实验检查点 0：网络热身

截止日期：1 月 12 日星期日晚上 11:59（延期截止时间：1 月 15 日星期三下午 7 点）

欢迎来到 CS144：计算机网络导论。在这个热身环节中，你将在电脑上安装 GNU/Linux 系统，学习如何手动执行一些互联网操作，编写一个用 C++获取网页的小程序，并实现（内存中）网络的一个关键抽象：写入者与读取者之间的可靠字节流。我们预计这个热身环节需要花费 2 到 6 小时完成（后续实验会占用更多时间）。关于实验任务的三点简要说明：

- 开始前通读全文是个好习惯！ 在这个由八部分组成的实验任务中，你将逐步构建起自己对互联网重要组成部分的实现——包括路由器、网络接口以及 TCP 协议（该协议将不可靠的数据报转换为可靠的字节流）。

- 大多数周次的任务会基于你之前完成的工作，也就是说，你将在整个学期中逐步构建自己的实现，并在后续周次中持续使用你的成果。这使得“跳过”某个检查点变得困难。

大多数周次的任务会基于你之前完成的工作，也就是说，你将在整个学期中逐步构建自己的实现，并在后续周次中持续使用你的成果。这使得“跳过”某个检查点变得困难。

若您尚未满足 CS144 的先修条件，请暂勿选修本课程——我们的教学资源有限。请以检查点 0 和 1 作为衡量标准：如果您在前两个检查点的编程环节中感到吃力，请考虑在后续年份（例如修完 CS 106L 课程后、开展自主编程项目或通过其他方式提升编程熟练度与经验后）再选修 CS144。\n\n 实验文档并非‘技术规范’——即它们并非单向传递的说明材料。其详细程度更接近软件工程师从上级或客户处获得的需求描述。我们建议您通过参与实验课并针对模糊或关键问题提出澄清疑问来充分理解内容。课程网站上的‘实验常见问题’文档将根据后续需要澄清的问题进行更新。

0 合作政策

编程作业必须是你独立完成的：除了我们作为作业提供的代码外，你提交的所有编程作业代码都必须由你亲自编写。请不要从 Stack Overflow、GitHub 或其他来源复制粘贴代码。如果你基于在网上或其他地方找到的示例编写自己的代码，请在你提交的源代码中以注释形式注明该 URL。

与他人合作：你不能向他人展示你的代码，也不能查看他人的代码或往年的解决方案。你可以与其他学生讨论作业，但不能复制任何人的代码。如果你与其他学生讨论了作业，请在你提交的源代码中以注释形式注明他们的姓名。请参考课程

更多详情请参阅行政发放的资料，如有任何不清楚之处可在 EdStem 上提问。GitHub Copilot 或 ChatGPT 等服务应被视为“往年修过 CS144 课程的学生”同等性质。

EdStem 平台：欢迎在 EdStem 上提问，但请勿发布任何源代码。

1 在你的电脑上安装 GNU/Linux 系统

CS144 课程作业要求使用 GNU/Linux 操作系统及支持 C++2023 标准的较新 C++编译器。请从以下三个选项中选择其一：

1. 推荐：安装 CS144 VirtualBox 虚拟机镜像（操作指南见 https://stanford.edu/class/cs144/vm howto/vm-howto-image.html）。
2. 使用 Google Cloud 虚拟机并输入本课程的优惠码（操作指南见 https://stanford.edu/class/cs144/vm howto）。
3. 运行 Ubuntu 24.04 版本，然后安装所需软件包：sudo apt update && sudo apt install git cmake gdb build-essential clang \

```
clang-tidy clang-format gcc-doc pkg-config glibc-doc tcpdump tshark
```

4. 使用其他 GNU/Linux 发行版“风险自负”，但请注意，在此过程中可能会遇到障碍，需要您能够自行调试解决。您的代码将在 Ubuntu 24.04 LTS 系统下使用 g++ 13.3 进行测试，必须确保在该环境下能正常编译和运行。
5. 若您使用的是 2020 至 24 年间搭载 ARM64 架构 M 系列芯片的 MacBook，VirtualBox 将无法正常运行。请改为安装 UTM 虚拟机软件，并从 https://stanford.edu/class/cs144/vm howto/下载我们的 ARM64 虚拟机镜像。

2 手动操作网络

让我们开始使用网络。您需要手动完成两项任务：获取网页（如同浏览器）和发送电子邮件（如同邮件客户端）。这两项任务都依赖于一种称为可靠双向字节流的网络抽象：您将在终端输入一串字节序列，最终这些字节会以相同顺序被传送到另一台计算机（服务器）上运行的程序。服务器会以它自己的字节序列作为响应，传回您的终端。

