

实验室检查点 0：网络热身

到期的：9 月 28 日星期二晚上 8:30

最迟截止日期：9 月 30 日晚上 11:59（接收反馈的最后一天；风格评分上限为 2/3） 实验

课：9 月 21 日和 28 日，下午 5:30–8:30

欢迎来到 CS144：计算机网络简介。在此热身中，您将在计算机上安装 Linux，学习如何手动通过 Internet 执行某些任务，用 C++ 编写一个通过 Internet 获取网页的小程序，并实现（在内存中）网络的关键抽象之一：写入器和读取器之间的可靠字节流。我们预计此热身将花费您 2 到 6 小时完成（未来的实验将占用您更多的时间）。关于实验任务的三个要点：

- 在深入研究之前最好先阅读整个文档！
- 在这个由 8 个部分组成的实验作业中，您将构建自己对 Internet 很大一部分的实现——路由器、网络接口和 TCP 协议（将不可靠的数据报转换为可靠的字节流）。*大多数时候你都会在之前的工作基础上继续努力*，也就是说，您将在本季度内逐步建立自己的实施，并将继续在未来几周内使用您的工作。这使得“跳过”检查点变得困难。
- 实验室文档不是“规范”——这意味着它们不是单向使用的。它们的编写更接近软件工程师从老板或客户那里获得的详细程度。我们希望您能从参加实验室课程中受益，如果您发现某些内容含糊不清并且您认为答案很重要，请提出澄清问题。（如果您认为某些内容可能没有完全说明，有时事实是这并不重要——您可以尝试一种方法或另一种方法，看看会发生什么。）

0 合作政策

编程作业必须是你自己的作品：您必须编写您提交的编程作业的所有代码，除了我们作为作业的一部分提供给您的代码。请不要从 Stack Overflow、GitHub 或其他来源复制粘贴代码。如果您根据在 Web 或其他地方找到的示例编写自己的代码，请在您提交的源代码的注释中引用 URL。

与他人合作：您不得向其他人展示您的代码，不得查看其他人的代码，也不得查看前几年的解决方案。您可以与其他学生讨论作业，但不得抄袭任何人的代码。如果您与其他学生讨论作业，请在您提交的源代码的评论中注明他们的名字。请参阅课程管理讲义了解更多详细信息，如果有任何不清楚的地方，请在 Piazza 上询问。

广场：请随意在 Piazza 上提问，但请不要发布任何源代码。

1 在你的计算机上设置 GNU/Linux

CS144 的作业需要 GNU/Linux 操作系统和支持 C++ 2017 标准的最新 C++ 编译器。请选择以下三个选项之一：

1. 受到推崇的：安装 CS144 VirtualBox 虚拟机映像（说明位于 <https://stanford.edu/class/cs144/vm-howto/vm-howto-image.html>）。
2. 运行 Ubuntu 18.04 LTS 版本，然后运行我们的安装脚本。您可以在实际计算机上、VirtualBox 中、EC2 或其他虚拟机上执行此操作（分步指南位于 <https://stanford.edu/class/cs144/vm-howto/vm-howto-iso.html>）。
3. 使用其他 GNU/Linux 发行版，但请注意，您可能在此过程中遇到障碍，并且需要熟悉如何调试它们。您的代码将在 Ubuntu 18.04 LTS 上进行测试，克++8.2.0 并且必须在这些条件下正确编译和运行。提示<https://stanford.edu/class/cs144/vm-howto/vm-howto-byo.html>。
4. 如果您有 2020-21 MacBook（配备 ARM64 M1 芯片），VirtualBox 将无法成功运行。请从以下位置安装 UTM 虚拟机软件 and 我们的 ARM64 虚拟机映像[https://stanford.edu/class/cs144/vm 操作方法/](https://stanford.edu/class/cs144/vm-操作方法/)。

