ICS（二），2021 年春季
实验室 3：Web 代理服务器
分配日期：2021 年 5 月 18 日，截止日期：2021 年 6 月 13 日，晚上 11：59

谢东方（20212010037@fudan.edu.cn）是本次任务的负责人。由于一些未知的遗留问题，第三部分被一些前 TA 删除。但考虑到缓存的重要性，我们最后添加了第三部分。为了得到补偿，我们给你更多的时间来完成它。祝您好运！

产品简介

Web 代理是一种在 Web 浏览器和端服务器之间充当中间人的程序。浏览器不是直接联系终端服务器以获取网页，而是联系代理，该代理将请求转发到终端服务器。当最终服务器响应代理时，代理将答复发送到浏览器。

代理被用于许多用途。有时代理用于防火墙中，因此代理是防火墙内浏览器与外部终端服务器联系的唯一方式。例如，代理可以在页面上进行转换，以使其可以在启用 Web 的手机上查看。代理也被用作匿名器。通过删除所有标识信息的请求，代理可以使浏览器对最终服务器匿名。代理甚至可以用来缓存 Web 对象，通过存储图像的副本，例如第一次发出请求，然后响应未来的请求服务图像，而不是去最终服务器。

在此实验室中，您将编写一个记录请求的并发 Web 代理。在实验室的第一部分，您将编写一个简单的顺序代理，它重复地等待请求，将请求转发到最终服务器，并将结果返回回浏览器，保留这些请求在磁盘文件中的日志。本部分将帮助您了解有关网络编程和 HTTP 协议的基本知识。

在实验室的第二部分中，您将升级您的代理，以便它使用线程同时处理多个客户端。这部分将给您一些并发和同步的经验，这是非常重要的计算机系统概念。

在第三部分也是最后一部分中，您将使用最近访问的 web 内容的简单主存缓存向代理添加缓存。

物流服务

你将独自在这个实验室上工作。对作业的任何澄清和修改都将张贴在课程网页上。

分发使用说明

从 svn 服务器上检查该实验室。用户名是您的学生身份证，密码以前已通过电子邮件发送给您。命令：svn 签出 svn：//10.176.63.161/labInfo--用户名=[您的学生身份证]。您可以在 labInfo/proxylab/ 中找到以下所有文件。

- 代理。c：这是您唯一要修改和提交的文件。它包含了代理的大部分逻辑。
- csapp.c：这是 CS：APP 教科书中描述的同名文件。它包含错误处理包装器和辅助功能，如里约热内卢(稳健 I/O)包(CS：APP11.4)、opeˆclientfd(CS：APP12.4.4)和 opeˆ监听程序(CS：APP12.4.7)。
- csapp.h：此文件包含 csapp.c 中函数的一些清单常数、类型定义和原型。
- 生成文件：编译和将 proxy.c 和 csapp.c 链接到可执行代理中。

proxy.c 文件可以调用 csapp.c 文件中的任何函数。

第一部分：实现一个顺序的 Web 代理

在本部分中，您将实现一个顺序的日志记录代理。您的代理应该打开一个套接字并侦听连接请求。当它接收到连接请求时，它应接受该连接，读取 HTTP 请求，并对其进行解析以确定最终服务器的名称。然后打开到终端服务器的连接，将请求发送，接收回复，如果请求未被阻止，将回复转发到浏览器。

由于您的代理是客户端和端服务器之间的中间人，因此它将具有两者之间的元素。它将充当 web 浏览器的服务器和端服务器的客户端。因此，您将获得使用客户端和服务器编程的经验。

日志记录

您的代理应该跟踪名为 proxy.log 的日志文件中的所有请求。每个日志文件条目都应为以下形式的文件：

日期：browserIPURL 大小

其中 browserIP 是浏览器的 IP 地址，URL 是请求的 URL，大小是返回的对象的字节大小。例如：

2002 年 10 月 27 日星期日 02：51：02 美国东部时间：128.2.111.38http://www.cs.cmu.edu/34314

请注意，大小本质上是从打开连接到关闭时从终端服务器接收到的字节数。仅应记录来自最终服务器的响应所满足的请求。我们在 csapp.c 中提供了函数 format_log_entry，以创建所需格式的日志条目。

