练习 14

客户端

对于本练习,您将编写一个名为 HTTPclient.c 的简单程序。它将能够使用 TCP 套接字发出（未加密的)HTTP 请求以获取网页或对 Internet 上的服务器进行 REST 调用。 REST 是一种使用我们用来检索网页的相同协议进行远程过程调用（如 RPC)的技术。

HTTP 是应用层协议的一个很好的例子。它是在进程内部实现的（即,不是由操作系统实现的）。它采用传输层协议（例如 TCP)提供的语义并添加额外的代码、控制信息和策略来创建适合完成特定任务的东西。

我们将在我们的实施中使用未加密的 HTTP。这使得编写我们自己的代码来创建一个简单的请求、将其发送到服务器并读回响应变得容易。实际上,使用加密的 HTTPS 更为常见。
为 HTTPS 编写我们自己的代码将是一项更大的工作,对于一些小的编程练习来说太多了。这意味着我们的客户端将只能与仍然提供未加密 HTTP 的服务器通信。本练习提供了一个使用 HTTP 的测试服务器,Internet 上仍然有一些这样的服务器。

我们的 HTTPclient.c 程序需要三个命令行参数。

- 第一个参数是我们将向其发送请求的主机的名称。当你
  在开发您的解决方案时,您可能会使用 localhost 连接到在同一系统上运行的测试服务器。
  一旦你的客户端开始工作,你应该能够连接到任何（未加密的)HTTP 服务器
  互联网。
- 第二个参数是服务器监听的端口号。在您进行开发时,您将运行一个测试服务器作为您在家庭作业 5 中使用的相同的、随机分配的端口号。因此,您需要为客户端提供相同的端口号。一旦您的客户端开始工作,您就可以连接到 Internet 上的任何（未加密的)HTTP 服务器,因此您需要使用该服务器运行的任何端口号（可能是 80）。 · 第三个命令行参数将是我们请求的URL 路径。它让我们访问不同的资源

  通过提供不同的路径在主机上。

这三个字段通常会在 URL 中打包在一起,如下所示。

http://主机名:端口/路径

将这三个字段分解为单独的命令行参数将稍微简化我们的客户端程序。它不必担心从单个 URL 字符串中解析这三个字段。

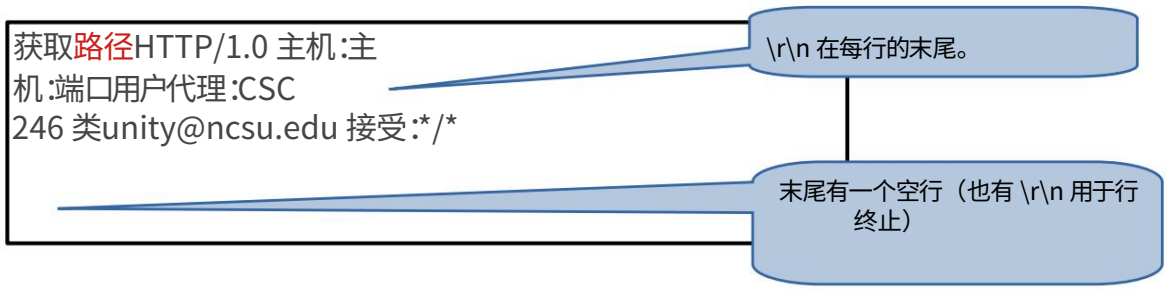
$ ./HTTPclient主机名 端口路径

发出 HTTP 请求

起始文件 HTTPclient.c 已经包含用于与给定主机和端口号建立 TCP 连接的代码。您需要编写代码来构建 HTTP 请求并将其发送到服务器。未加密的 HTTP 请求表示为文本。它可以有一个标题和一个正文,但我们只需要为我们的客户提供标题部分。正文将是空的,因此您的客户端可以只将标头发送到服务器,后面什么也没有。

对于标头,您需要构建一个包含以下文本的字符串。红色斜体文本表示您需要修改的标题部分。路径、主机和端口是命令行参数的值。您可以将这些参数的文本复制到标头的这些部分。对于 User-Agent 行,我们将使用一个字符串来指示我们正在为一个类开发客户端,并且我们将包括开发人员（您)的联系信息以防万一

错误的。用您自己的统一 ID替换标头中的统一。



HTTP 标头应在每行末尾具有 Windows 样式的行终止符。每行都应以回车换行序列 (\r\n) 结尾。您还需要在标题末尾有一个空行（也以回车符、换行符结尾）。空行表示标题的结尾。

您可以将此标头准备为字符串，然后通过将字符串的内容写入套接字来将其发送到服务器。标准库函数 snprintf() 在将标头准备为字符串时很有用，但您可以根据需要创建它。

处理 HTTP 响应

对 HTTP 请求的响应将在开头包含一个标头，然后是正文。就像在请求中一样，标题将是以回车换行符结尾的文本行，空行标记标题的结尾。主体将是在空行末尾的行终止之后收到的任何内容。

一旦完成发送请求，您的客户端就可以从套接字连接中读取响应。为此使用 read() 系统调用。响应可能无法在对 read() 的一次调用中全部可用，因此您的客户端应继续从套接字读取连续的数据块，直到它从 read() 返回零（表示文件结束）或返回的 -1（表示错误）。

