求

```
while (true)
{
    acceptSocket = accept(ProxyServer, (SOCKADDR*)&acceptAdd, &addLen); // 4. 阻塞函数accept对每个到来的连接请求建立连接
    lpProxyParam = new ProxyParam;
    if (lpProxyParam == NULL)
    {
        continue;
    }
    // 实现不允许部分用户连接
    if(strcmp(inet_ntoa(acceptAdd.sin_addr), banedIP)==0 && ifBanUser==1)
    {
        printf("用户 %s 禁止访问\n", banedIP);
        continue;
    }
    printf("用户 %s 已连接\n", inet_ntoa(acceptAdd.sin_addr));
    lpProxyParam->clientSocket = acceptSocket;
    hThread = (HANDLE)_beginthreadex(NULL, 0,&ProxyThread, (LPVOID)lpProxyParam, 0, 0); // 创建新线程，执行ProxyThread函数
    CloseHandle(hThread);   // 关闭线程句柄，但子线程仍在运行
    Sleep(200);
}
closesocket(ProxyServer);
```

5. 网站引导功能：将用户对某个网站的访问引导至一个模拟网站

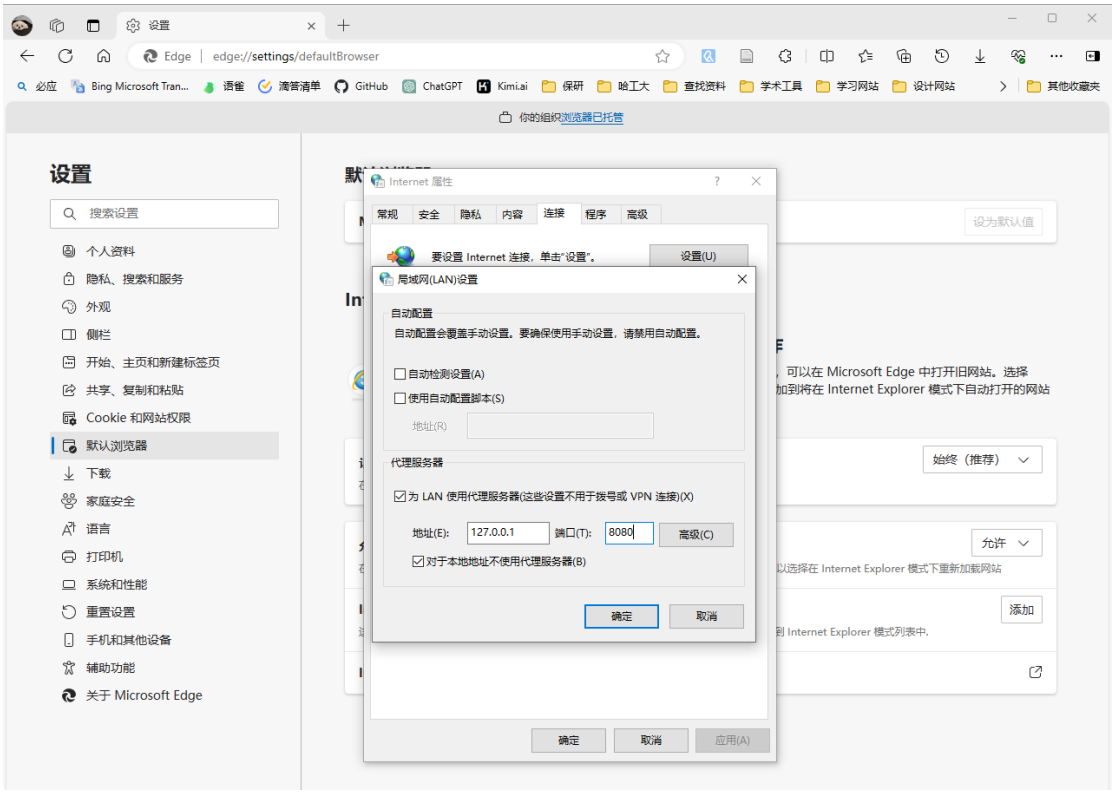
在ProxyThread函数中得到目的服务器的URL后，如果该URL与预设的引导原网站相同，则修改提取到的目的服务器的Host、URL，将其改为引导目的网站的，这样转发请求时就会转发至引导目的网站，实现网站引导

```
// 网站钓鱼
if(strstr(httpHeader->url, fishWebSrc) != NULL)
{
    memcpy(httpHeader->host, fishWebHost, strlen(fishWebHost) + 1);
    memcpy(httpHeader->url, fishWebTarget, strlen(fishWebTarget) + 1);
    printf("钓鱼成功，已引导至 %s\n", fishWebTarget);
}
```