2 手动联网

让我们开始使用网络。您将手动执行两项任务：检索网页（就像 Web 浏览器一样）和发送电子邮件消息（就像电子邮件客户端一样）。这两项任务都依赖于称为 *可靠的双向字节流*：您将在终端中输入一系列字节，最终将以相同的顺序将相同的字节序列传送到另一台计算机（服务器）上运行的程序。服务器将以其自己的字节序列进行响应，并将其传送回您的终端。

2.1 获取网页

1. 在 Web 浏览器中，访问<http://cs144.keithw.org/hello>并观察结果。
2. 现在，您将手动执行与浏览器相同的操作。

（一个）在您的虚拟机上，跑步 `telnet cs144.keithw.org http`。这告诉远程登录在你的计算机和另一台计算机之间打开可靠字节流的程序（名为 `cs144.keithw.org`）、并且特别 *服务* 在该计算机上运行：“http” 服务，用于万维网使用的超文本传输 协议。¹

如果您的计算机已正确设置并已连接到互联网，您将看到：

¹计算机名称具有数字等价物（104.196.238.229，一个 *Internet 协议 v4 地址*），服务名称也是如此（80，一个 *TCP 端口号*）我们稍后会详细讨论这些内容。

用户@电脑:~\$telnet cs144.keithw.org http

尝试 104.196.238.229... 连接到
cs144.keithw.org。转义字符
为“^”。

如果需要退出，请按住 控制 并按]，然后输入 关闭 ↵。

(b) 类型 获取 /hello HTTP/1.1 ↵。这会告诉服务器小路URL的一部分。
(从第三个斜线开始的部分。)

(c) 类型 主办方: cs144.keithw.org ↵。这会告诉服务器主持人部分
URL。(http://和第三个斜线。)

(d) 类型 连接: 关闭 ↵。这告诉服务器您已完成
发出请求，并在回复完成后立即关闭连接。

(e) 再按一次 Enter 键：。这将发送一个空行并告诉服务器您已完成 HTTP 请求。

(f) 如果一切顺利，你将看到与浏览器相同的响应，前面是 HTTP 标题告诉浏览器如何解释响应。

3.任务：现在您已经知道如何手动获取网页，请向我们展示一下！使用上述技术获取 URL <http://cs144.keithw.org/lab0/> 苏内蒂德²，替换苏内蒂德使用您自己的主要 SUNet ID。您将在X-您的代码是：标题。保存您的 SUNet ID 和代码，以便将其包含在您的写作中。

2.2 给自己发一封电子邮件

现在您知道了如何获取网页，接下来是时候发送电子邮件了，同样使用可靠的字节流向另一台计算机上运行的服务。

1. 登录cardinal.stanford.edu (以确保你在斯坦福的网络上)，然后运行
telnet 148.163.153.234 smtp ↵。这²“SMTP”服务指的是简单邮件
传输协议，用于发送电子邮件。如果一切顺利，您将看到：

用户@电脑:~\$telnet 148.163.153.234 smtp 尝试

148.163.153.234...








已连接到 148.163.153.234。转义字
符为“^”。

220 mx0b-00000d03.pphosted.com ESMTP mfa-m0214089

2.第一步：识别你的电脑 电子邮件服务器。类型

HELO mycomputer.stanford.edu ↵ 等待看到类似“250...你好
cardinal3.stanford.edu [171.67.24.75]，很高兴认识你”。

²这些说明可能也适用于斯坦福网络之外，但我们不能保证。

3. 下一步：谁发了邮件？输入 `邮件发件人：苏内蒂德@斯坦福大学` 。
代替苏内蒂德使用您的 SUNet ID。³如果一切顺利，你会看到“250 2.1.0 发送者正常”。
4. 下一步：收件人是谁？首先，tr 向自己发送电子邮件。输入
`收件地址：苏内蒂德@斯坦福大学` 。代替苏内蒂德使用您自己的 SUNet ID。
如果一切顺利，你会看到“250 2.1.5 收件人同意。”
5. 现在该上传电子邮件本身了。输入 `数据`  告诉服务员你
准备开始。如果一切顺利，你会看到“354 以 <CR><LF>.<CR><LF> 结束数据”。
6. 现在，您正在给自己输入一封电子邮件。首先，输入标题您将在电子邮件客户端中看到
该邮件。在标题末尾留一个空行。
`354 以 <CR><LF>.<CR><LF> 结束数据 从：
苏内蒂德@stanford.edu 致：苏内蒂德
@stanford.edu 主题：来自 CS144 实验室 0
的问候！` 