2.1 获取网页

1. 在网页浏览器中访问 http://cs144.keithw.org/hello 并观察结果。

2. 现在，你将手动完成浏览器所做的相同操作。（a）在你的虚拟机（或你自己的电脑上——例如 macOS 上的终端程序），

运行 telnet cs144.keithw.org http。这指示 telnet 程序在你的电脑与另一台名为 cs144.keithw.org 的电脑之间建立一个可靠的字节流连接，并访问该电脑上运行的特定服务：“http”服务，即超文本传输协议，万维网所使用的协议。

如果你的电脑配置正确且已接入互联网，你将看到：

```
用户@计算机:~$ telnet cs144.keithw.org http 正在尝
试连接 104.196.238.229...
已连接到 cs144.keithw.org。
转义字符为'^]'。
```

如需退出，请按住 ctrl 键并按下]，然后输入 close。

(b) 输入 GET /hello HTTP/1.1 。这会告知服务器 URL 的路径部分。（从第三个斜杠开始的部分。）

（c）输入主机：cs144.keithw.org。这告诉服务器 URL 的主机部分。（位于 http://和第三个斜杠之间的部分。）

（d）输入连接：close。这告诉服务器你已完成请求，它应在回复后立即关闭连接。

（e）再次按下回车键：。这会发送一个空行，告知服务器你已完成 HTTP 请求。（f）如果一切顺利，你将看到与浏览器相同的响应，前面带有告诉浏览器如何解析该响应的 HTTP 头信息。

3. 作业：既然你已经掌握了手动获取网页的方法，现在就来展示一下吧！运用上述技巧访问 URL http://cs144.keithw.org/lab0/sunetid，将 sunetid 替换为你自己的主要 SUNet ID。你将在 X-Your-Code-Is:头部收到一个秘密代码。请保存你的 SUNet ID 和该代码，以便在报告中包含。

2.2 给自己发送一封电子邮件

既然你已经学会了如何获取网页，现在是时候通过另一台计算机上运行的可靠字节流服务来发送电子邮件了。

1. 使用 SSH 登录 sunetid@cardinal.stanford.edu（确保你处于斯坦福网络中），然后运行 telnet 148.163.153.234 smtp。"smtp"服务指的是简单邮件传输协议

[1] 计算机名称有其对应的数字形式（104.196.238.229，即 IPv4 地址），服务名称亦然（80，一个 TCP 端口号）。我们稍后会详细讨论这些内容。

[2] 这些指令在斯坦福大学网络之外可能同样适用，但我们无法保证这一点。

邮件传输协议，用于发送电子邮件消息。如果一切顺利，您将看到：

```
用户@计算机:~$ telnet 148.163.153.234 smtp 正在尝
试连接 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
```

第一步：向邮件服务器标识您的计算机。输入 HELO mycomputer.stanford.edu。等待出现类似 “250... Hello cardinal3.stanford.edu [171.67.24.75], pleased to meet you” 的响应。

3. 下一步：邮件由谁发送？输入 MAIL FROM: sunetid@stanford.edu，将 sunetid 替换为您的 SUNet ID。若一切顺利，您将看到 “250 2.1.0 发件人确认”。

4. 接下来：收件人是谁？初次尝试，可以给自己发送一封邮件。输入 RCPT TO: sunetid@stanford.edu，将 sunetid 替换为您自己的 SUNet ID。
若一切顺利，您将看到 “250 2.1.5 收件人确认”。

5. 现在是上传邮件正文的时候了。输入 DATA 告知服务器你已准备开始。如果一切顺利，你会看到 “354 以<CR><LF>.<CR><LF>结束数据”。

6. 现在你正在给自己撰写一封邮件。首先，输入你将在邮件客户端中看到的邮件头。在邮件头末尾留一个空行。

```
354 以<CR><LF>.<CR><LF>结束数据 发件人：
sunetid@stanford.edu 收件人：
sunetid@stanford.edu 主题：来自 CS144 实
验 0 的问候！
```

7. 输入邮件正文内容——可以是你喜欢的任何内容。完成后，单独一行输入一个点号：.。预期会看到类似这样的信息：“250 2.0.0 33h24dpdsr-1 邮件已接受投递”。

