20XX 年秋季 15 月 213 日

代理实验室：编写缓存 Web 代理

分配：11 月 19 日星期四

截止日期：12 月 8 日，星期二，晚上 11：59

最后可能的时间：12 月 11 日星期五，晚上 11：59

1 介绍

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最终服务器以获取网页，而是联系代理，该代理将请求转发到最终服务器。当最终服务器响应代理时，代理会将回复发送给浏览器。

代理对许多用途都很有用。有时代理被用在防火墙中，因此防火墙后面的浏览器只能通过代理联系防火墙以外的服务器。代理还可以充当匿名器：通过剥离所有识别信息的请求，代理可以使浏览器对 Web 服务器进行匿名。代理甚至可以用来缓存 web 对象，通过存储服务器上对象的本地副本，然后通过从缓存中读取而不是与远程服务器再次通信来响应未来的请求。

在这个实验室中，您将编写一个缓存 Web 对象的简单 HTTP 代理。在实验室的第一部分，您将设置代理来接收传入的连接、读取和解析请求、将请求转发给 web 服务器、读取服务器的响应，并将这些响应转发给相应的客户端。这第一部分将涉及学习基本的 HTTP 操作，以及如何使用套接字编写通过网络连接进行通信的程序。在第二部分中，您将升级代理以处理多个并发连接。这将向您介绍处理并发性，这是一个重要的系统概念。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到代理添加缓存。

2 物流

这是一个单独的项目。

3 传出说明

特定地点：在这里插入一段，解释教师将如何向学生分发 proxylab-handout.tar 文件。

将讲义文件复制到您计划执行工作的 Linux 机器上的受保护目录中，然后发出以下命令：

```
linux>焦油 xvfproxylab-handout.tar
```

这将生成一个名为代理实验室讲义的讲义目录。自述文件描述了各种文件。

4 第一部分：实现一个连续的 Web 代理

第一步是实现一个处理 HTTP/1.0GET 请求的基本顺序代理。其他请求类型，如 POST，都是严格可选的。

启动时，您的代理应该侦听将在命令行上指定编号的端口上的传入连接。一旦建立了连接，您的代理应该从客户端读取整个请求并解析请求。它应该确定客户端是否发送了有效的 HTTP 请求；如果是，它可以建立自己到适当的 web 服务器的连接，然后请求客户端指定的对象。最后，代理应该读取服务器的响应并将其转发给客户端。

4.1 HTTP/1.0GET 请求

当最终用户在 web 浏览器的地址栏中输入 http://www.cmu.edu/hub/index.html 等 URL 时，浏览器将向代理发送一个 HTTP 请求，该请求可能以类似以下内容的行开头：

获得 http://www.cmu.edu/hub/index.htmlHTTP/1.1

在这种情况下，代理应该将请求解析为以下字段：主机名 www.cmu.edu；以及路径或查询及其之后的所有内容 /hub/index.html。这样，代理就可以确定它应该打开到 www.cmu.edu 的连接，并以以下表单开头发送它自己的 HTTP 请求：

GET/hub/index.htmlHTTP/1.0

注意，HTTP 请求中的所有行都以车厢返回“\r”，后面是换行“、\n”结束。同样重要的是，每个 HTTP 请求都由一个空行：“\r\n”终止。

在上面的示例中，您应该注意到 web 浏览器的请求行以 HTTP/1.1 结束，而代理的请求行以 HTTP/1.0 结束。现代的 web 浏览器将生成 HTTP/1.1 请求，但是您的代理应该处理它们，并将它们作为 HTTP/1.0 请求转发。

重要的是要考虑到 HTTP 请求，即使只是 HTTP/1.0GET 请求的子集，也可能是非常复杂的。该教科书描述了 HTTP 事务的某些细节，但您应该参考 RFC1945 以获得完整的 HTTP/1.0 规范。理想情况下，您的 HTTP 请求解析器将根据 RFC1945 的相关部分进行完全健壮，除了一个细节：虽然规范允许多行请求字段，但您的代理不需要正确处理这些字段。当然，您的代理不应该因为格式错误的请求而提前中止。