实验结果：

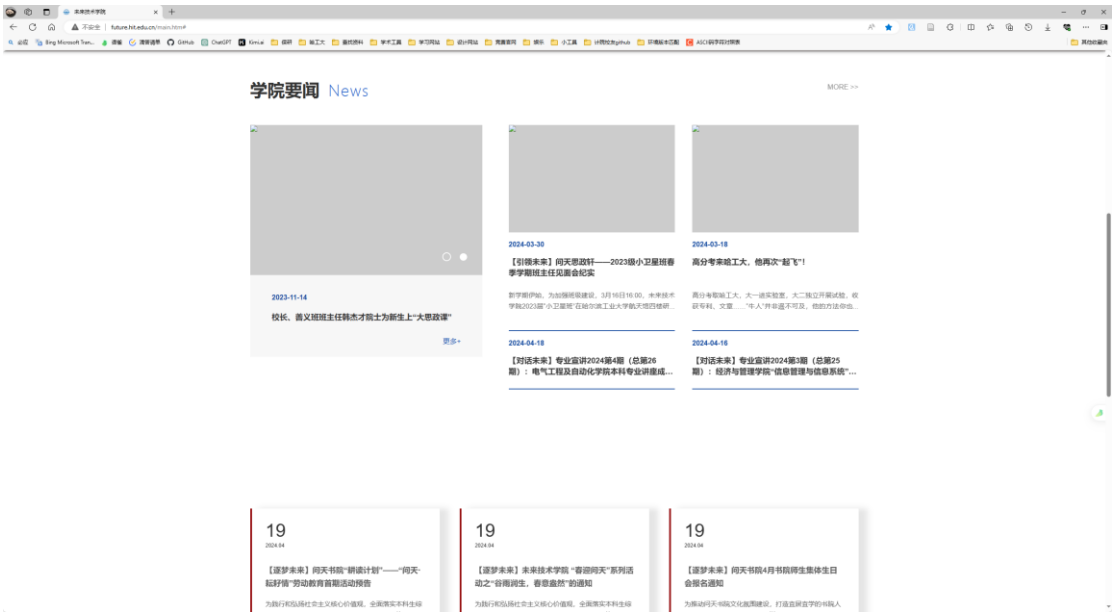
1. **修改Edge浏览器的代理设置**

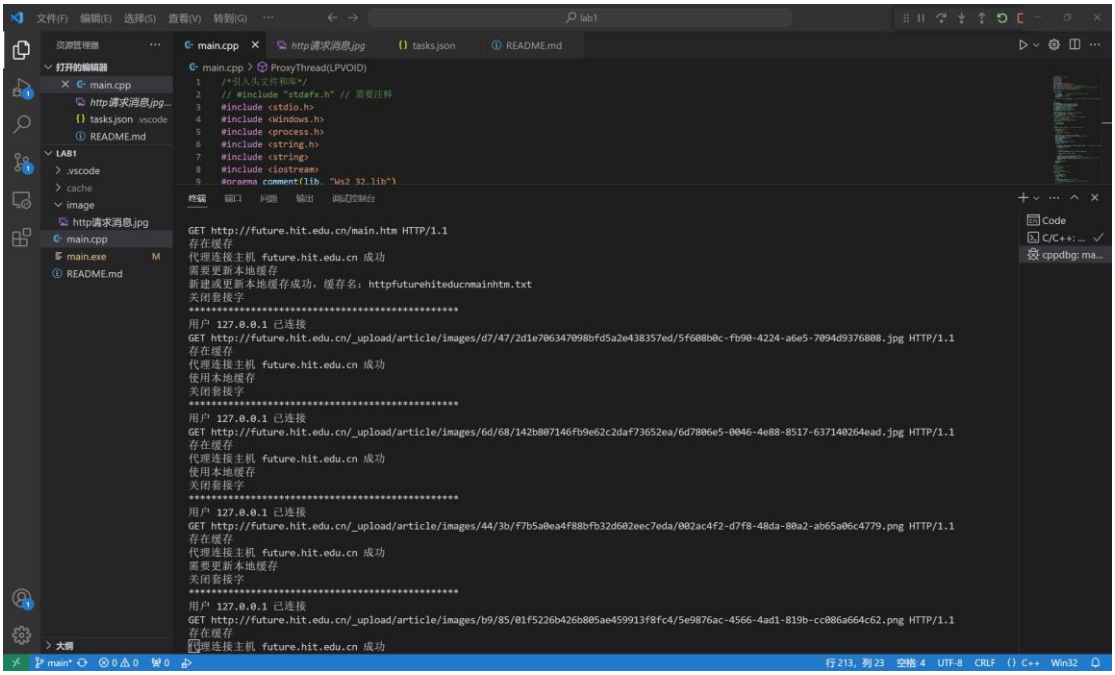
IP地址设为本地回环地址127.0.0.1。由于代码中的代理服务器的监听端口为8080，所以端口号设置为8080