8. 输入 QUIT 可结束与邮件服务器的对话。请检查收件箱及垃圾邮件文件夹，确保已收到邮件。

---

[2] 是的，伪造 “发件人” 地址是可能的。电子邮件有点像邮政服务的实体邮件，其 “退件地址” 的准确性（大部分）依赖于诚信体系。你可以在明信片上随意填写退件地址，电子邮件也基本如此。但请勿滥用此功能——务必谨记。掌握技术知识意味着责任！伪造发件地址的行为常见于垃圾邮件发送者和犯罪分子，他们借此冒充他人。虽然假装自己是 santaclaus@northpole.gov 很有趣，但务必确保不欺骗任何收件人。另外：即使收件人知情这是玩笑，也切勿冒充任何斯坦福大学员工发送邮件（否则可能触发学校 IT 安全警报）。

9. 任务：既然你已经学会如何手动给自己发送电子邮件，现在尝试发送一封给朋友或实验伙伴，并确保他们能收到。最后，向我们证明你能给我们发一封邮件。使用上述方法，从你自己发送一封邮件至 cs144grader@gmail.com。

2.3 倾听与连接

你已经见识了 telnet 的功能：一个能向其他计算机上运行的程序发起出站连接的客户端程序。现在是时候体验作为一个简单服务器的角色了：这类程序会等待客户端与其建立连接。

1. 在一个终端窗口中，在你的虚拟机上运行 netcat -v -l -p 9090。你应该会看到：user@computer:~$ netcat -v -l -p 9090

```
监听在 [0.0.0.0]（协议族 0，端口 9090）
```

2. 保持 netcat 持续运行。在另一个终端窗口中（同样在你的虚拟机上），运行 telnet localhost 9090。

3. 如果一切顺利，netcat 会打印类似“接收到来自 localhost 端口 53500 的连接！”的信息。

4. 现在尝试在任一终端窗口——netcat（服务器端）或 telnet（客户端）——输入内容。注意，你在一个窗口输入的内容会出现在另一个窗口中，反之亦然。
你需要敲击键盘以传输字节。

5. 在 netcat 窗口中，通过输入 ctrl-C 退出程序。注意 telnet 程序也会立即退出。

3 使用操作系统流式套接字编写网络程序

在本热身实验的下一部分，你将编写一个简短的程序，通过互联网获取网页。你将利用 Linux 内核（以及大多数其他操作系统）提供的一项功能：能够在两台计算机之间创建可靠的双向字节流，一台运行在你的计算机上，另一台位于互联网另一端（例如，像 Apache 或 nginx 这样的 Web 服务器，或是 netcat 程序）。
此功能被称为流式套接字。对于您的程序和 Web 服务器而言，套接字看起来像一个普通的文件描述符（类似于磁盘上的文件，或 stdin、stdout I/O 流）。当两个流式套接字连接时，写入其中一个套接字的任何字节最终将以相同顺序从另一台计算机上的另一个套接字输出。

然而实际上，互联网并未提供可靠的字节流服务。它真正所做的只是尽最大努力将称为互联网数据报的短数据片段传递至目的地。每个数据报包含一些元数据（头部），用于指明诸如源地址和目的地址——数据来自哪台计算机，又将送往哪台计算机——以及需递交给目标计算机的部分有效载荷数据（约 1500 字节以内）。

尽管网络试图传递每一个数据报，但实际上数据报可能会（1）丢失，（2）乱序送达，（3）内容被篡改后送达，甚至（4）被复制并多次送达。通常需要连接两端的操作系统将“尽力而为的数据报”（互联网提供的抽象概念）转换为“可靠的字节流”（应用程序通常期望的抽象概念）。

两台计算机必须协同工作，确保流中的每个字节最终都能按正确顺序传递到另一端的流套接字。它们还需相互告知各自准备接收多少数据，并确保不发送超过对方愿意接受的数据量。这一切都是通过 1981 年制定的既定方案——传输控制协议（TCP）来实现的。

在本实验中，你将直接使用操作系统对传输控制协议的现有支持。你将编写一个名为“webget”的程序，创建 TCP 流套接字，连接到 Web 服务器并获取页面——类似于本实验早期的操作。在后续实验中，你将实现这一抽象的另一面，即自行实现传输控制协议，从而在不太可靠的数据报基础上构建出可靠的字节流。