7. 输入身体的电子邮件 留言——任何你喜欢的。完成后，以点结尾
单独占据一行：`.` 。预计会看到类似以下内容：“250 2.0.0 33h24dpdsr-1
消息已接受并传送”。
8. 类型 `辞职`  结束与电子邮件服务器的对话。检查您的收件箱并
垃圾邮件文件夹，以确保您收到了电子邮件。
9. 任务：现在您已经知道如何亲自发送电子邮件给自己，请尝试向朋友或实验室伙伴发送
一封电子邮件，并确保他们收到。最后，向我们展示您可以向我们发送一封电子邮件。
使用上述技巧，从您自己向 cs144grader@gmail.com。


2.3 聆听与联系

您已经了解了如何使用远程登录：一个客户与其他计算机上运行的程序建立传出连接的程序。现在是时候尝试成为一个简单的服务器：等待客户端连接的程序类型。

1. 在一个终端窗口中，运行 `netcat -v -l -p 9090` 在您的虚拟机上。您应该看到：

³是的，可以给出一个虚假的“发件人”地址。电子邮件有点像邮政服务的真实邮件，因为回信地址的准确性（大部分）取决于诚信制度。您可以在明信片上写任何您喜欢的回信地址，电子邮件也基本如此。请不要滥用这一点——说真的。工程知识意味着责任！垃圾邮件发送者和犯罪分子通常会使用虚假的“发件人”地址发送电子邮件，这样他们就可以假装是其他人。

用户@电脑:~\$netcat -v -l -p 9090 正在监听 [0.0.0.0]
(系列 0, 端口 9090)

2. 离开网络猫正在运行。在另一个终端窗口中，运行（也在您 telnet 本地主机 9090 的 VM 上）。
3. 如果一切顺利，网络猫将会打印类似“来自本地主机 53500 的连接已接收！”。
4. 现在尝试在任一终端窗口中输入 netcat（服务器）或远程登录（客户端）。请注意，您在一个窗口中输入的任何内容都会出现在另一个窗口中，反之亦然。您必须点击才能传输字节。
5. 在网络猫窗口，通过键入程序立即退出程序。控制-C 请注意远程登录

3 使用 OS 流套接字编写网络程序

在本热身实验的下一部分中，您将编写一个简短的程序，用于通过 Internet 获取网页。您将利用 Linux 内核和大多数其他操作系统提供的功能：创建 *可靠的双向字节流* 两个程序之间，一个在你的计算机上运行，另一个在互联网上的另一台计算机上运行（例如，Apache 或 nginx 等 Web 服务器，或网络猫程序）。

此功能称为 *流套接字* 对于你的程序和 Web 服务器来说，套接字看起来就像一个普通的文件描述符（类似于磁盘上的文件，或标准输入或者标准输出 I/O 流）。当两个流套接字 *已连接*，写入一个套接字的任何字节最终都会以相同的顺序从另一台计算机的另一个套接字出来。

但实际上，互联网并不提供可靠的字节流服务。相反，互联网真正做的唯一事情就是尽其所能地传递短数据片段，称为 *互联网数据报*，到达目的地。每个数据报都包含一些元数据（报头），用于指定源地址和目标地址（它来自哪台计算机，它要去往哪台计算机）以及一些 *有效载荷* 数据（最多约 1,500 字节）传送到目标计算机。

尽管网络会尽力传送每个数据报，但实际上数据报可能会 (1) 丢失、(2) 无序传送、(3) 传送时内容发生改变，甚至 (4) 重复传送并传送多次。连接两端的操作系统通常负责将“尽力而为的数据报”（互联网提供的抽象概念）转换为“可靠字节流”（应用程序通常需要的抽象概念）。