2. **实现基本代理服务器功能。接收来自客户的HTTP请求，并根据其中的URL地址访问目标服务器，接收目标服务器的响应报文，并将响应报文转发给客户进行浏览**

编译并运行主程序，随后输入0，表示不对用户127.0.0.1进行过滤，然后打开Edge浏览器，打开一个使用HTTP协议的网站，例如：未来技术学院官网http://future.hit.edu.cn/，可以发现网站正常访问



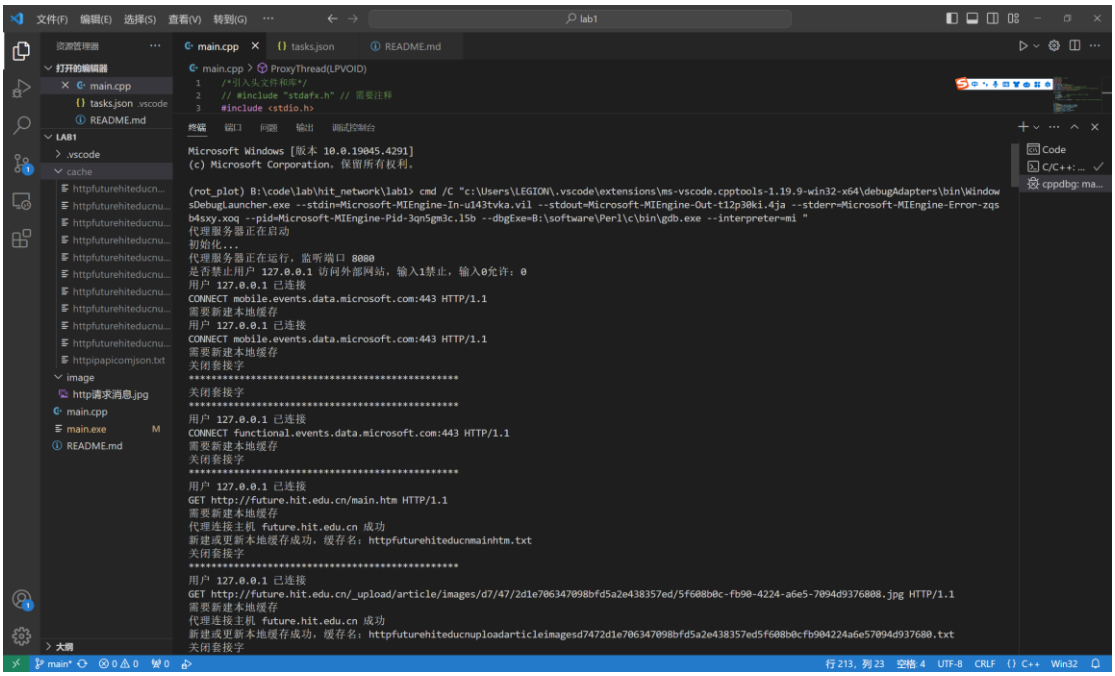


