《网络编程》大作业任务书

说明

– **所需环境**：系统环境 Linux，编程语言 C。

– **项目内容**：包含三个任务，均为必做内容。

– **考核方式**：包括项目报告和源代码两部分。项目报告需展示所有必要的内容，如服务器处理流程、结构体定义和命令实现函数的关键代码截图和相关说明、测试截图等，不要大范围复制任务书原文；源代码需包含必要的注释和运行说明文档（README 文件）。

– **提交方式**：请将项目报告（名为 report）和所有源代码（code 文件夹，包括源代码和 README）保存到一个压缩包。提交链接为请直接点击蓝字跳转或右键复制链接到浏览器中进入提交页面。请务必于 7 月 12 日或之前提交，过期不候。

– **注意事项**：本课程没有笔试考试，大作业的成绩占比为 70%，无论完成程度如何，请务必提交大作业的项目报告和源代码，否则按挂科处理。鼓励与其他人讨论交流问题与思路，但报告和代码均需个人单独完成，不要抄袭！报告或代码雷同者同样按挂科处理。

chirc

一. 引言

在该项目中，你将实现一个名为 chirc 的简单网络中继聊天服务器（IRC）。IRC 是最早的文本消息传递和多人聊天网络协议之一。如今，它仍是一个流行的标准，在某些社区，特别是开源软件社区，仍然大量使用。

你的实现必须与其他 IRC 客户端（不是由你编程）的官方 IRC 规范足够兼容才能与你的服务器配合使用。虽然我们将提供一些支架代码，但大多数任务都要求你查阅官方的 IRC 规范，或者试验现有的 IRC 服务器。因此，该项目不仅可以帮助你开发网络编程技能，还可以提升你阅读和解释网络协议的能力。

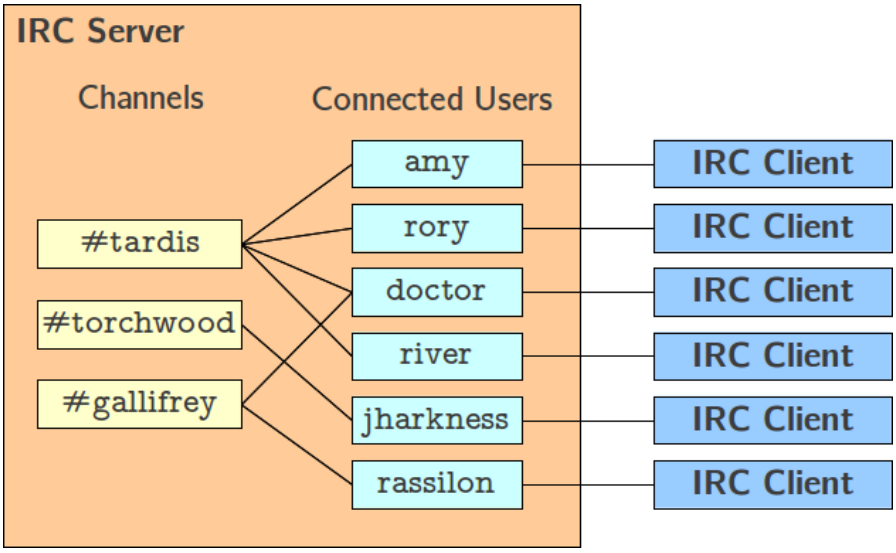
该项目分为三个部分。 第一部分是一个相对较短的热身运动；第二部分主要围绕支持多个客户端和个人用户之间的消息发送；第三部分主要围绕实施 IRC“频道”（相当于“聊天组”或“聊天室”）。

chirc 文档分为以下几个部分：

- 网络中继聊天和示例 IRC 通信提供了 IRC 协议的概述及几个有效的 IRC 通信示例。
- 安装，构建和运行 chirc 描述了如何获取 chirc 代码以及如何构建和运行。
- 任务 1：基本消息处理，任务 2：多客户端支持，任务 3：通道和模式，描述该项目的三个部分。
- 测试实现提供了测试实现的建议和策略。

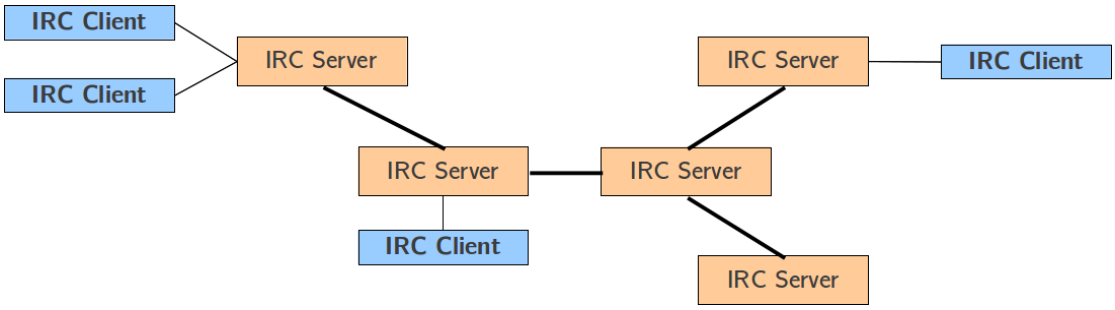
二. 网络中继聊天服务器

IRC 是最早的文本消息传递和多人聊天网络协议之一。它创建于 1988 年，尽管后来出现了更复杂的消息传递协议（包括 XMPP 和 SIP / SIMPLE 等开放性标准，以及 Microsoft 的 MSNP，AOL 的 OSCAR 和 Skype 等专有协议），IRC 仍然是一个受欢迎的标准，在某些社区，特别是开源软件社区，仍然大量使用。