4.2 请求标题

此实验室的重要请求头是主机、用户代理、连接和代理连接头：

- 始终发送一个主机标头。虽然这种行为在技术上不受 HTTP/1.0 规范的批准，但有必要从某些 Web 服务器，特别是那些使用虚拟主机的服务器。

  主机头描述了终端服务器的主机名。例如，要访问 http://www。在使用 cmu.edu/hub/index.html 时，您的代

理将发送以下标题：

主机：www.cmu.edu

web 浏览器可能会将它们自己的主机头附加到它们的 HTTP 请求中。如果是这样，您的代理应该使用与浏览器相同的主机头。

- 您可以选择始终发送以下用户代理标头：

  用户代理：Mozilla/5.0(X11；Linuxx86_64；rv：10.0.3)Gecko/20120305Firefox/10.0.3

  头提供在两行上，因为它不适合在编写中作为一行提供，但是您的代理应该将头作为单行发送。
  用户代理头标识客户端（根据操作系统和浏览器等参数），web 服务器通常使用标识信息来操作它们所服务的内容。发送这个特定的用户代理：字符串可能会改善内容和多样性，您在简单的 telnet 风格的测试中返回的材料。

- 始终发送以下连接头：

  连接：关闭

- 始终发送以下代理连接头：

  代理连接：关闭

  连接和代理连接头用于指定在第一次请求/响应交换完成后，连接是否将保持激活。让您的代理为每个请求打开一个新的连接。指定关闭作为这些报头的值，将提醒 web 服务器，您的代理打算在第一次请求/响应交换后关闭连接。

为方便起见，所描述的用户代理头的值在 proxy.c 中作为字符串常数提供给您。

最后，如果浏览器作为 HTTP 请求的一部分发送任何额外的请求头，那么您的代理应该不变地转发它们。

4.3 端口号

这个实验室有两种重要的端口号类型：HTTP 请求端口和代理的侦听端口。

HTTP 请求端口是 HTTP 请求的 URL 中的一个可选字段。也就是说，URL 可能是 http://www.cmu.edu：8080/hub/index.html，在这种情况下，您的代理应该连接到端口 8080 上的主机 www.cmu.edu，而不是默认的 HTTP 端口，这是端口 80。无论端口号是否包含在 URL 中，代理都必须正常工作。

侦听端口是代理应该在上面侦听传入连接的端口。代理应该接受一个指定代理的监听端口号的命令行参数。例如，使用以下命令，代理应该侦听端口 15213 上的连接：

linux>./proxy15213

您可以选择任何非特权监听端口（大于 1024 和小于 65536），只要它不被其他进程使用。由于每个代理必须使用一个唯一的监听端口，并且许多人将同时在每台机器上工作，因此提供了脚本 port-for-user.pl 来帮助您选择自己的个人端口号。使用它可以根据用户 ID 生成端口号：

```
linux>。/port-for-user.pldroh 机器人：45806
```

port-for-user.pl 返回的端口 p 总是偶数。因此，如果您需要一个额外的端口号，比如对于小服务器，您可以安全地使用端口 p 和 p+1。

请不要选择您自己的随机端口。如果你这样做了，你就会有干扰其他用户的风险。

5 第二部分：处理多个并发请求

一旦您有了一个工作的顺序代理，您就应该修改它以同时处理多个请求。实现并发服务器的最简单的方法是生成一个新的线程来处理每个新的连接请求。其他设计也是可能的，如教科书第 12.5.5 节中描述的预线程服务器。

- 请注意，线程应该以分离的模式运行，以避免内存泄漏。
- CS：APP3e 教科书中描述的ˆ客户端和ˆ监听函数是基于现代和独立于协议的获取信息函数，因此是线程安全的。

6 第三部分：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，它将最近使用的 Web 对象存储在内存中。HTTP 实际上定义了一个相当复杂的模型，通过该模型，web 服务器可以提供关于它们所服务的对象应该如何缓存的说明，并且客户端可以指定应该如何代表它们使用缓存。但是，您的代理将采用一种简化的方法。