3.1 让我们开始吧——在你的虚拟机和 GitHub 上设置代码仓库

1. 实验作业将使用一个名为“Minnow”的初始代码库。请在你的虚拟机中运行 使用 git clone https://github.com/cs144/minnow 获取实验的源代码。
2. 输入命令 cd minnow 进入 minnow 目录。
3. 在网页浏览器中，你需要在你的 GitHub 账户下创建一个仓库来存放实验作业的解决方案。

   (a) 如果你还没有 GitHub 账户，请访问 https://github.com 创建一个。

   (b) 访问 https://github.com/new 创建一个新仓库。

   (c) 在您的 GitHub 账户中将仓库命名为“minnow”。

   (d) 确保将仓库设置为“私有”，这样您的解决方案就不会公开。

   (e) 点击“创建仓库”。

   (f) 在下一个屏幕上，点击“邀请协作者”，然后“添加人员”。

（g）添加“cs144-grader”为协作者（这将允许我们查看并评分你的代码，同时保持其私密性）。

4. 回到你的虚拟机，通过运行以下命令将 GitHub 仓库注册为目标：
git remote add github https://github.com/用户名/minnow（将“用户名”替换为你实际的 GitHub 用户名）。这将在你的本地实验作业副本（位于虚拟机上）与 GitHub 上的副本（用于备份本地副本并接受评分）之间建立关联。

5. 运行 git push github 将初始代码推送至你的 GitHub 仓库。如果一切顺利，你将看到几行输出信息，最后以* [新分支] main -> main 结尾。若出现错误提示，请仔细检查是否已正确执行上述步骤。此命令会将你的代码上传至 GitHub 上的私有仓库副本，以便我们进行作业评分。

3.2 编译初始代码

1. 仍在“minnow”目录下，创建一个目录来编译实验软件：cmake -S . -B build

2. 编译源代码：cmake --build build

3. 使用你喜欢的文本编辑器（许多学生倾向于通过 SSH 使用 VS Code 编辑文件，但你可以选择任何你喜欢的工具）：打开并开始编辑 writeups/check0.md 文件。这是你实验检查点报告的模板，将包含在你的提交中。

3.3 现代 C++：基本安全，同时保持快速与底层特性

CS144 是一门编程密集课程。实验作业采用现代 C++风格编写，充分利用 2011 年及以后的新特性以实现最大程度的安全编程。这可能与你过去接触的 C++编写方式有所不同。关于此风格的参考，请查阅《C++核心指南》
（http://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines）。

核心理念是确保每个对象的设计具备尽可能小的公开接口，包含大量内部安全检查机制，难以被误用，并能自主清理资源。我们要避免"成对"操作（如 malloc/free 或 new/delete），因为这些操作的后者可能因函数提前返回或抛出异常而无法执行。相反，操作应在对象构造函数中完成，而逆向操作则在析构函数中自动执行。这种模式被称为"资源获取即初始化"（RAII）。

具体而言，我们希望您能做到：

使用 https://en.cppreference.com 上的语言文档作为参考资源。（我们建议避免使用更可能过时的 cplusplus.com。）

切勿使用 malloc()或 free()。

切勿使用 new 或 delete。

基本上不要使用原始指针（*），仅在必要时使用“智能”指针（unique_ptr 或 shared_ptr）。（在 CS144 中你不需要使用这些。）

避免使用模板、线程、锁和虚函数。（在 CS144 课程中你不需要用到这些。）

避免使用 C 风格字符串（char *str）或字符串函数（如 strlen()、strcpy()）。这些极易出错，请改用 std::string。

切勿使用 C 风格类型转换（例如(FILE *)x）。必要时使用 C++的 static_cast（CS144 课程中通常不需要）。

函数参数传递优先采用常量引用方式（例如：const Address & address）。

除非需要修改变量，否则将所有变量声明为 const。

除非需要修改对象状态，否则应将每个方法声明为 const。

避免使用全局变量，并尽可能为每个变量赋予最小的作用域。

提交作业前，运行 cmake --build build --target tidy 获取关于改进 C++编程实践的建议，并运行 cmake --build build --target format 以保持代码格式一致。

关于使用 Git：实验材料以 Git（版本控制）仓库的形式分发——这是一种记录变更、版本检查点以辅助调试以及追踪源代码来源的方式。请在工作过程中频繁进行小规模的提交，并使用能明确说明更改内容及原因的提交信息。理想情况下，每次提交都应能编译通过，并逐步让越来越多的测试用例通过。进行小的“语义化”提交有助于调试（如果每次提交都能编译且提交信息清晰描述该次提交的一个明确改动，调试会容易得多），并通过记录你逐步的进展来保护你免受作弊的指控——这是一项在任何包含软件开发的职业中都有用的技能。评分者将通过阅读你的提交信息来理解你是如何开发实验解决方案的。 若您尚未掌握 Git 的使用方法，请务必在 CS144 的办公时间寻求帮助或参考相关教程（例如：https://guides.github.com/introduction/git-handbook）。最后，虽然我们要求您通过 GitHub 的私有仓库备份并提交代码，但请确保您的代码不会公开可见。

