实验 1：虚拟机配置和 Socket 编程

A：设置虚拟机

1. 安装 Vagrant

Vagrant 是一种使用单个“Vagrantfile ”中给出的指令自动配置虚拟机的工具。

macOS & Windows:下载链接 **https://www.vagrantup.com/downloads.html.**

Windows:安装结束后，系统会要求你重启电脑。点击“是 ”立即重启，或者稍后手动重启，但不要忘记重启，否则 Vagrant 将无法运行!

Linux: 首先，运行 sudo apt-get update 命令确保你的软件包安装程序是最新的。在安装 Vagrant 之前，运行 sudo apt-add-repository universe 添加“Universe ”版本库。最后，运行 sudo apt-get install vagrant 命令安装 Vagrant。

2. 安装 VirtualBox

VirtualBox 是一种虚拟机提供程序（管理程序）。

macOS & Windows:下载链接 Download_Old_Builds_7_0 – Oracle VirtualBox（请安装 7.0 版本，不要直接使用 7.1 版本）

For macOS Big Sur：安装后，您需要进入系统偏好设置>安全与隐私，并允许来自 Oracle 的系统软件更新（安装过程中可能会有相关提示）。然后，重启 Mac。

Windows only: 使用所有默认安装设置，但可以取消选中“Start Oracle VirtualBox 7.x.x after installation”复选框。

Linux: 运行命令 sudo apt-get 来下载 virtualbox。如果 virtualbox 尚未添加到软件包仓库（如果 apt-get 提示“virtualbox has no installation candidate”），请按照此处的说明操作：https://www.virtualbox.org/wiki/Linux_Downloads

PS: 这也会在计算机上安装 VirtualBox 应用程序，但您不需要运行它，尽管它可能会有所帮助（见步骤 5）。

3. 在 Windows 上安装 Git 和支持 SSH 的终端

Git 是一种分布式版本控制系统。

macOS & Windows:下载链接 https://git-scm.com/downloads

macOS:打开 .dmg 安装文件后，你会看到一个 Finder 窗口，其中包含一个 .pkg 文件，即安装程序。正常打开该文件时，可能会出现提示，称由于该文件来自不明开发者，因此无法打开。要覆盖这种保护，可以右键单击 .pkg 文件并选择“打开”。这时会出现一个提示，询问你是否确定要打开它。选择“是”。这将带你进入（直接的）安装程序。

Windows:安装过程中会有很多选项供你选择；使用所有默认设置就足够了（你可以在最后取消选中“View release notes”）。安装包含一个支持 SSH 的 Bash 终端，通常位于 C:\Program Files\Git\bin\bash.exe。除非你喜欢其他支持 SSH 的终端，否则你应该使用它作为本课的终端。你可以随意为它创建快捷方式；不过，将可执行文件复制并粘贴到其他地方是行不通的。

Linux: sudo apt-get install git.

4. 下载 X Server

您需要一个 X Server 来向虚拟机输入命令。

macOS: 下载 XQuartz。您需要退出并重新登录才能完成安装（如最后的提示所述）。

Windows: 下载 XmingDownload Xming-6-9-0-31-setup.exe (Xming X Server for Windows) (sourceforge.net) 使用默认选项，最后取消选中“启动 Xming”。

Linux: 已预装 X Server

5. 使用 Vagrant 配置虚拟机

打开您的终端（如果使用 Windows，请使用步骤 3 中提到的终端），然后 cd 到您在计算机上保存本课程文件的位置，cd COS461-Public/assignments 进入课程任务目录。在刚刚输入的任务目录下，运行 vagrant up 命令启动虚拟机。

你可能需要等待几分钟。你可能会看到 default，例如 "default: dpkg-preconfigure: unable to re-open stdin: No such file or directory"，不过你不用担心。