端口号

代理应该侦听其在命令行上传入的端口号上的连接请求：

unix>。/代理服务器 15213

您可以使用任何端口号 p，其中 1024<p<65536，以及 p 当前没有被任何其他系统或用户服务（包括其他学生的代理）使用。有关其他系统服务保留的端口号的列表，请参见/etc/服务。

第二部分：同时处理多个请求

实际代理不按顺序处理请求。它们会同时处理多个请求。一旦您有了一个工作的顺序日志记录代理，您应该更改它以同时处理多个请求。最简单的方法是创建一个新线程来处理到达的每个新连接请求(CSAPP1338)。

使用这种方法，多个对等线程有可能同时访问日志文件。因此，您需要使用信号量来同步对文件的访问，以

便一次只能有一个同级线程修改它。如果不同步这些线程，则日志文件可能已损坏。例如，文件中的一行可能从另一行的中间开始。

第三部分：正在缓存 web 对象

对于实验室的最后一部分，您将向代理添加一个缓存，以存储最近使用的 Web 对象在内存中。HTTP 实际上定义了一个相当复杂的模型，通过这个模型，web 服务器可以说明其服务的对象应该如何缓存，客户端可以指定如何代表他们使用缓存。但是，您的代理将采用一种简化的方法。

当代理从服务器接收到 web 对象时，它应该在将对象传输到客户端时将其缓存到内存中。如果另一个客户端从同一服务器请求相同对象，代理无需重新连接到服务器；它可以简单重新发送缓存的对象。

显然，如果您的代理要缓存曾经请求的每个对象，它将需要无限量的内存。此外，由于一些 web 对象比其他对象要大，因此一个巨大的对象可能会消耗整个缓存，从而完全阻止了其他对象被缓存。为了避免这些问题，您的代理应该具有最大缓存大小和最大缓存对象大小。

最大高速缓存大小

代理的整个缓存应具有以下最大大小：

```
MAX_CACHE_SIZE=1MiB
```

在计算其缓存的大小时，代理只能计算用于存储实际 web 对象的字节；应忽略任何无关的字节，包括元数据。

最大对象大小

代理应只缓存不超过以下最大大小的 web 对象：

```
MAX_OBJECT_SIZE=100 基比
```

为了方便，这两个大小限制都作为 proxy.c 中作为宏提供。

实现正确缓存的最简单方法是为每个活动连接分配一个缓冲区，并在从服务器接收数据时积累数据。如果缓冲区的大小曾经超过最大对象大小，则可以丢弃该缓冲区。如果在超过最大对象大小之前读取了 web 服务器的全部响应，则可以缓存该对象。使用此方案，代理将用于 web 对象的最大数据量如下，其中 T 是活动连接的最大数量：

```
MAX_CACHE_SIZE+t*MAX_OBJECT_SIZE
```

驱逐出境的政策

代理的缓存应使用接近最近使用的(LRU)驱逐策略的驱逐策略。它不需要严格 LRU，但它应该是相当接近。注意，读取对象和写入对象都是使用该对象。

“同步处理

访问缓存必须是线程安全的，并且确保缓存访问没有竞赛条件可能是实验室这部分中更有趣的方面。事实上，有一个特殊的要求是，多个线程必须能够同时从缓存中读取。当然，一次只允许一个线程写入缓存，但读卡器不存在这种限制。

因此，使用一个大型独占锁来保护对缓存的访问是不可接受的解决方案。您可能需要探索一些选项，如分区缓存、使用 Pthreades 读写器锁，或使用信号量来实现您自己的读写器解决方案。在任何一种情况下，你都不需要执行严格的 LRU 驱逐政策，这将给你在支持多个读者方面的一些灵活性。

评价

该作业共分为 70 分：

基本的代理功能（30 分）。顺序代理应正确接受连接，将请求转发到最终服务器，并将响应传递回浏览器，为每个请求创建一个日志项。您的程序应该能够将浏览器请求代理到以下网站，并正确记录这些请求：

一 http://www.baidu.com

—http://www.sohu.com

—http://www.qq.com

- 处理并发的请求（20 分）。

  您的代理应该能够处理多个并发连接。我们将使用以下测试来确定这一点：（1）使用 telnet 打开与代理的连接，然后不输入任何数据而不打开它。（2）使用 Web 浏览器（指向代理）从某些端服务器请求内容。