IRC 基本结构

如上图所示，IRC 的基本架构非常简单。在最简单的情况下，多个 IRC 客户端可以连接到一个 IRC 服务器。IRC 客户端使用特定标识连接到服务器。 最值得注意的是，每个客户必须选择一个独特的昵称（“nick”）。客户端连接后，它可以与其他用户进行一对一的通信。另外，客户端可以通过命令来查询服务器的状态（例如，获得连接用户的列表、获得关于特定“nick”的额外细节）。IRC 还支持创建名为频道（channels）的聊天室，以进行一对多通信。用户可以加入 channels 并发送消息，这些消息将被发送给频道中的每个用户。



多用户 IRC 结构

IRC 还支持服务器网络的形成，其中多个服务器形成连接树以支持更多客户端并提供更大的容量。 同一网络中的服务器共享本地事件信息（例如，新客户端连接，连接到特定服务器的用户加入 channel 等），因此所有服务器将具有一份相同的全局状态副本。在这个项目中，我们只考虑有一个 IRC 服务器的情况。

三. IRC 协议

IRC 服务器和客户端使用的 IRC 协议是基于文本的 TCP 协议。最初在 1993 年[RFC1459]中指定，随后在 2000 年通过以下 RFC 进行了更详细的说明：

- [RFC2810] Internet Relay Chat：Architecture。 本文档描述了 IRC 的整体架构。
- [RFC2811] Internet Relay Chat: Channel Management。本文档描述了如何在 IRC 中管理频道（channels）。
- [RFC2812] Internet Relay Chat: Client Protocol。本文档描述了 IRC 客户端和服务器之间使用的协议（有时称为“客户端 - 服务器”协议）。
- [RFC2813] Internet Relay Chat: Server Protocol。本文档描述了同一网络中 IRC 服务器之间使用的“服务器 - 服务器”协议。

不建议你阅读以上所有文档，进一步来说：

- 建议你阅读 [RFC2810]全部内容，因为它可以让你很好地了解 IRC 架构的外观。你首先应粗略的阅读它，并在你更熟悉 IRC 协议的细节时重读该文档。
- 在第二个任务中，你将实现[RFC2812]的部分内容。建议你阅读[RFC2812 §1]和[RFC2812 §2]。对于 RFC 的其余部分，你应该只阅读与你将要实现的 IRC 协议部分相关的部分。
- 在第三个任务中，你将实现[RFC2811]中描述的部分功能，这将需要实现[RFC2812]的其他部分。建议你继续阅读[RFC2811]，直到我们完成第三项任务为止；如果你想阅读介绍部分，请考虑到我们只支持“ # ”命名空间中的“standard channels”，并且我们不会支持服务器网络。
- 不实现[RFC2813]的任何部分。

最后，你应该考虑到虽然 IRC 有官方规范，但大多数 IRC 服务器和客户端都不符合这些 RFC。大多数（如果不是全部）服务器不会实现完整的规范（在某些情况下甚至与其相矛盾），并且有许多基于特定实现的特性。在该项目中，我们将生成一个部分符合这些 RFC 的实现，并且足以与当前可用的一些主要 IRC 客户端协同工作。

在本节的其余部分中，我们将看到 IRC 中使用的消息格式的概述。 然后，在下一节中，我们将看到几个示例通信（涉及客户端和服务器之间的多条消息）。

消息格式

IRC 客户端和服务器通过 TCP 发送纯 ASCII 消息进行通信。这些消息的格式在[RFC2812 §2.3]中有描述，可以这样概括：

- IRC 协议是基于文本的协议，意味着消息以纯 ASCII 编码。 尽管不如纯二进制格式有效，但它具有相当人性化的优点，并且只需逐字读取客户端和服务器之间交换的消息即可轻松调试。
- 单个消息是一串字符，最大长度为 512 个字符。字符串的末尾用 CR-LF（回车-换行）对表示（即“\ r \ n”）。没有空终止符。512 个字符的限制包括此分隔符，这意味着消息仅包含 510 个有用字符的空间。
- 虽然许多服务器和客户端支持非标准解决方案（包括完全忽略 512 限制），但 IRC 规范不包括支持长度超过 512 个字符的消息。在我们的实现中，任何超过 510 个字符（不包括分隔符）的消息都将被截断，最后两个字符将替换为“\ r \ n”。
- 消息至少包含两部分：命令和命令参数。最多可能有 15 个参数。命令和参数都由单个 ASCII 空格字符分隔。以下是有效 IRC 消息的示例：

```
NICK amy


WHOIS doctor


MODE amy +o


JOIN #tardis


QUIT
```

- 当最后一个参数以冒号为前缀时，该参数的值将是消息的剩余部分（包括空格字符）。以下是带有“长参数”的有效 IRC 消息的示例：

```
PRIVMSG rory :Hey Rory...


PRIVMSG #cmsc23300 :Hello everybody
```

```
QUIT :Done for the day, leaving
```

- 某些消息还包括处于命令和命令参数前的前缀。前缀的存在用单个前导冒号表示，用于指示消息的来源。例如，当用户向频道发送消息时，服务器会将该消息转发给频道中的所有用户，并且将包括前缀以指定最初发送该消息的用户。我们将在下一节中更详细地解释前缀的使用。
  以下是带有前缀的有效 IRC 消息的示例：

```
:borja!borja@polaris.cs.uchicago.edu PRIVMSG #cmsc23300 :Hello everybody


:doctor!doctor@baz.example.org QUIT :Done for the day, leaving
```