当您的代理从服务器接收到一个 web 对象时，它应该在将该对象传输到客户端时将其缓存到内存中。如果另一客户端从同一台服务器请求相同的对象，则代理不需要重新连接到服务器；它可以简单地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，它将需要无限数量的内存。此外，由于一些 web 对象比其他对象更大，可能一个巨大的对象将消耗整个缓存，从而阻止其他对象被缓存。为了避免这些问题，您的代理应该同时具有最大缓存大小和最大缓存对象大小。

6.1 最大缓存大小

代理的整个缓存应该具有以下最大大小：

```
MAX_CACHE_SIZE=1 混合物
```

在计算其缓存的大小时，代理必须只计算用于存储实际 Web 对象的字节；应忽略任何无关的字节，包括元数据。

6.2 最大对象大小

您的代理应该只缓存不超过以下最大大小的 Web 对象：

```
MAX_OBJECT_SIZE=100KiB
```

为了方便起见，这两个大小限制都是在 proxy.c 中作为宏提供的。

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，并在从服务器接收到数据时积累数据。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据数：

```
max_cache_size+t*max_object_size
```

6.3 规避政策

代理的缓存应该使用近似于最近使用最少的(LRU)驱逐策略的驱逐策略。它不需要是严格的 LRU，但它应该是一些相当接近的东西。请注意，读取一个对象和写入它都等于使用该对象。

6.4 同步化

对缓存的访问必须是线程安全的，并且确保缓存访问不受竞赛条件的影响，这可能是实验室这一部分更有趣的方面。事实上，有一个特殊的要求，即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，但对于阅读器不存在该限制。

因此，使用一个大型的独占锁保护缓存访问是一个不可接受的解决方案。您可能希望探索一些选项，如分区缓存、使用 Pshoine 阅读器-编写器锁，或使用信号量来实现您自己的阅读器-编写器解决方案。在任何一种情况下，您都不需要实施严格的 LRU 驱逐政策，这将为您在支持多个读者方面提供一些灵活性。

7 评价

该作业共分为 70 分：

- 基本正确度：基本代理操作 40 分（自动分级）
- 并发性：处理并发请求的 15 分（自动分级）
- 缓存：工作缓存的 15 分（自动分级）

7.1Autograding

您的讲义材料包括一个名为 driver.sh 的自动评分器，您的讲师将使用它来分配基本正确性、并发性和缓存的分数。

linux>./driver.sh

您必须在 Linux 机器上运行该驱动程序。

7.2 稳健性

像往常一样，您必须提供一个对错误甚至畸形或恶意输入具有健壮的程序。服务器通常是长期运行的进程，web 代理也不例外。仔细考虑长期运行的进程应该如何对不同类型的错误做出什么反应。对于许多类型的错误，您的代理当然不适合立即退出。

鲁棒性也意味着其他需求，包括对错误情况的入侵，如分割错误和缺乏内存泄漏和文件描述符泄漏。

8 测试和调试

除了简单的自动分级器，您将没有任何示例输入或测试程序来测试您的实现。您必须想出您自己的测试，甚至可能是您自己的测试工具来帮助您调试代码，并决定何时有一个正确的实现。这在现实世界中是一项有价值的技能，在那里，确切的操作条件很少被知道，参考解决方案往往不可用。

幸运的是，您可以使用许多工具来调试和测试代理。确保执行所有代码路径并测试一组代表性输入，包括基本例、典型例和边缘例。

8.1 小网络服务器

您的讲义目录中的 CS：APP 小 web 服务器的源代码。虽然没有 thttpd 那么强大，CS：APP 小 web 服务器将很容易修改，因为你认为合适的。这也是代理代码的一个合理的起点。它是驱动程序代码用来获取页面的服务器。

8.2 端粒

如教科书（11.5.3）所述，您可以使用 telnet 打开到代理的连接，并向其发送 HTTP 请求。

8.3 一绺鬈发

您可以使用 curl 向任何服务器生成 HTTP 请求，包括您自己的代理。这是一个非常有用的调试工具。例如，如果您的代理和 Tiny 都在本地计算机上运行，则 Tiny 侦听端口 15213，而代理侦听端口 15214，那么您可以通过以下 curl 命令通过代理从 Tiny 请求页面：

linux>curl-v 一一 proxyhttp://localhost：15214http://localhost：15213/home.html*即将连接()到代理本地主机端口 15214（#0）
尝试 127.0.0.1…连接