  此外，您的代理应该是线程安全的，保护日志文件的所有更新，并保护对任何线程不安全函数的调用，如代代 stbyaddr。

- 提供工作高速缓存（10 分）。

- 样式（10 分）。可读性和评论良好的代码最多可获得 10 分。代码应该以注释块开头，一般描述代理的工作方式。此外，每个函数都应该有一个注释块来描述该函数的作用。此外，线程应该分离地运行，并且代码不应该有任何内存泄漏。

提示信息

- 使用代理的最好方法是从基本的 echo 服务器(CS：APP12.4.9)开始，然后逐渐添加将服务器变成代理的功能。

- 最初，您应该使用 telnet 作为客户端来调试代理(CS：APP12.5.3)。

- 稍后，使用真正的浏览器测试代理。浏览浏览器设置，直到找到“代理”，然后输入运行的主机和端口。对于网景，请选择编辑、首选项、高级、代理、手动代理配置。在因特网浏览器中，依次选择工具、选项、连接，然后选择局域网设置。选中“使用代理服务器”，然后单击“高级”。只需设置 HTTP 代理，因为这是您将能够处理的所有代码。

- 由于我们希望您专注于这个实验室的网络编程问题，我们为您提供了两个辅助例程：parse_uri，它从maURI 中提取主机名、路径和端口组件，以及格式_log_entry，它以正确的格式为日志文件构建一个条目。
- 请小心内存泄漏。当 HTTP 请求的处理因任何原因失败时，线程必须关闭所有打开的套接字描述符，并在终止之前释放所有内存资源。
- 您会发现为每个线程分配一个小的唯一整数 ID（如当前请求号），然后将此 ID 作为参数之一传递给线程例程将非常有用。如果在每个调试输出语句中显示此 ID，则可以准确地跟踪每个线程的活动。
- 为了避免潜在的致命内存泄漏，线程应以分离、不可连接运行（CS：APP13.3.6）。
- 由于日志文件正由多个线程写入，因此每次写入时都必须使用互斥符号来保护它（CS：APP13.5.2 和 13.5.3）。
- 在线程内调用线程不安全函数，如 inet_ntoa、函名和函数。特别是，csapp.c 中的 open_clientfd 函数是线程不安全的，因为它调用了 open_clientfdaddr，一个类 3 线程不安全函数(CSAPP13.7.1)。您将需要编写一个线程安全版本的 open_clientfd，称为 open_clientfd_ts，它在调用 open_clientfd 时使用锁定和复制技术(CS：APP13.7.1)。
- 对于插座上的所有 I/O，请使用里约热内卢（强健的 I/O）包（CS：APP11.4）。不要在插座上使用标准的输入输出。如果你这样做了，你很快就会遇到问题。但是，标准的 I/O 调用，如 fopen 和 fwrite，对于日志文件上的 I/O 很适用。
- csapp.c 中的 Rio_readn、Rio_readlineb 和 Rio_writen 错误检查包装器不适合于实际的代理，因为它们在遇到错误时终止进程。相反，您应该编写名为 Rio_readn_w、Rio_readlineb_w 和 Rio_writen_w 的新包装器，以便在 I/O 失败时打印完警告消息后即可返回。当任何一个读取包装器检测到错误时，它应该返回 0，就好像它在套接字上遇到了 EOF 一样。
- 读取数和写数可能会因为各种原因而失败。最常见的读取故障是从另一端的对等连接读取导致的 =ECONNRESET 错误，通常是过载的端服务器。最常见的写入失败是由于写入其另一端的同行已关闭的连接而导致的 errno=EPIPE 错误。例如，当用户在长时间传输过程中点击浏览器的停止按钮时，可能会发生这种情况。
- 写入对等机首次关闭的连接会导致错误，errno 设置为 EPIPE。第二次写入这样的连接会引出一个 SIGPIPE 信号，其默认操作是终止该进程。为了防止您的代理崩溃，您可以使用它的 SIG“GN 参数的信号函数(CS：APP8.5.3)显式地忽略这些 SIGPIPE 信号