回复

IRC 协议包括称为回复的特殊类型消息。当客户端向服务器发送命令时，服务器将发送回复（除少数不应回复的特殊命令外）。回复用于确认命令被正确处理，指示错误，或在命令执行服务器查询时提供信息（例如，询问用户或频道列表）。

回复是具有以下特征的消息：

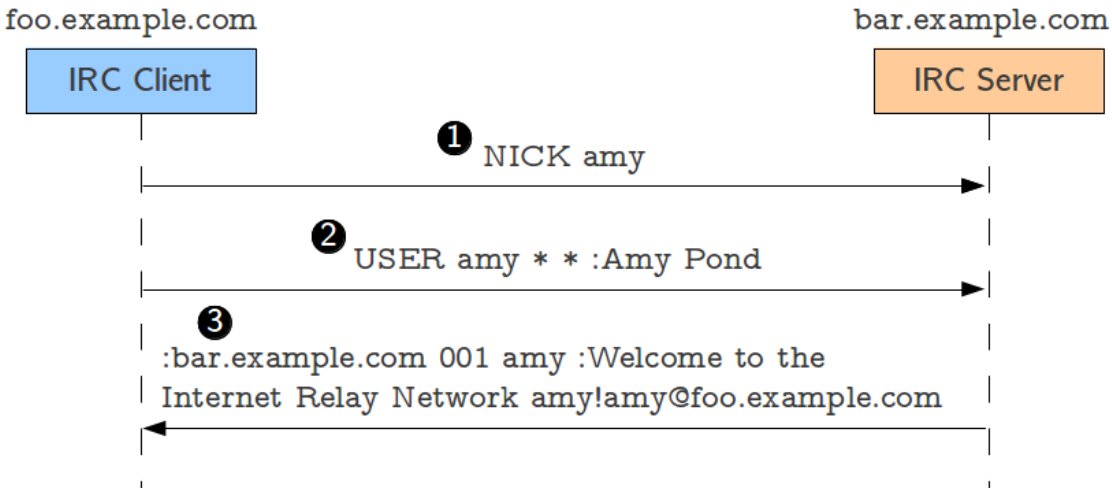
- 总是包含一个前缀。
- 是一个三位数的代码。可能的答复完整列表在[RFC2812§5]中指定。
- 第一个参数是回复的目标，通常是昵称（nick）。

```
:irc.example.com 001 borja :Welcome to the Internet Relay Network
 borja!borja@polaris.cs.uchicago.edu


:irc.example.com 433 * borja :Nickname is already in use.


:irc.example.org 332 borja #cmsc23300 :A channel for CMSC 23300 s
tudents
```

四. IRC 通信示例

在本节中，我们将描述三个示例 IRC 通信。在深入研究 IRC RFC 之前，建议你仔细阅读这些示例，以便更好地了解 IRC 客户端和服务器之间的通信。这些示例还将用于阐明消息、前缀和回复的格式。

登录到 IRC 服务器



连接到 IRC 服务器

当 IRC 客户端连接到 IRC 服务器时，必须首先注册其连接。这是通过发送 NICK 和 USER（上图中的消息 1 和 2）来完成的。NICK 指定用户的昵称（在本例中为 amy），USER 提供有关用户的其他信息。更具体地说，USER 指定用户的用户名（amy）和用户的全名（Amy Pond）（我们不实现 USER 的第二和第三个参数）。用户名通常由 IRC 客户端根据用户的身份自动获得。例如，如果你以用户 jrandom 登录到 UNIX 计算机，则默认情况下，大多数 IRC 客户端将使用该名称作为你的用户名。你的昵称不必一定与你的用户名相匹配。

假设你选择了尚未采用的 nick，IRC 服务器将发回 RPL_WELCOME 回复（分配代码 001）。此回复包含以下部分：

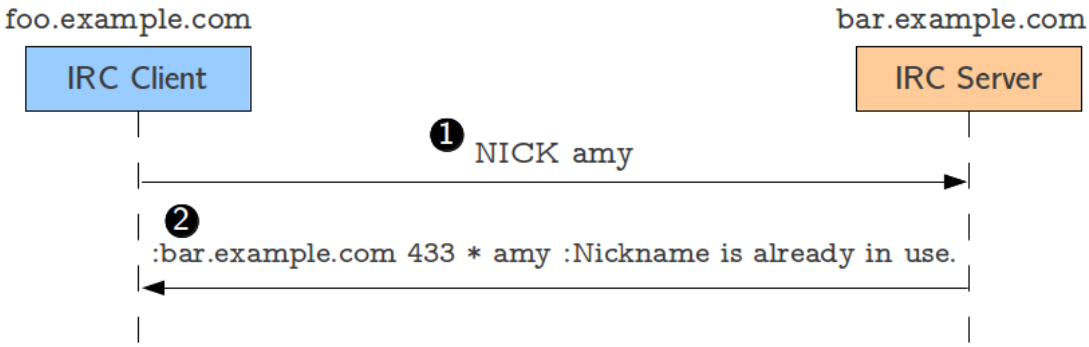
- ：bar.example.com：前缀，用于指示消息的来源。由于此回复源自服务器 bar.example.com，因此前缀仅包含该主机名。这可能看起来多余，因为客户端可能已经知道它已连接到该服务器；但是，在 IRC 网络中，回复可能源自客户端连接的服务器以外的服务器。
- 001：RPL_WELCOME 的数字代码。
- amy：第一个参数，在回复消息中，必须始终是此回复所针对的用户的 nick。
- ：Welcome to the Internet Relay Network borja!borja@polaris.cs.uchicago.edu：第二个参数。该参数的内容在[RFC2812§5]中指定：

```
001    RPL_WELCOME

       "Welcome to the Internet Relay Network

        <nick>!<user>@<host>"
```