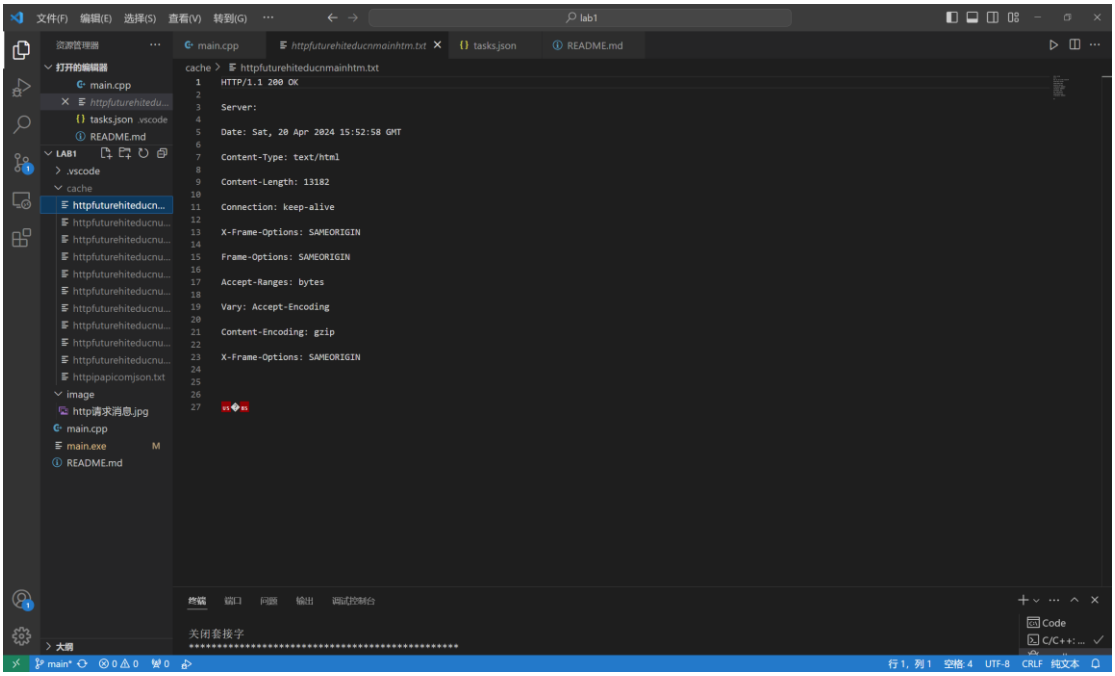
3. 验证代理服务器的Cache功能

首先清空所有Cache文件，随后再次访问未来技术学院官网http://future.hit.edu.cn/，检查以下结果：

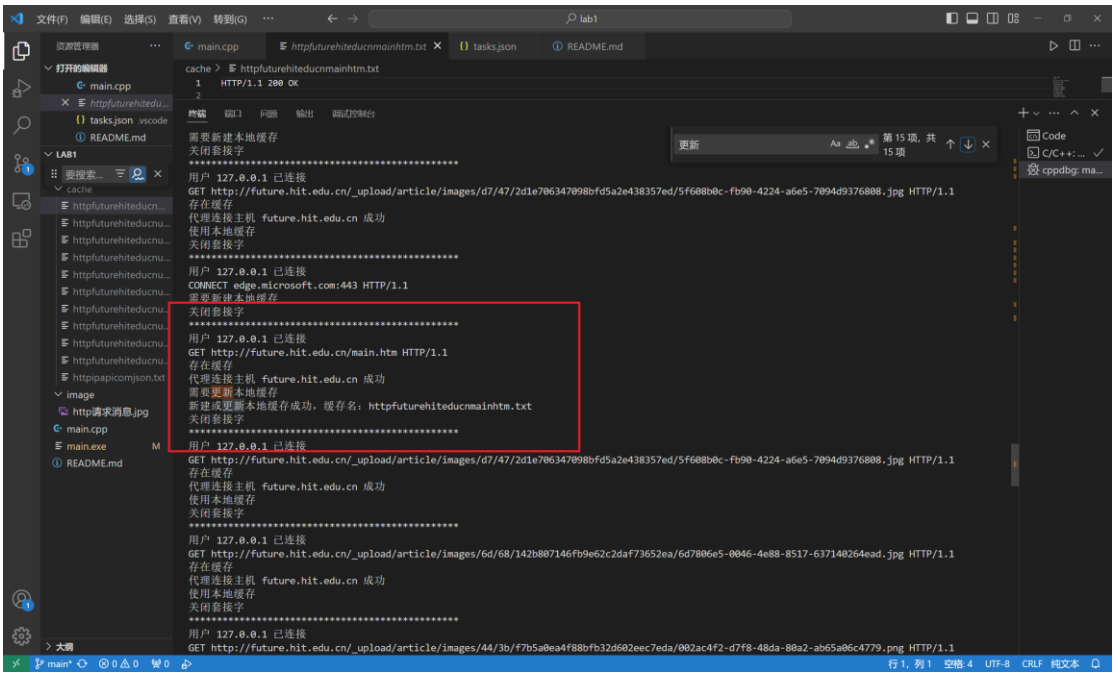
新建Cache、更新Cache、使用本地Cache（当本地有Cache且无需更新）