注 1：以下命令允许你在任何时候停止虚拟机（比如当你完成当天的任务时）：
vagrant suspend：将保存虚拟机的状态并停止它。
vagrant halt：会关闭虚拟机操作系统，并关闭虚拟机电源。
vagrant destroy：会从系统中删除虚拟机的所有痕迹。如果你在虚拟机上保存了重要文件（如作业解决方案），切勿使用此命令。
此外，如果你不确定虚拟机的状态（如正在运行、已关机、已保存……），可以使用 vagrant status 命令查看。使用上述命令时，你必须在包含 Vagrant 文件的目录下的某个子目录中，否则 Vagrant 无法知道你所指的是哪台虚拟机。

注 2：在步骤 2 中安装的 VirtualBox 应用程序提供了一个可视化界面，作为这些命令的替代，你可以在这里查看虚拟机的状态，打开/关闭虚拟机或保存其状态。不过，不建议使用它，因为它没有与 Vagrant 集成，而且键入命令的速度也不会慢。它也不能替代最初的 vagrant up，因为它会创建虚拟机。

6. 测试 SSH 连接 VPN

在终端运行 vagrant ssh。这是你每次访问虚拟机时都要使用的命令。如果运行成

功，你的终端提示将变为 vagrant@cos461:~$。

所有其他命令都将在虚拟机上执行。然后，你就可以运行 cd /vagrant 命令，进入课程目录。

Vagrant 的这种共享目录结构特别有用。你无需将文件复制到虚拟机或从虚拟机复制文件。Vagrant 文件所在的任务目录中的任何文件或目录都会自动在你的电脑和虚拟机之间共享。这意味着你可以在虚拟机外使用自己选择的集成开发环境编写代码（但仍需在虚拟机内构建和运行）。

logout 命令可在任何时候停止 SSH 连接。

B：Socket 编程

套接字编程是编写网络通信程序的标准方法。虽然套接字编程最初是为使用 C 语言编程的 Unix 计算机开发的，但套接字抽象是通用的，与任何特定的操作系统或编程语言无关。这使得程序员可以在许多情况下使用套接字编写正确的网络程序。

您将编写两对 TCP 客户端和服务器程序，用于在互联网上发送和接收文本信息。其中一对客户端/服务器程序必须用 C 语言编写，另一对可以用 Python 或 Go 语言编写。您可以选择其中一种语言。

服务器要求：

1.每个服务器程序都应该监听 socket，等待客户端连接，从客户端接收消息，将消息打印到标准输出，然后无限期地等待下一个客户端。

2.每台服务器都应采用一个命令行参数：监听客户端连接的端口号。

3.每个服务器应该在无限循环中接受和处理客户端通信，从而允许多个客户端向同一服务器发送消息。 服务器应该只在响应外部信号 (例如按 ctrl-c 的 SIGINT) 时退出。

4.每个服务器应维护一个短的 (5-10 个) 客户端队列，并依次处理多个客户端连接尝试。 在实际应用中，TCP 服务器将派生一个新进程来并发地处理每个客户端连接，但这对于此分配来说不是必需的。

5.每个服务器都应该妥善处理 socket 编程库函数可能返回的错误值（请参阅下面每种语言的具体细节）。 与处理客户端连接相关的错误不应导致服务器在处理错误后退出。

客户端规范：

1.每个客户端程序都应该联系服务器，从标准输入读取消息，发送消息，然后退出。

2.每个客户端都应准确读取和发送输入中的信息，直至到达 EOF（文件结尾）。

3.每个客户端应接受两个命令行参数：服务器的 IP 地址和服务器的端口号。

4.每个客户端都必须能够通过迭代读取和发送信息块来处理任意大的信息，而不是先将整个信息读入内存。

5.每个客户端在处理部分发送（当套接字只发送上次发送调用中给出的部分数据时）时，都应尝试重新发送其余数据，直到全部数据都发送完毕。

6.每个客户端都应妥善处理 socket 编程库函数可能返回的错误值。

测试脚本发送信息：

0. The short message "Go Tigers!\n"
1. A long, randomly generated alphanumeric message
2. A long, randomly generated binary message
3. Several short messages sent sequentially from separate clients to one server
4. Several long, random alphanumeric messages sent concurrently from separate clients to one server