*已连接到本地主机（127.0.0.1）端口 15214（#0）
> 获得 http://localhost：15213/home.htmlHTTP/1.1
> 用户代理：curl/7.19.7(x86_64-redhat-linux-gnu)。
> 主机：本地主机：15213

```
>  接受：*/*
>  代理连接：保持实时
>    *HTTP1.0，假设在车身结束后关闭
<  http/1.0 200 好
<  服务器：TinyWeb 服务器
<  内容长度：120
<  内容类型：文本/html
<    <html>
<head><title>test</title></head><body>
<img align=“middle” src=“godzilla.gif”>
戴夫·奥哈拉伦
</body>
</html>
*关闭连接#0
```

8.4netcat

netcat，也被称为 nc，是一个多功能的网络实用工具。您可以像 telcet 一样使用 netcat 来打开到服务器的连接。因此，想象你的代理使用端口 12345 运行在猫鲨上，你可以做下面的事情来手动测试你的代理：

```
sh>nc catshark.ics.cs.cmu.edu12345
获得 http://www.cmu.edu/hub/index.htmlHTTP/1.0

http/1.1 200 好
```

除了能够连接到 Web 服务器外，Netcat 还可以作为服务器本身进行操作。通过以下命令，您可以在 netcat 作为服务器侦听端口 12345 上运行：

```
sh>nc-l12345
```

一旦您设置了一个 Netcat 服务器，您就可以通过代理生成对其中一个假对象的请求，并且您将能够检查您的代理发送到 Netcat 的确切请求。

8.5 Web 浏览器

最终，你应该使用最新版本的火狐来测试你的代理。访问关于火狐将自动更新您的浏览器到最新版本。
要将火狐配置为与代理合作，请访问

```
首选项>高级>网络>设置
```

看到你的代理通过一个真正的 Web 浏览器工作将是非常令人兴奋的。虽然您的代理的功能将会很有限，但您会注意到，您可以通过您的代理浏览绝大多数网站。
一个重要的警告是，在使用 Web 浏览器测试缓存时必须非常小心。所有现代的 Web 浏览器都有自己的缓存，在尝试测试代理的缓存之前，您应该禁用这些缓存。

9 处理程序说明

提供的生成文件包括构建最终处理文件的功能。从工作目录中发出以下命令：

```
Linux>制作手机
```

输出是文件../proxylab-handin.tar，然后你可以上交它。

特定于网站：在这里插入一段，告诉每个学生如何提交他们的 proxylab-handin.tar 解决方案文件。

- 该教科书的第 10-12 章包含了关于系统级 I/O、网络编程、HTTP 协议和并发编程的有用信息。
- RFC1945(http://www.ietf.org/rfc/rfc1945.txt)是 HTTP/1.0 协议的完整规范。

10 铰链

- 正如教科书第 10.11 节所述，对套接字输入和输出使用标准的输入/输出函数是一个问题。相反，我们建议您使用强大的 I/O（里约热内卢）包，它在讲义目录中的 csapp.c 文件中提供。
- csapp.c 中提供的错误处理功能不适用于您的代理，因为一旦服务器开始接受连接，它就不应该终止。你需要修改它们或写你自己的内容。
- 您可以随意以任何您喜欢的方式修改讲义目录中的文件。例如，为了良好的模块化，您可以将缓存函数实现为一个称为 cache.c 和缓存。h 的文件库。当然，添加新文件将需要您更新所提供的生成文件。
- 正如在 CS:APP3e 文本的第 964 页之外所讨论的,您的代理必须忽略签名管信号,并且应该优雅地处理返回 EPIPE 错误的写操作。
- 有时，调用读取从过早关闭的套接字接收字节将导致读取返回-1，errno 设置为重新重置。您的代理也不应因为此错误而终止。
- 请记住，并不是网络上的所有内容都是 ASCII 文本。网络上的大部分内容都是二进制数据，比如图像和视频。在选择和使用网络 I/O 函数时，确保考虑二进制数据。
- 将所有请求转发为 HTTP/1.0，即使原始请求为 HTTP/1.1。

祝你好运！