新建Cache（包括提示信息及Cache文件内容）

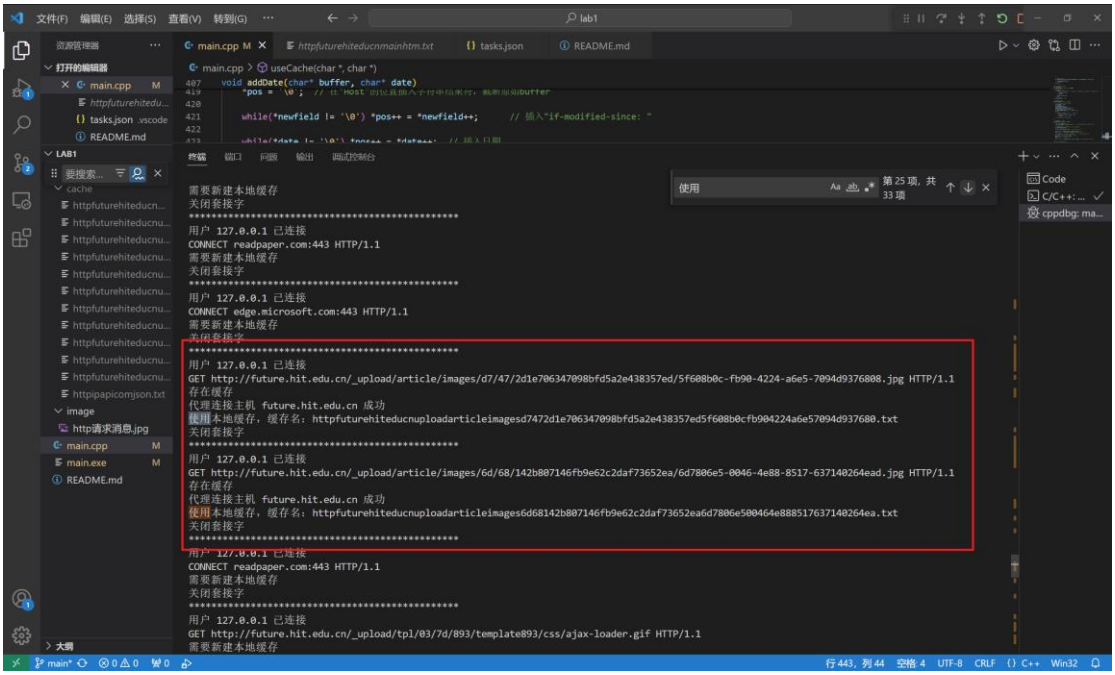

Microsoft Windows [版本 10.0.19045.4291]
(c) Microsoft Corporation。保留所有权利。
(rot_plot) B:\code\lab\hit_network\lab1> cmd /C "c:\Users\LEGION\.vscode\extensions\ms-vscode.cpptools-1.19.9-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe --stdin=Microsoft-MIEngine-In-u143tvka.vil --stdout=Microsoft-MIEngine-Out-t12p30Ki.4ja --stderr=Microsoft-MIEngine-Error-zqsb4sxy.xoq --pid=Microsoft-MIEngine-Pid-3qn5gm3c.15b --dbgExe=B:\software\Perl\c\bin\gdb.exe --interpreter=mi "
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 8080
是否禁止用户 127.0.0.1 访问外部网站，输入1禁止，输入0允许：0
用户 127.0.0.1 已连接
CONNECT mobile.events.data.microsoft.com:443 HTTP/1.1
需要新建本地缓存
用户 127.0.0.1 已连接
CONNECT mobile.events.data.microsoft.com:443 HTTP/1.1
需要新建本地缓存
关闭套接字
关闭套接字
用户 127.0.0.1 已连接
CONNECT functional.events.data.microsoft.com:443 HTTP/1.1
需要新建本地缓存
关闭套接字
用户 127.0.0.1 已连接
GET http://future.hit.edu.cn/main.htm HTTP/1.1
需要新建本地缓存
代理连接主机 future.hit.edu.cn 成功
新建或更新本地缓存成功，缓存名：httpfuturehiteducnmainhtm.txt
关闭套接字
用户 127.0.0.1 已连接
GET http://future.hit.edu.cn/_upload/article/images/d7/47/2d1e706347098bfd5a2e438357ed/5f608b0c-fb90-4224-a6e5-7094d9376808.jpg HTTP/1.1
需要新建本地缓存
代理连接主机 future.hit.edu.cn 成功
新建或更新本地缓存成功，缓存名：httpfuturehiteducnuploadarticleimagesd7472d1e706347098bfd5a2e438357ed5f608b0cfb904224a6e57094d937680.txt
关闭套接字