请注意回复是如何省略第一个参数，该参数始终是回复的接收者。因此，规范列出了第二个和后续参数（如果有的话）。

在 RPL_WELCOME 回复中发回的一部分是完整的客户端标识符（<nick>！<user> @ <host>），它也用于其他类型的消息。它由 NICK 命令中指定的 nick，USER 中指定的用户名和客户端的主机名组成（如果服务器无法解析客户端的主机名，则使用 IP 地址）。

下图显示了上述通信的变形：



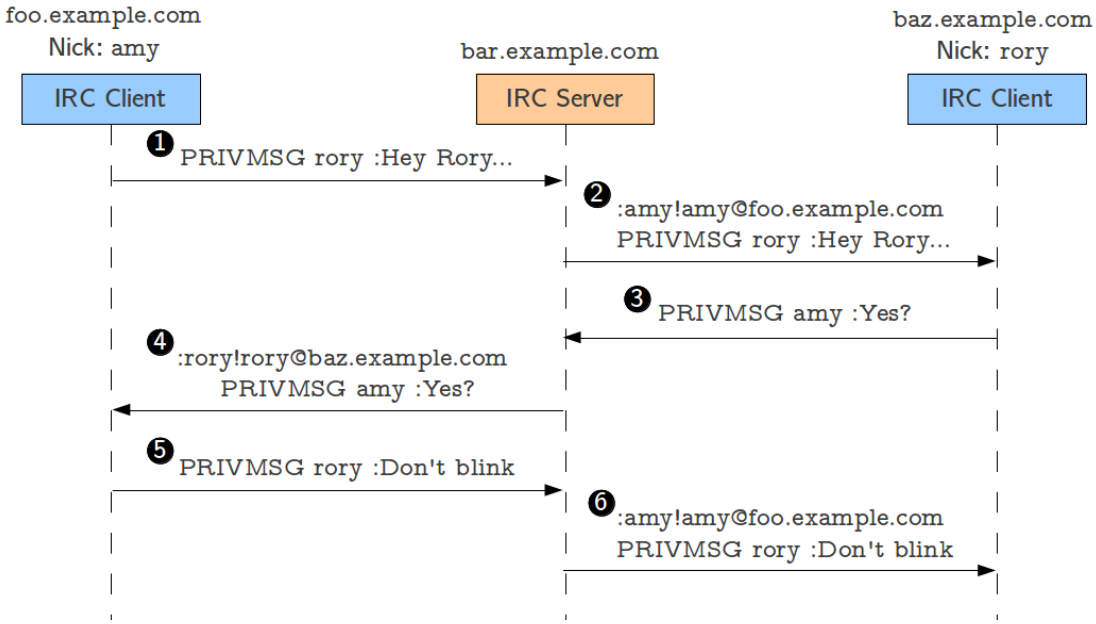
当 nick 已经被使用时连接 IRC 服务器

如果用户尝试用已被使用的 nick 进行注册，服务器将发回 ERR_NICKNAMEINUSE 回复（代码 433）。请注意此回复中的参数有何不同：

- *：第一个参数应该是此回复所针对的用户的 nick。但是，由于用户还没有 nick，因此使用星号字符。
- amy：在 ERR_NICKNAMEINUSE 回复中，第二个参数是“有问题的 nick”（即无法选择的 nick，因为它已被采用）。
- ：Nickname is already in use：第三个参数仅包含人可读的错误描述。 IRC 客户通常会逐字打印这些内容。

因此，请注意在所有回复中没有统一的参数集（除了第一个参数，它始终是收件人 nick）。在实现回复时，你必须参考[RFC2812 §5]以确定你应该在回复中发回的内容。

用户间的消息传递



向另一用户发送消息

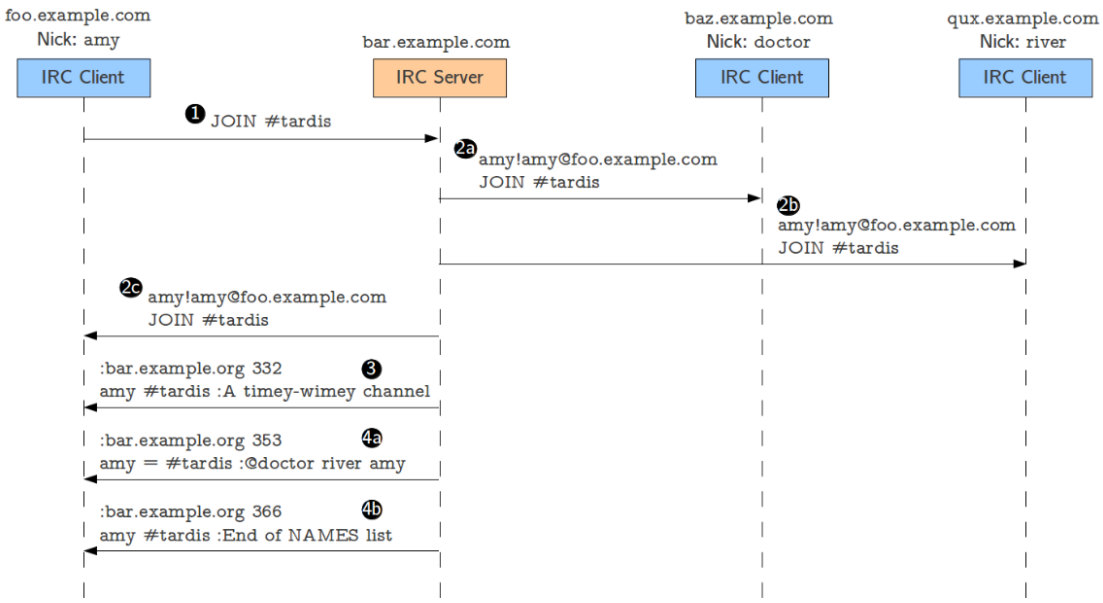
一旦连接了多个用户，即使没有频道，他们也可以相互发送消息。第二项任务的大部分侧重于实现用户之间的消息传递，而第三项任务将侧重于增加对频道的支持。