两台计算机必须合作，确保流中的每个字节最终都能在正确的位置传送到另一端的流套接字。它们还

必须告诉对方他们准备从另一台计算机接收多少数据，并确保发送的数据不超过对方愿意接受的数据。所有这些都是使用 1981 年制定的一致方案完成的，该方案称为传输控制协议 (TCP)。

在本实验中，您将仅使用操作系统对传输控制协议的现有支持。您将编写一个名为“网络获取”它创建 TCP 流套接字、连接到 Web 服务器并获取页面 — 就像您之前在本实验中所做的一样。在将来的实验中，您将实现此抽象的另一面，通过自己实现传输控制协议来从不太可靠的数据报中创建可靠的字节流。

3.1 让我们开始吧——获取并构建起始代码

1. 实验室作业将使用入门代码库 填满“海绵”。在您的虚拟机上，跑步
`git 克隆 https://github.com/cs144/sponge` 获取实验室的源代码。
2. 选修的：请随意将您的存储库备份到私人的GitHub/GitLab/Bitbucket 存储库（例如，使用以下说明<https://stackoverflow.com/questions/10065526/github-how-to-make-a-fork-of-public-repository-private>），但请绝对保证您的作品的私密性。
3. 进入Lab 0目录：`CD海绵`
4. 创建目录来编译实验软件：`mkdir 构建`
5. 进入构建目录：`光盘制作`
6. 设置构建系统：`制作...`
7. 编译源代码：`制作`（你可以运行 `制作-j4` 使用四个处理器）。
8. 外面建造目录，打开并开始编辑writeups/lab0.md fi这是您的实验室报告模板，将包含在您的提交内容中。

3.2 现代 C++：基本安全，但速度很快，而且很低级

实验室作业将以现代 C++ 风格完成，使用最新 (2011) 功能尽可能安全地进行编程。这可能与您过去被要求编写 C++ 的方式不同。有关此风格的参考资料，请参阅 C++ 核心指南 (<http://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines>)。

基本思想是确保每个对象都设计为具有尽可能小的公共接口，具有大量内部安全检查，难以不当使用，并且知道如何在之后进行清理。我们希望避免“成对”操作（例如 malloc/free，或

在某些情况下，函数调用可能会发生错误（例如，如果函数提前返回或抛出异常），而另一半则可能不会发生。相反，操作发生在对象的构造函数中，而相反的操作发生在析构函数中。这种风格称为“资源获取即初始化”，简称 RAII。

具体来说，我们希望您：

- 使用以下语言文档<https://en.cppreference.com>作为资源。
- 切勿使用 `malloc()` 或者自由的 `()`。
- 切勿使用新的或者删除。
- 基本上永远不要使用原始指针（*），而要使用“智能”指针（唯一指针或者 共享指针）仅在必要时使用。（在 CS144 中您不需要使用这些。）
- 避免使用模板、线程、锁和虚拟函数。（在 CS144 中你不需要使用这些内容。）
- 避免使用 C 风格的字符串（字符 `*str`）或字符串函数（您可以使用 `strlen()` 和 `strcpy()` 来创建字符串。这些很容易出错。使用 `std::string` 反而。
- 切勿使用 C 风格的强制类型转换（例如，`(文件 *)x`）。使用 C++ 静态转换如果必须的话（在 CS144 中你通常不需要这个）。
- 最好通过以下方式传递函数参数常量参考（例如：`const 地址 & 地址`）。
- 使每个变量常量除非需要进行变异。
- 使每种方法都常量除非它需要改变对象。
- 避免使用全局变量，并赋予每个变量尽可能最小的范围。
- 在提交作业之前，请运行制作格式规范化编码风格。