HTTP/1.1 200 OK
Server:
Date: Sat, 20 Apr 2024 15:52:58 GMT
Content-Type: text/html
Content-Length: 13182
Connection: keep-alive
X-Frame-Options: SAMEORIGIN
Frame-Options: SAMEORIGIN
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Encoding: gzip
X-Frame-Options: SAMEORIGIN
关闭套接字

更新Cache

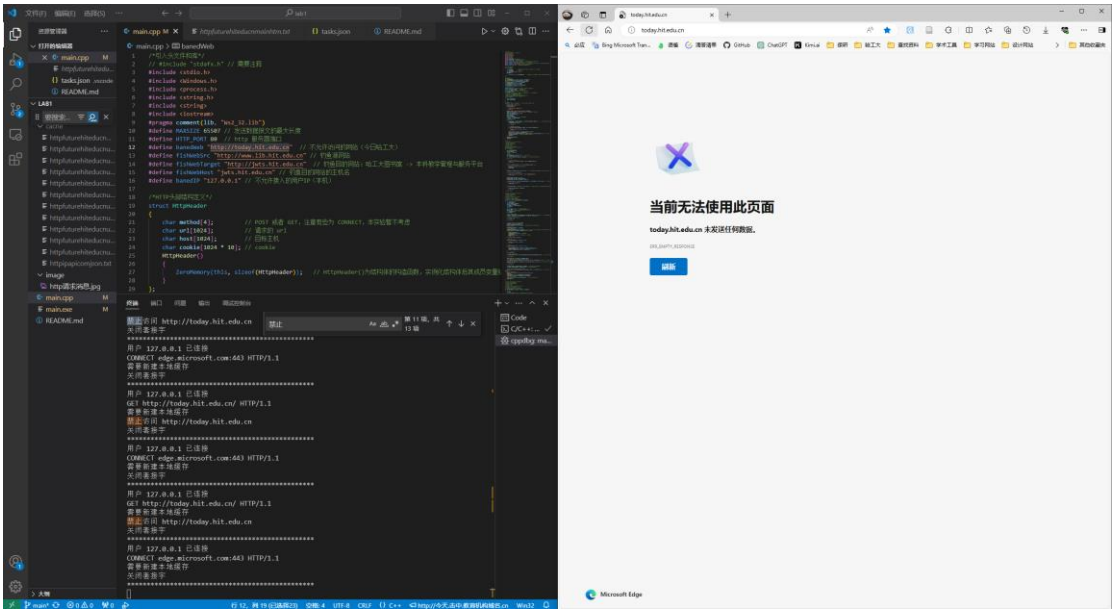


使用本地Cache



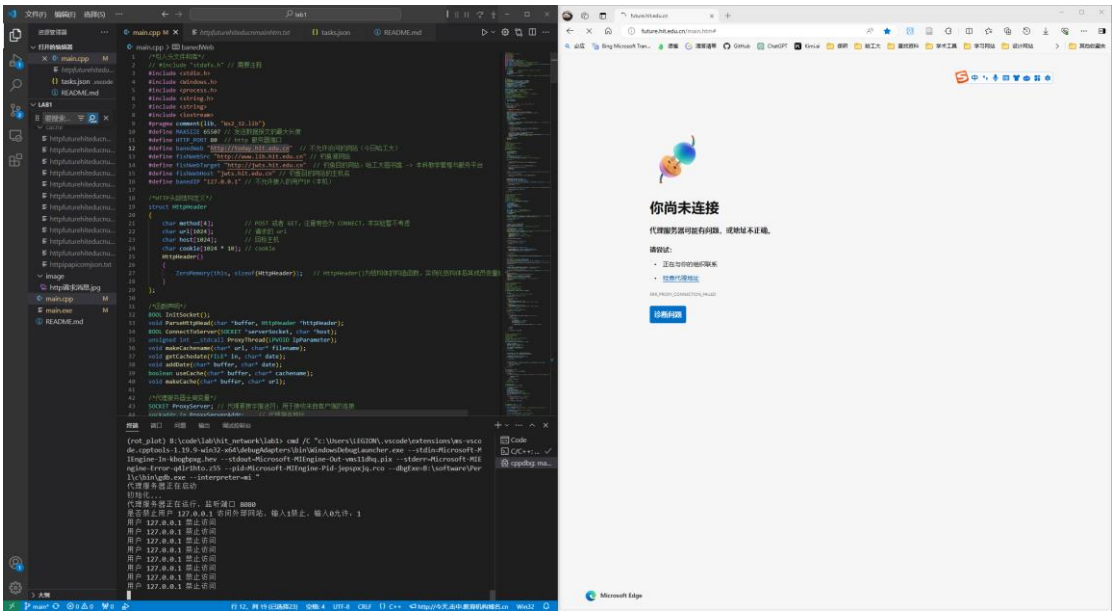
4. 网站过滤

代码设置的屏蔽网站为今日哈工大，地址为http://today.hit.edu.cn/，当尝试访问该网址时，网页无法使用，同时程序给出提示：禁止访问网页



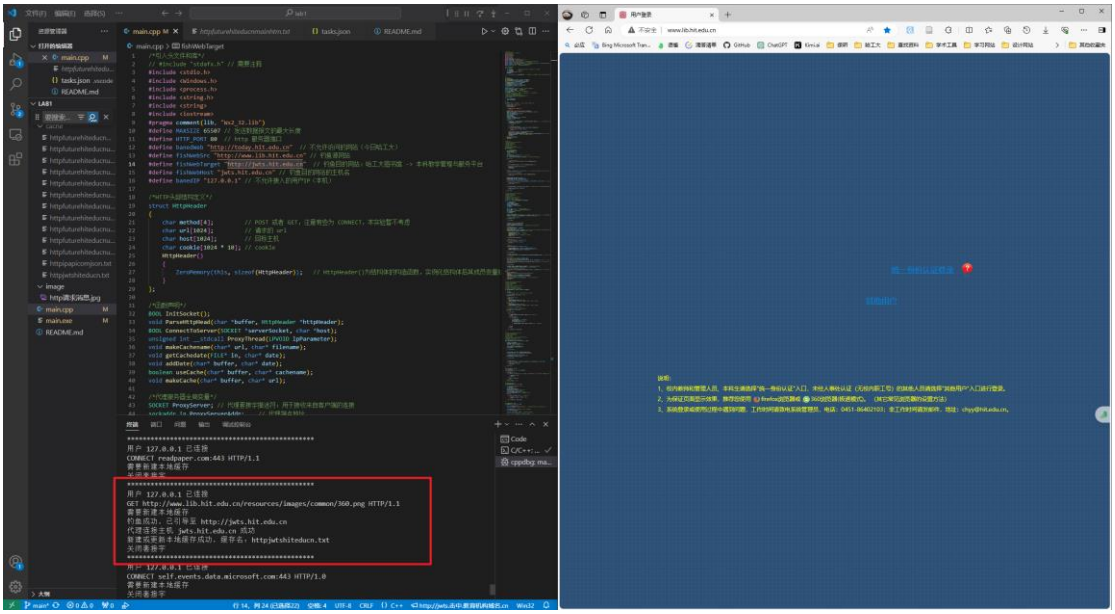
5. 用户过滤

代码屏蔽的用户是127.0.0.1，即本机。在运行代码时，输入1表示屏蔽该用户。输入后，所有网页包括刚刚能打开的未来技术学院官网，都无法打开，同时程序给出提示：用户禁止访问



6. 网站引导