要将消息发送到特定 nick，用户必须将 PRIVMSG 发送到服务器。上图显示了两个用户 amy 和 rory，交换了三条消息。在消息 1 中，用户 amy 向 rory 发送消息。PRIVMSG 的参数非常简单：第一个参数是消息所针对的用户的 nick，第二个参数是消息本身。

当服务器收到该消息并假设有一个 rory 用户时，它会将消息转发给注册该 nick 的 IRC 客户端。这是在消息 2 中完成的，并注意它只是消息 1 的副本，但以 amy 的完整客户端标识符为前缀（否则，接收方 IRC 客户端将不知道消息来自谁）。消息 3 和 4 显示了类似的交换，除了从 rory 到 amy，消息 5 和 6 显示从 amy 到 rory 的另一条消息。

注意所有消息是如何通过 IRC 服务器中继的（因此协议名称：Internet Relay Chat）。IRC 规范不支持非中继消息传递，我们不会在此项目中实现此类功能。但是，IRC 有两个扩展（CTCP：客户端到客户端协议和 DCC：直接客户端到客户端），它们是 IRC 上非中继聊天的标准。大多数 IRC 服务器和客户端都支持这些扩展，即使它们从未被正式指定为 RFC（最接近规范的是本文档：http：//www.irchelp.org/irchelp/rfc/ctcpspec.html）。

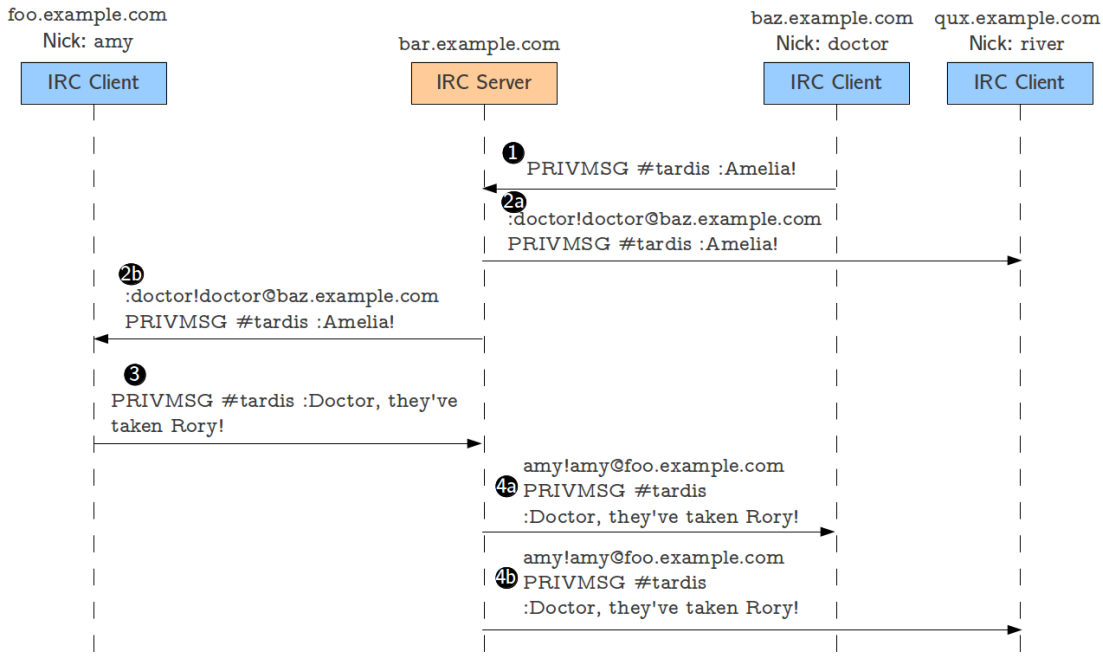
加入，聊天，以及离开频道



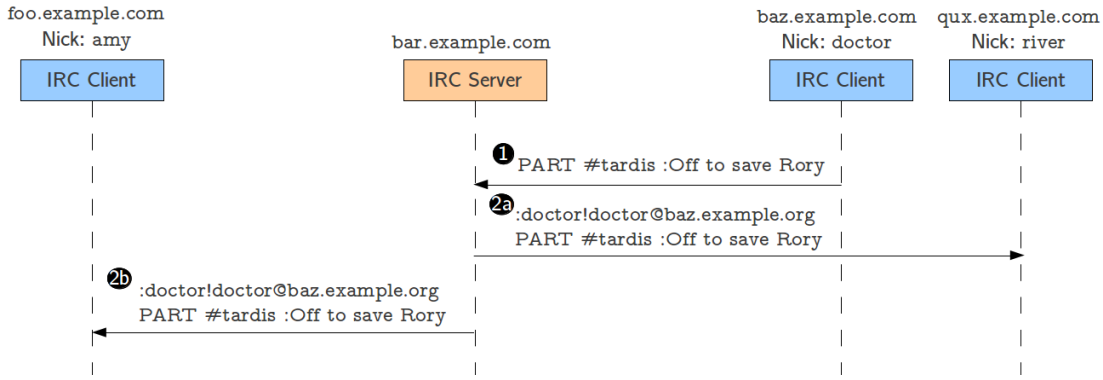
连接到 IRC 服务器的用户可以使用 JOIN 消息加入现有频道。消息本身的格式非常简单（唯一的参数是用户想要加入的频道的名称），但它会导致多个回复不仅发送给加入频道的用户，还发送给当前频道中的所有用户。上图显示了当用户 amy 加入频道#tardis 时会发生什么，其中两个用户（doctor 和 river）已经存在。