再次强调（因为我们曾教授过这门课程）：在开发过程中频繁进行小规模提交，并使用能清晰说明变更内容及原因的提交信息。

3.4 阅读 Minnow 支持代码

为支持这种编程风格，Minnow 的类将操作系统函数（可从 C 语言调用）封装在“现代”C++中。我们为您提供了针对 CS 111 课程中您应已熟悉概念的 C++封装，特别是套接字和文件描述符。

请仔细阅读公共接口部分（即文件 util/socket.hh 和 util/file_descriptor.hh 中“public:”之后的内容）。（请注意，Socket 是 FileDescriptor 的一种类型，而 TCPSocket 又是 Socket 的一种类型。）

3.5 编写 webget 程序

是时候实现 webget 了，这是一个利用操作系统的 TCP 支持和流套接字抽象从互联网上获取网页的程序——就像你在本实验早些时候手动操作的那样。

1. 在构建目录下，用文本编辑器或 IDE 打开文件../apps/webget.cc。

2. 在 get URL 函数中，按照本文件描述实现简单的 Web 客户端，使用你之前用过的 HTTP（Web）请求格式。利用 TCPSocket 和 Address 类。

3. 提示：请注意在 HTTP 协议中，每行必须以“\r\n”结尾（仅使用“\n”或 endl 是不够的）。

   • 别忘了在客户端请求中包含“Connection: close”这一行。这会告知服务器在本次请求后无需等待客户端发送更多请求。服务器将发送一个响应后立即关闭其输出字节流（即从服务器套接字到您套接字的流）。当您读取完服务器传来的整个字节流时，您的套接字会到达“EOF”（文件结束符），此时您会发现输入字节流已终止。这就是客户端确认服务器已完成响应的方式。• 务必读取并打印服务器所有输出，直到套接字到达“EOF”——仅调用一次 read 操作是不够的。

   我们预计你需要编写大约十行代码。

4. 通过运行 cmake --build build 来编译您的程序。如果出现错误信息，需先修复才能继续。

5. 通过运行 ./apps/webget cs144.keithw.org /hello 测试您的程序。这与在浏览器访问 http://cs144.keithw.org/hello 所见内容有何差异?

网页浏览器？与第 2.1 节的结果相比如何？欢迎随意实验——用你喜欢的任何 http 网址测试它！

6. 当它看起来运行正常时，执行 `cmake --build build --target check_webget` 来运行自动化测试。在实现获取 URL 功能之前，你预期会看到以下内容：

```
$ cmake --build build --target check_webget 测试
项目 /home/cs144/minnow/build
开始 1：使用 bug-checkers 编译 1/2 测试 #1：使用 bug-checkers 编译 通过
1.02 秒
开始测试 2：t_webget 2/2 测试#2：t_webget ***失败 0.01 秒 调用函数：
get_URL(cs144.keithw.org, /nph-hasher/xyzzy) 警告：get_URL()尚未实现。错误：
webget 返回的输出与测试预期不符
```

完成作业后，您将看到：

```
$ cmake --build build --target check_webget 测试
项目 /home/cs144/minnow/build
开始测试 1：带错误检查的编译 1/2 测试#1：带错误检查的编译 通过 1.09 秒
开始测试 2：t_webget 2/2 测试#2：t_webget 通过 0.72 秒

100%测试通过，2 个测试中 0 个失败
```

7. 评分员将使用与 make check webget 不同的主机名和路径运行你的 webget 程序——因此请确保它不仅适用于单元测试中使用的主机名和路径。

4 内存中的可靠字节流

至此，你已经看到了可靠字节流的抽象在跨互联网通信中的实用性，尽管互联网本身仅提供“尽力而为”（不可靠）的数据报服务。

为了完成本周的实验，您将在单台计算机的内存中实现一个提供此抽象功能的对象（您可能在 CS 110/111 课程中完成过类似任务）。字节从“输入”端写入，并可以按相同顺序从“输出”端读取。字节流是有限的：写入者可以终止输入，之后便无法再写入更多字节。当读取者读到流末尾时，将遇到“EOF”（文件结束符），且无法再读取更多字节。