提示：

请在 Vagrant 虚拟机上完成所有构建和测试。你既可以在 Vagrant 虚拟机上编写代码（预装了 Emacs 和 Vim 文本编辑器），也可以直接在操作系统上编写代码（允许你使用任何已安装的编辑器）。在终端运行 vagrant ssh 后，运行 cd /vagrant 进入课程目录。

我们在 assignment1/client_server/ 目录中提供了基础代码。在开始编程之前，你应该阅读并理解这些代码。

您只能在所提供文件中标有 TODO 注释的位置编程。每个客户端和每个服务器都有一个 TODO 部分。您可以根据需要添加函数，但不要更改文件名，因为它们将用于自动测试。

以下各节将详细介绍每种语言的客户端和服务器程序。

C：

文件 client-c.c 和 server-c.c 包含基本代码。您需要在标有 TODO 的位置添加 socket 编程和 I/O 代码。参考解决方案在每个文件的 TODO 部分都有大约 70 行代码（注释清楚、间隔整齐）。您的实现可能更短或更长。

在错误处理方面，对于设置全局变量 errno 的 socket 编程函数，可以调用 perror（Beej's Guide 会告诉你哪些函数设置了 errno）。对于那些不设置全局变量 errno 的函数，只需在标准错误中打印一条信息即可。

您应在 assignment1/client_server 目录下运行.c 文件来构建解决方案。您的代码必须使用提供的 Makefile 生成。服务器应该以 ./server-c [port] > [output file] 的方式运行。客户

端应以 ./client-c [server IP] [server port] < [message file] 的方式运行。更多详情，请参阅“测试”。

参考资料：

Beej's Guide to Network Programming Beej's Guide to Network Programming

函数介绍：Beej's Guide to Network Programming

范例：Beej's Guide to Network Programming

Python：

文件 client-python.py 和 server-python.py 包含基础代码。您需要在标有 TODO 的位置添加套接字编程代码。参考解决方案在每个文件的 TODO 部分都有大约 15 行代码（注释清晰、间距合理）。您的实现可能更短或更长。

Python socket 函数会自动引发异常，并给出有用的错误信息。无需额外的错误处理。

服务器应该以 python server-python.py [port] > [output file] 的方式运行。客户端应以 python client-python.py [server IP] [server port] < [message file] 的方式运行。详见“测试”。

参考资料：

Python socket 编程的文档：17.2. socket — Low-level networking interface — Python 2.7.18 documentation

函数介绍：17.2. socket — Low-level networking interface — Python 2.7.18 documentation

范例：17.2. socket — Low-level networking interface — Python 2.7.18 documentation

Go：

您应该在 assignment1/client_server 目录下运行 go 文件来构建您的解决方案。您的代码必须使用提供的 Makefile 构建。服务器的运行方式应为 ./server-go [port] > [output file]。客户端的运行方式应为 ./client-go [服务器 IP] [服务器端口] < [信息文件]。更多详情，请参阅“测试”。

参考资料：

Go socket 编程文档：net package - net - Go Packages

范例：net package - net - Go Packages

测试：

您应尝试从客户端向服务器发送信息，来完成测试。服务器可以在后台运行（在命令中

添加 & 号)，也可以在单独的 SSH 窗口中运行。服务器 IP 应为 127.0.0.1，服务器端口号应在 10000 到 60000 之间。可以使用 fg 命令将后台服务器切换到前台，然后按 ctrl-c 键将其关闭。

Bash 脚本 test_client_server.sh 将尝试在所有 4 种客户端和服务器（C client to C server, C client to Python/Go server, etc.）之间发送几条不同的消息，来完成测试。

请使用./test_client_server.sh [python|go] [server port]命令来运行测试脚本。

如果出现权限错误，请运行 chmod 744 test_client_server.sh，赋予脚本执行权限。

对于每一对客户机/服务器，如果信息收发正确，测试脚本将打印“SUCCESS”。否则，它将打印发送和接收信息的差异（即仅针对测试 1 和测试 4）。

在运行 test_client_server.sh 之前，请确保构建了 C client/server。