消息 1 是 amy 对服务器的 JOIN 消息。当收到此消息时，服务器将其转发给已经在频道（doctor 和 river）中的用户，使他们知道频道中有新用户（消息 2a 和 2b）。 注意转发的 JOIN 如何以 amy 的完整客户端标识符作为前缀。JOIN 也被转发回 amy，作为她成功加入频道的确认。

后面的消息（3,4a 和 4b）为 amy 提供有关该频道的信息。消息 3 是 RPL_TOPIC 回复，提供频道的主题（这是可以由某些用户设置的频道的描述，我们将在后面详细讨论）。消息 4a 和 4b 分别是 RPL_NAMREPLY 和 RPL_ENDOFNAMES 回复，它们告诉 amy 当前频道中存有哪些用户。注意 doctor 用户在他的昵称之前有一个签名，这表明医生是渠道#tardis 的渠道运营商。正如我们在第三个任务中看到，用户可以拥有在服务器或单个频道上的特权模式。例如，频道操作员通常是唯一可以更改频道主题的用户类型。



一旦用户加入了频道，向频道发送消息与向个人用户发送消息基本相同。不同之处在于服务器会将消息转发到频道中的所有用户，而不是仅仅是单个用户。上图显示了发送到频道 #tardis 的两条消息。首先，用户 doctor 发送 PRIVMSG 消息，指定频道作为目标（而不是 nick，正如我们在“用户之间的消息传递”中所看到的）。然后，服务器将此消息转发到 river 和 amy，在消息前面加上 doctor 的完整客户端标识符（消息 1,2a 和 2b）。同样，amy 向该频道发送一条消息，该消息被转发给 doctor 和 river，前缀为 amy 的完整客户端标识符（消息 3,4a 和 4b）。



发送 PART 消息可以离开频道，该消息类似加入频道和在频道中聊天的模式：希望离开的用户发送 PART 消息，并且该消息被转发到频道中的每个人，让他们知道该用户已经离开了。服务器还在内部从频道中删除该客户端，这意味着他将不再接收任何定向到该频道的消息。上图显示了这种情况的示例。PART 消息包括两个参数：用户想要离开的频道，以及“退出消息”（作为 PART 消息的一部分被转发到频道中的所有用户）。

五. 安装，构建及运行 chirc

chirc 的源代码可以在以下 GitHub 存储库中找到：

https://github.com/uchicago-cs/chirc

要进行任务，你需要做的就是克隆此仓库。

软件要求

除标准 C 编译器和标准 C 库外，chirc 本身没有特殊的软件要求。但是，自动化测试（在测试实施中描述）需要以下软件：

Python 3.4 或更高版本

py.test，包括插件 pytest-html 和 pytest-json。 所有这些都可以使用 pip 安装（pip3 install pytest pytest -html pytest-json）

构建

获得 chirc 代码后，只需运行 make 即可构建它。这将生成一个名为 chirc 的可执行文件，它接受以下参数：

- -p：服务器将侦听的端口。
- -o：指定操作员密码。
- -q，-v 或-vv：控制日志记录的级别。 请参阅下面的“记录”部分。

要修改代码，只需将文件添加到 src/目录。请注意，如果添加其他.c 文件，则需要修改 Makefile 文件，以便把它们包含在构建中（更具体地说，你需要在 OBJS 变量中包含新的目标文件）。

运行

你至少要使用-o 选项运行可执行文件，尽管此选项在第三个任务之前都没有太大关系。例如：

```
./chirc -o foobar
```

但是，提供的代码不会执行任何其他命令行参数的代码。你应该验证它是否正确构建和运行。

注意：你的代码必须遵守命令行参数中指定的值。更重要的是，如果不使用-p 参数中指定的端口，则代码所有自动化测试都将失败。

记录

chirc 服务器使用简单日志记录函数 chilog（）将消息打印到标准输出，该函数在 src / log.h 中声明。如果需要将消息打印到标准输出，则必须使用 chilog（）函数。 这是一个简单的函数，它需与 printf 相同的参数，以及指定日志记录级别的附加参数。 例如：

```
chilog(INFO, "User with nick %s has connected", nick);
```

不要在代码中直接使用 printf（）。仅能使用 chilog（）将消息打印到标准输出。

chilog（）的第一个参数用于指定日志级别：

- CRITICAL：用于严重错误，唯一的解决方案是退出程序。
- ERROR：用于非严重错误，可能允许程序继续运行，但其中的特定部分失败（例如单个套接字）。
- WARNING：用于指示意外情况，虽然技术上不是错误，但可能导致错误。
- INFO：用于打印有关程序状态的概括信息。
- DEBUG：用于打印有关程序状态的详细信息。
- TRACE：用于打印低级信息，例如函数入口/出口点，整个数据结构的转储等。