本代码的钓鱼源网站为哈工大图书馆http://www.lib.hit.edu.cn/，钓鱼目的网站为本科教学管理与服务平台http://jwts.hit.edu.cn/。运行程序并访问哈工大图书馆，会发现网页被成功重定向，尽管地址栏仍显示哈工大图书馆URL，但网页被引导至了本科教学管理与服务平台，同时程序也输出了钓鱼成功的提示



问题讨论：

1. Socket编程的客户端和服务端的主要步骤

a) 客户端主要步骤

初始化套接字库 → 创建发送端Socket → 绑定套接字的端点地址 → 向服务器发送连接请求 → 连接建立 → 向服务器发送请求报文，并等待服务器的响应报文 → 关闭连接 → 关闭套接字

具体步骤如下：

i. 初始化套接字库

ii. 创建发送端Socket：使用socket()函数创建一个套接字，这个函数会返回一个套接字描述符，用于后续操作。

iii. 绑定套接字的端点地址：使用bind()函数将套接字与一个本地地址和端口绑定。对于客户端而言，这一步通常是可选的。

iv. 向服务器发送连接请求：使用connect()函数向服务器的IP地址和端口发起连接请求。

v. 连接建立：一旦connect()成功，客户端和服务端之间的TCP连接就建立了。

vi. 向服务器发送请求报文，并等待服务器的响应报文：通过套接字描述符使用send()和recv()函数进行数据的发送和接收。

vii. 关闭连接

viii. 关闭套接字：最后，使用closesocket()函数关闭套接字描述符，释放系统资源。

b) 服务端主要步骤

初始化套接字库 → 创建监听Socket → 绑定套接字的端点地址 → 监听传入的连接请求 → 接受连接请求 → 与客户端通信 → 关闭与客户端的连接 → 关闭套接字