关于使用 Git：实验室以 Git（版本控制）存储库的形式分布——一种记录更改、检查点版本以帮助调试以及跟踪源代码来源的方式。请在工作时经常进行小提交，并使用提交消息来识别更改的内容和原因。柏拉图式的理想是每次提交都应该编译，并稳步朝着通过越来越多的测试的方向发展。进行小的“语义”提交有助于调试（如果每次提交都编译，并且消息描述了提交所做的一件事，则调试会容易得多），并通过记录您随着时间的推移取得的稳步进展来保护您免受作弊指控——这是一项有用的技能，将有助于任何包括软件开发在内的职业。评分员将阅读您的提交消息，以了解您如何为实验室开发解决方案。如果您还没有学会如何使用 Git，请在 CS144 办公时间寻求帮助或查阅教程（例如，<https://guides.github.com/introduction/git-handbook>）最后，欢迎您将代码存储在私人的 GitHub、GitLab、Bitbucket 等上的存储库，但请确保您的代码不可公开访问。

3.3 阅读 Sponge 文档

为了支持这种编程风格，Sponge 的类用“现代” C++ 包装了操作系统函数（可以从 C 中调用）。

1. 使用 Web 浏览器，阅读以下位置的起始代码文档：<https://cs144.github.io/doc/lab0>。
2. 特别注意[文件描述符](#)，[插座](#)，[TCP套接字](#)，和[地址](#)类。（请注意插座是一种文件描述符，和一个TCP套接字是一种插座。）
3. 现在，找到并阅读描述这些类的接口的头文件libsponge/实用程序目录：文件描述符.hh、套接字.hh、和地址.hh。

3.4 写作网络获取

是时候实施了网络获取，一个程序，使用操作系统的 TCP 支持和流套接字抽象通过互联网获取网页 - 就像您之前在此实验中手动执行的操作一样。

1. 从建造目录中，打开文件../应用程序/webget.cc在文本编辑器或 IDE 中。
2. 在获取 URL函数中，找到以“//您的代码在此。”
3. 按照此文件中的描述，使用您之前使用的 HTTP (Web) 请求格式实现简单的 Web 客户端。使用TCP套接字和地址课程。
- 4.提示：
 - 请注意，在 HTTP 中，每行必须以“\r\n”（仅使用“\n”或完）。
 - 不要忘记在客户端的请求中包含“Connection: close”行。这告诉服务器它不应该等待客户端在此请求之后发送更多请求。相反，服务器将发送一个回复，然后立即结束其传出字节流（一个从服务器的套接字到您的套接字）。您会发现传入的字节流已结束，因为当您读取了来自服务器的整个字节流时，套接字将到达“EOF”（文件末尾）。这样您的客户端就知道服务器已完成回复。
 - 确保阅读并打印全部来自服务器的输出，直到套接字到达“EOF”（文件结尾） - 只需拨打读还不够。
 - 我们预计您需要编写大约十行代码。
5. 继续之前，通过运行 fix it 来编译您[制作]。如果您看到错误消息，则需要程序。

6. 运行测试你的程序 `./apps/webget cs144.keithw.org/你好`。如何
这与您访问时看到的相比如何<http://cs144.keithw.org/hello>在 Web 浏览器中？它与第
节的结果相比如何2.1? 随意尝试——用任何http您喜欢的 URL！

7. 当它看起来正常工作时，运行 `检查 webget` 运行自动化
测试。在实施之前获取 URL函数中，你应该会看到以下内容：

```
1/1 测试 #25: lab0_webget.....***失败函数调用: 0.00 秒
get_URL(cs144.keithw.org, /hasher/xyzyzy)。警告: get_URL() 尚未
实现。
错误: webget 返回的输出与测试的预期不符
```

完成作业后，你将看到：

```
4/4 测试#4: lab0_webget ..... 已通过 0.14 秒
```

```
100% 测试通过, 4 项测试中 0 项失败
```

8. 评分员将评估你的网络获取程序的主机名和路径与 进行检查运行——所以确保它不会 *仅*
*有的*使用主机名和路径进行检查。

4 内存中可靠的字节流

到目前为止，你已经看到了 *可靠字节流* 尽管互联网本身只提供“尽力而为”（不可靠）数据报
的服务，但它在互联网通信方面很有用。

为了完成本周的实验，您将在一台计算机的内存中实现一个提供此抽象的对象。（您可能在
CS 110 中做过类似的事情。）字节在“输入”端写入，并可以按照相同的顺序从“输出”端
读取。字节流是有限的：写入器可以结束输入，然后就不能再写入字节了。当读取器读到流的
末尾时，它将到达“EOF”（文件末尾），并且不能再读取字节了。