运行 chirc 时，记录级别由 q 和-v 参数控制：

- 无-q 或-v：仅打印 CRITICAL，ERROR，WARNING 和 INFO 消息。
- -v：还打印 DEBUG 消息。
- -vv：还打印 TRACE 消息。
- -q：安静模式。不会打印任何记录消息。

使用 chilog（）而不是 printf（）将使你可以轻松控制日志记录中的详细程度，而无需添加和/或注释掉 printf（）语句。

六. 任务 1：基本消息处理

第一个任务是作为热身练习来重新认识套接字编程。你必须实现一个 IRC 服务器，该服务器能很好地实现 NICK 和 USER 消息以执行单个用户注册，如 IRC 通信示例中所示。满足这些要求，并通过大多数自动测试的简单服务器可以用大约 50 行 C 代码编写（事实上，我们将为你提供这 50 行代码）。虽然这样偷懒的解决方案将在测试中获得一些积分，但它将在设计等级上不得分。

因此，你对此任务的解决方案应满足以下要求：

- （A）必须在收到 NICK 和 USER 消息后才发送 RPL_WELCOME。
- （B）必须考虑到从套接字读取时可能会获得多于或少于一个完整的消息。你可能无法通过从套接字一次读取一个字符来解决此问题。
- （C）你的解决方案必须解析 NICK 和 USER 消息中的昵称和用户名，并撰写正确的 RPL_WELCOME 回复。

虽然此任务不需要，但你应该考虑到项目的其余两部分将涉及添加对其他消息和回复的支持。你花在编写消息解析器和构造函数（不仅仅是 NICK 和 USER）上的时间都是值得的。但是，如果你对此作业的解决方案采用一些便捷方式，不如只处理 NICK 和 USER 消息以及 RPL_WELCOME 回复也可以。

你的服务器必须用 C 实现，并且必须使用套接字。现在还不需要使用多线程。

七. 任务 2：支持多用户

在这一部分，你的主要目标是实现用户相互发送消息。你还将实现一些额外的消息，使你的服务器足够兼容和现有的 IRC 客户端进行测试。

由于你将支持多个用户，因此你现在必须为连接到服务器的每个用户生成一个新线程。相应的，这可能会导致代码中的条件竞争。你必须识别服务器中的共享资源，并确保它们受到足够的同步原语的保护。

你必须实现的内容在建议的实施顺序中显示。尽管如此，一旦你实现了连接注册，其余

的消息大多是相互独立的。

连接注册

连接注册的实现如[RFC2812 §3.1]中所述，但有以下例外：

- 你必须实现 NICK，USER 和 QUIT 消息。不得实现 PASS，SERVICE 或 SQUIT 消息。而且不需要实现 OPER 和 MODE 消息（将在下一个任务中实现它们）。
- 在 NICK 消息中，必须实现 ERR_NONICKNAMEGIVEN 和 ERR_NICKNAMEINUSE 回复。
- 可以忽略 USER 消息的<mode>和<unused>参数。
- 在 USER 消息中，必须实现 ERR_ALREADYREGISTRED 和 ERR_NEEDMOREPARAMS。**注意**：你需要在其他消息中支持 ERR_NEEDMOREPARAMS 回复。编写一个验证消息中参数数量的函数十分有用，如果参数数量不足，则返回 ERR_NEEDMOREPARAMS 回复。
- 注册连接后，RPL_WELCOME 回复后面必须跟着 RPL_YOURHOST，RPL_CREATED，RPL_MYINFO 回复（按此顺序）。对于 RPL_MYINFO 回复，用户模式为 ao，通道模式为 mtov。
- 在回复 QUIT 时发送的 ERROR 消息必须包含以下错误消息：

```
Closing Link: HOSTNAME (MSG)
```

其中 HOSTNAME 是用户的主机名，MSG`是 QUIT 消息中提供的<Quit Message>参数。 如果未提供，则默认为客户端退出。

还需考虑以下因素：

- 可以按任何顺序接收 NICK 和 USER 消息，并且在未收到两个消息之前连接不会完全注册（并且这两个消息都不包含任何错误）。
- 如果在连接注册完成之前收到除 NICK 或 USER 之外的任何消息，且如果该消息包含合法的命令，则必须发送 ERR_NOTREGISTERED 回复（比如我们要求你在此项目中实现的命令之一）。否则忽略该消息。请注意，一旦注册完成，此行为将发生变化（请参阅下面的 ERR_UNKNOWNCOMMAND）。
- 在连接注册完成后也可以使用 NICK 命令来更改用户的昵称。
- 你可以跳过 QUIT 命令并以后再重新处理它，因为没有其他命令依赖该命令。
- 发送欢迎消息后，大多数 IRC 服务器会发送对应于 MOTD 和 LUSER 消息的回复。大多数测试应都这样，但是在实现 MOTD 和 LUSER 之前，你可以通逐字发送以下回复来偷懒：

```
:hostname 251 user1 :There are 1 users and 0 services on 1 servers

:hostname 252 user1 0 :operator(s) online

:hostname 253 user1 0 :unknown connection(s)

:hostname 254 user1 0 :channels formed

:hostname 255 user1 :I have 1 clients and 1 servers

:hostname 422 user1 :MOTD File is missing
```

这足以通过连接注册的测试（这些信息检查是否发送了正确的回复，但实际上并未检查它们是否包含准确的信息）。

PRIVMSG 和 NOTICE

实现用户间的消息传递如[RFC2812 §3.3]中所述，但以下情况除外：

- 唯一支持的<msgtarget>是昵称。
- 必须实现 ERR_NORECIPIENT，ERR_NOTEXTTOSEND 和 ERR_NOSUCHNICK 回复。