你的字节流也将受到流量控制，以限制其在任何给定时间的内存消耗。该对象初始化时设定了一个特定的“容量”：即它在任何时刻愿意存储在自己内存中的最大字节数。字节流会限制写入者在任何时刻可以写入的数据量，以确保流不会超过其存储容量。随着读取者读取字节并将其从流中排出，写入者被允许写入更多数据。你的字节流适用于单线程环境——无需担心并发写入者/读取者、锁定或竞态条件。

需要明确的是：字节流虽然是有限的，但在写入方结束输入并完成流之前，其长度几乎可以任意延长。您的实现必须能够处理远超容量限制的流。容量限制的是在某一时刻内存中暂存（已写入但尚未读取）的字节数量，而非流的长度限制。即使是一个容量仅为 1 字节的对象，仍可承载长达 TB 级别的流数据，只要写入方每次只写入一个字节，且读取方在写入方被允许写入下一个字节前及时读取当前字节。

以下是写入端的接口定义：

```
void push( std::string data ); // 将数据推入流中，但仅限当前可用容量允许的范围内。
void close(); // 发出流已结束的信号。后续不再写入任何数据。

bool is_closed() const; // 该流是否已关闭？

uint64_t available_capacity() const; // 当前可向流中推送多少字节？
uint64_t bytes_pushed() const; // 累计推送到流中的字节总数
```

以下是读取器的接口：

```
std::string_view peek() const; // 查看缓冲区中的下一个字节 void pop( uint64_t len );
// 从缓冲区移除`len`个字节

bool is_finished() const; // 流是否结束（已关闭且全部弹出）？
bool has_error() const; // 流是否发生过错误？

uint64_t bytes_buffered() const; // 当前缓冲的字节数（已推送未弹出） uint64_t bytes_popped() const; // 从流中累计弹出的字节总数
```

请打开 src/byte_stream.hh 和 src/byte_stream.cc 文件，并实现一个提供此接口的对象。在开发字节流实现时，您可以通过运行 cmake --build build --target check0 来自动化测试。

如果所有测试通过，check0 测试将随后运行您实现的速度基准测试。对于本课程而言，在测试的三种弹出长度下，任何快于 0.1 Gbit/s（即每秒 1 亿比特）的速度都是可接受的。（实现有可能达到超过 10 Gbit/s 的速度，但这取决于您的计算机速度，并非强制要求。）

如有最新问题，请查阅课程网站上的实验常见问题解答，或在实验课中（或通过 EdStem）向同学或教学人员咨询。

[4] 至少可达 2 字节，在本课程中，我们将视其为基本任意长度。

接下来呢？在接下来的四周里，你将实现一个系统，提供相同的接口，但不再局限于内存中，而是在不可靠的网络上运行。这就是传输控制协议（TCP）——其实现可以说是全球最普遍的计算机程序。

5 提交

1. 提交时，请仅修改 webget.cc 及 src 顶层目录下的源代码（byte_stream.hh 和 byte_stream.cc）。请不要改动任何测试文件或 util 中的辅助工具。

2. 编码时请记住频繁进行小规模提交，并附上清晰的提交信息。每次提交后，通过运行 git push github 命令，将虚拟机中的仓库备份至你的私有 GitHub 仓库。你的代码必须提交并推送到 GitHub 才能被评分。

3. 提交任何作业前，请按顺序执行以下操作：(a) 确保已将所有更改提交至 Git 仓库。你可以运行 git status 命令确认没有未提交的变更。记住：编码时请进行小规模多次提交。

   (b) 运行 cmake --build build --target format （以统一代码风格）

   (c) 运行 cmake --build build --target check0 （确保自动化测试通过）

   （注：条目 ID 2 的原文实际应为条目 1 的延续，此处按 YAML 原结构保持独立翻译）

   （d）可选操作：运行 cmake --build build --target tidy（该命令会提出改进建议，以遵循良好的 C++编程实践）

4. 完成编辑 writeups/check0.md 文件，填写完成本作业所花费的小时数及其他任何评论。

5. 确保你的代码已提交并推送至你的私有 GitHub 仓库（执行 git push github）。

6. 请于周日晚上 11:59 前在 Gradescope 作业中提交你的提交记录 ID（commit ID）。

7. 请尽快在周三实验课上向课程工作人员反馈任何问题，或在 EdStem 上发帖提问。祝你好运，欢迎加入 CS144！