当您的客户端读取服务器的响应时，它应该将其打印到终端。将标头中的字节打印到标准错误，并将正文中的字节打印到标准输出。这意味着您的客户端将需要代码来检测标头末尾的空行。那时您需要从打印到标准错误切换到打印到标准输出。

请记住，标题行（包括末尾的空白行）以回车符、换行符 (\r\c) 终止。

测试你的客户端

您应该能够使用以下编译命令构建您的客户端。 gnu99 标准（而不是 c99）应该允许您使用 getaddrinfo() 函数来进行名称解析。

```
gcc -Wall -std=gnu99 -g HTTPclient.c -o HTTPclient
```

我们提供了一个 Java 测试服务器来帮助您在开发过程中测试您的客户端。它称为 TestServer.java。您应该能够像典型的 Java 程序一样使用 javac 编译它：

```
javac 测试服务器.java
```

如果您在大学机器上开发您的客户端，其他学生可能会同时使用相同的系统。因此，当您运行测试服务器时，我们需要确保您不会尝试使用与其他学生相同的端口号。我们将使用与任务 5 相同的随机分配端口号。如果您单击“端口号分配”分配，您可以下载一个反馈文件，告诉您应该使用哪个端口号。请务必使用

用于您的测试服务器,这样您就不会干扰可能在同一系统上开发的其他学生。

当您运行测试服务器时,它需要端口号作为命令行参数。在下面的示例执行中,我正在运行端口号为 99999 的测试服务器。它比任何合法的 16 位端口号都大,因此如果您尝试它将会失败。只需将此号码替换为您从“端口号分配”分配中分配的端口号。

Java 测试服务器 99999

测试服务器将继续运行,直到您使用 ctrl-C 将其终止。对于每个请求,它都会发回一个回复,其中包含请求中的路径、请求的来源地址以及到目前为止已处理的请求总数。

在同一主机上的另一个终端中,您可以使用如下命令来运行您的客户端程序并告诉它连接到您的测试服务器。将 99999 替换为您的服务器正在使用的端口号。此外,请确保您的客户端和服务器在同一台主机上运行,否则本地主机名将无法连接到您的服务器。

下面显示了您应该期望的输出,具体取决于您实际发出请求的时间和您之前可能发出的请求的数量。

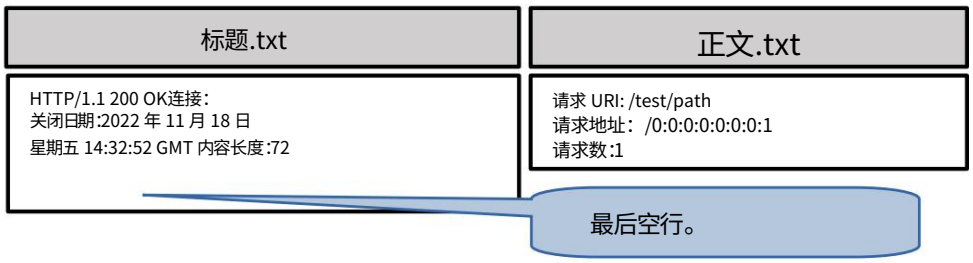
$ ./HTTPclient localhost 99999 /test/path HTTP/1.1 200 OK 连接:关闭
日期:2022 年 11 月 18 日星
期五 14:32:52 GMT 内容长度:
72

请求 URI: /test/path
请求地址：/0:0:0:0:0:0:0:1
请求数:1

您可以按如下方式运行您的客户端,以捕获对不同文件的响应的标头和主体。这将帮助您检查以确保服务器响应的正确部分是标准错误而不是标准输出。

./HTTPclient localhost 99999 /test/path 2> header.txt > body.txt

如果您像这样运行您的客户端,那么您应该在每个文件中看到以下内容,具体取决于请求的时间和您的服务器处理的先前请求。 header 信息到 header.txt 文件,body 到 body.txt。



有趣的 HTTP 客户端

一旦您的 HTTP 客户端开始工作,您应该能够使用它来连接到仍然提供未加密的 HTTP 连接的 Web 服务器或 REST 服务。这些不像以前那么普遍,但是,在编写此作业时（2022 年 11 月）,您应该能够尝试以下内容。

对于以下每个 URL,您需要将其分解为单独的字段以向您的客户提出请求。 HTTP 服务的默认端口号是 80,因此您可以将其用于这些示例中的任何一个。

- 无聊的 API

  这是一个 REST 接口,用于在您感到无聊时获取建议的活动。下面将建议一个随机活动。 http://www.boredapi.com/api/activity

- 数字 API

  这是一个用于获取数字信息的 REST 接口。以下内容将为您提供有关随机选择的数字的琐事事实。 http://numbersapi.com/random/trivia

- 借口 API

  这是一个 REST 接口,用于获取可以用来摆脱某些事情的随机借口。 http://excuser.herokuapp.com/v1/excuse

提交您的解决方案

当您的代码运行时,将完成的客户端 HTTPclient.c 提交到名为 EX14 的 Gradescope 作业。为了对作业评分,我们将在测试服务器上运行您的客户端,就像作业提供的那样,尽管它可能
不在客户端的同一主机上运行。