你的字节流也将 *佛罗里达州流量控制* 限制其在任何给定时间的内存消耗。该对象初始化为具
有特定“容量”：它愿意在任何给定点存储在其自身内存中的最大字节数。字节流将限制写
入器在任何给定时刻可以写入多少，以确保流不超过其存储容量。当读取器读取字节并从流
中耗尽它们时，写入器可以写入更多。您的字节流用于 *单身的* 线程——您不必担心并发的写入
器/读取器、锁定或竞争条件。

要明确的是：字节流是有限的，但它可以几乎任意长⁴在写入器结束输入并完成流之前。您的实现必须能够处理比容量长得多的流。容量限制了给定点保存在内存中的字节数（已写入但尚未读取），但不限制流的长度。容量仅为一个字节的对象仍然可以承载数 TB 和 TB 长的流，只要写入器每次只写入一个字节，并且读取器在写入器被允许写入下一个字节之前读取每个字节即可。

以下是作者界面的样子：

```
// 将一串字节写入流中。写入尽可能多的字节，并返回写入的字节数。
```

```
size_t 写 (常量标准::细绳&数据) ;
```

```
// 返回流中可以容纳的附加字节数 size_t 剩余容量 () 常量 ;
```

```
// 表示字节流已经到达末尾 空白结束输入 () ;
```

```
// 表明流发生错误 空白设置错误 () ;
```

以下是读者的界面：

```
// 查看流的下一个 “len” 个字节 标准::字符串
```

```
peek_output (常量 size_t 长度) 常量 ;
```

```
// 从缓冲区中删除 “len” 个字节 空白
```

```
pop_output (常量 size_t 长度) ;
```

```
// 读取（即复制然后弹出）流的下一个 “len” 个字节 标准::字符串 读取 (常量 size_t 长度) ;
```

```
布尔值输入结束 () 常量 ; 布尔 // 如果流输入已结束则为 `true`
```

```
值结束语 () 常量 ; 布尔值错误 // 如果输出已到达结尾，则为 `true` // 如果流出现错  
() 常量 ; 误，则为 `true`
```

```
size_t 缓冲区大小 () 常量 ; // 目前可以查看/读取的最大数量 布尔值缓冲区为空 () 常量 ; // `如果缓  
冲区为空，则为 true`
```

```
size_t 写入的字节数 () 常量 ; // 写入的总字节数 // 弹出的总字节数
```

```
size_t 字节读取 () 常量 ;
```

请打开libsp sponge/字节流.hh和libsp sponge/字节流.cc files，和

⁴至少 2⁶⁴字节，在这个类中我们将其视为本质上任意长度

实现提供此接口的对象。在开发字节流实现时，可以使用以下代码运行自动化测试：

`检查 lab0`。

下一步是什么？在接下来的四周内，你将实现一个系统来提供相同的接口，但不再是在内存中，而是在不可靠的网络上。这就是传输控制协议。

5 提交

1. 提交时，请仅对以下内容进行更改网络获取以及顶层的源代码libspoon (字节流.hh和字节流.cc)。请不要修改任何测试或帮助程序libspoon/util。
2. 在提交任何作业之前，请按顺序运行以下内容：
 - (一) `制作格式` (规范化编码风格)
 - (二) `制作` (确保代码可以编译)
 - (三) `检查 lab0` (确保自动化测试通过)
3. 完成编辑writeups/lab0.md，并填写完成这项作业所花费的小时数以及任何其他评论。
4. 准备提交时，请按照以下说明操作<https://cs144.github.io/提交>。提交前请确保您已提交了所有想要提交的内容。只有提交了代码，我们才能对其进行评分。
5. 如果在周三晚上的实验课上遇到任何问题，请尽快告知课程工作人员，或者在 Piazza 上发布问题。祝您好运，欢迎来到 CS144!