具体步骤如下

i. 初始化套接字库

ii. 创建监听Socket：使用socket()函数创建一个用于监听的套接字。

iii. 绑定套接字的端点地址：使用bind()函数将套接字与一个本地地址和端口绑定。这一步是必要的，因为它将套接字与特定的网络接口和端口关联起来。

iv. 监听传入的连接请求：使用listen()函数将套接字设置为监听状态，准备接受来自客户端的连接请求。

v. 接受连接请求：使用accept()函数接受一个传入的连接请求。accept()函数会创建一个

新的套接字描述符，专门用于与已连接的客户端通信。

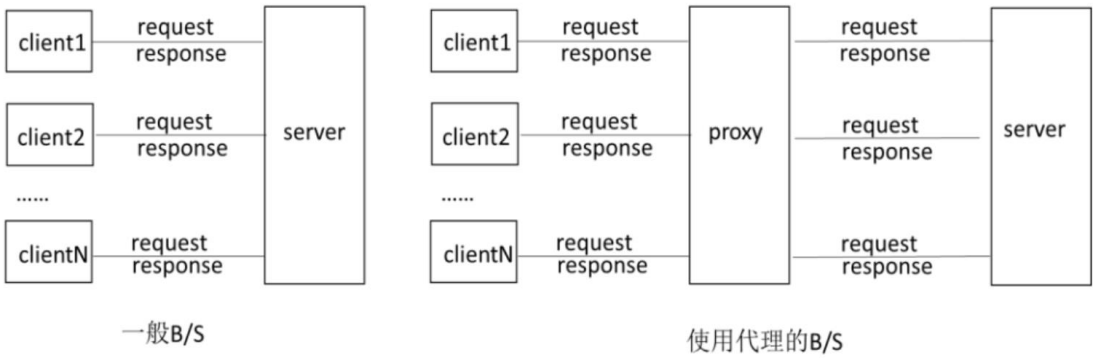
vi. 与客户端通信：使用recv()和send()函数与客户端进行数据的接收和发送。

vii. 关闭与客户端的连接

viii. 关闭套接字：使用closesocket()函数关闭与客户端通信的套接字描述符，释放系统资源。

2. HTTP代理服务器的基本原理

代理服务器，俗称“翻墙软件”，允许一个网络终端（一般为客户端）通过这个服务与另一个网络终端（一般为服务器）进行非直接的连接。如下图所示，为普通 Web 应用通信方式与采用代理服务器的通信方式的对比。



代理服务器在指定端口（例如8080）监听浏览器的访问请求（需要在客户端浏览器进行相应的设置），接收到浏览器对远程网站的浏览请求时，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<If-Modified-Since: 对象文件的最新被修改时间>，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。

3. HTTP代理服务器的程序流程图

4. 实现HTTP代理服务器的关键技术及解决方案

多用户代理服务器：多用户的简单代理服务器可以实现为一个多线程并发服务器。首先，代理服务器创建HTTP代理服务的TCP主套接字，通过该主套接字监听等待客户端的连接请求。当客户端连接之后，创建一个子线程，由子线程执行一对一的代理过程，服务结束之后子线程终止。与此同时，主线程继续接受下一个客户的代理服务。

在代码中，多用户代理服务器通过代码：`hThread = (HANDLE)_beginthreadex(NULL, 0,&ProxyThread, (LPVOID)lpProxyParam, 0, 0);` 实现，该代码创建了一个子线程用于执行ProxyThread函数，执行一对一的代理过程

Cache技术：在代理服务器接收到目标服务器的响应报文时，我们将该缓存覆盖到已有的缓存文件中（如果已有缓存），或者新建并写入到一个新的缓存文件中（如果没有缓存）。具体实现方式如下：首先创建缓存文件的名称，由于URL的唯一性，选择提取URL作为缓存文件名，随后检查是否已存在该名称的缓存文件，并进行新建写入或覆写操作。

5. HTTP代理服务器实验验证过程以及实验结果

见上文

6. HTTP代理服务器的源代码

见压缩包中main.cpp文件

心得体会：
1. 对HTTP代理服务器有了更深刻的理解，明白了它的具体原理以及实现过程 2. 加深了对TCP传输协议的理解 3. 初步掌握了Socket编程的能力 4. 通过实际编写代码，提升了我的实践编程能力 5. 对代理服务器的扩展功能，例如网站屏蔽、网站引导有了初步的了解