考虑以下因素：

- 如果用户 user1 向 user2 发送一系列 PRIVMSG 消息，则 user2 必须按照 user1 发送的顺序接收。
- 如果用户 user1 和 user2 都 user3 发送单条消息，则消息不必以 user1 和 user2 发送的顺序到达。

PING 和 PONG

实现 PING 和 PONG 命令，如[RFC2812 §3.7.2]和[RFC2812 §3.7.3]中所述，但有以下例外：

- 可以忽略 PING 中的参数，只需将 PONG 的响应发送给发 PING 消息的客户端即可。
- 必须以默认删除收到的任何 PONG 消息（不要发送 ERR_UNKNOWNCOMMAND 回复）。

考虑以下因素：

- 实现 PING 和 PONG 对用真正的 IRC 客户端测试服务器至关重要。IRC 客户端将定期发送 PING 消息，如果他们没有收到 PONG 消息，他们会关闭连接。

MOTD

实现 MOTD 命令，如[RFC2812 §3.4.1]中所述，但有以下例外：

- 可以忽略<target>参数

考虑以下因素：

- 你的服务器应该从运行服务器的目录中的 motd.txt 中读取“当日消息”。
- 如果该文件不存在，则必须返回 ERR_NOMOTD 回复。

LUSERS

实现 LUSERS 命令，如[RFC2812§3.4.2]中所述，但有以下例外：

- 可以忽略<mask>和<target>参数。
- 必须按以下顺序发送回复：RPL_LUSERCLIENT，RPL_LUSEROP，RPL_LUSERUNKNOWN，RPL_LUSERCHANNELS，RPL_LUSERME。
- 不需要支持 ERR_NOSUCHSERVER 回复。

考虑以下因素：

- 即使报告零值，也必须发送回复（即，从[RFC2812§5.1]中忽略此条：“回复时，服务器必须发送回 RPL_LUSERCLIENT 和 RPL_LUSERME。其他回复仅在非零时发回。“）。

- 任何未完全注册的连接客户端是“unknown connection”（即，未成功发送 NICK 和 USER 的任何客户端）。
- RPL_LUSERCLIENT 回复中的用户数是已注册用户的数量（即，所有打开的连接减去未知连接）。
- RPL_LUSERME 回复中的客户端数是连接总数，包括未知连接。

WHOIS

实施 WHOIS 命令如[RFC2812 §3.6.2]所述，但以下情况除外：

- 该命令只接受一个参数：一个 nick（即，只有一个<mask>且必须是一个 nick，忽略<target>参数）。
- 通常，WHOIS 命令可以在没有参数的情况下使用，因此 RFC 在这种情况下不需要 ERR_NEEDMOREPARAMS 回复。 但是，由于我们不支持没有参数的 WHOIS，如果你收到这样的消息，直接忽略（即，根本不发回任何回复）。
- 只能按以下顺序发回这些回复： RPL_WHOISUSER， RPL_WHOISSERVER，RPL_ENDOFWHOIS。
- 必须为 RPL_WHOISSERVER 中的参数<server info>提供值，但不会检查其内容。
- 必须支持 ERR_NOSUCHNICK 回复。

考虑以下因素：

- 将在下一个任务中实现 RPL_WHOISOPERATOR， RPL_WHOISCHANNELS 和 RPL_AWAY。

ERR_UNKNOWNCOMMAND

如果在正确注册后，服务器收到上面未描述过的任何消息（或在下一个任务中），则必须返回 ERR_UNKNOWNCOMMAND 回复。

稳健性

你的代码必须通过“稳健性”测试（有关如何运行测试的说明，请参阅测试实施）。 这些测试检查你的代码在某些极端情况下是否会崩溃（例如，当使用长度为 511、512 或 513 字节的消息时），以及命令包含任意数量的空白时。这在 RFC 中没有规定，但大多数 IRC 服务器能够处理“稳健性”测试发送的消息类型。

八. 任务 3：频道和模式

在项目的这一部分，你的主要目标是增加对频道和模式的支持。你现在要处理这样的情况：消息可能被转发到多个用户，有时会跨多个频道。

该任务的部分内容按建议的实施顺序列出。尽管如此，一旦你实现了频道（包括 JOIN，PART 和向频道发送消息），实现模式和剩余的消息都相互独立。

Join

实现 JOIN 命令如[RFC2812 §3.2.1]中所述，但有以下例外：

- 该命令必须接受单个参数：频道名称。
- 只能支持 RPL_TOPIC，RPL_NAMREPLY 和 ERR_NEEDMOREPARAMS 回复。

考虑以下因素：

- 如果频道有主题的话，必须只发送 RPL_TOPIC 回复（稍后你将实施 TOPIC 消息）。否则，将跳过该回复。
- 虽然未在 [RFC2812 §3.2.1] 中明确说明，但 RPL_NAMREPLY 回复必须后跟 RPL_ENDOFNAMES。基本上，你发送的是在收到 NAMES 消息（参数为此频道）时生成的相同回复。
- JOIN 的第一个自动测试将检查是否已发送 RPL_NAMREPLY 和 RPL_ENDOFNAMES 回复，但不会验证其内容。因此，你只需发送以下回复（用收件人 nick 替换“nick”）就可以偷懒：

```
:hostname 353 nick = #foobar :foobar1 foobar2 foobar3

:hostname 366 nick #foobar :End of NAMES list
```

实现 NAMES 消息后，只需调用处理 NAMES 消息的相同代码即可。请注意，在正确实现 NAMES 之前，大多数 IRC 客户端会将你的频道显示为在 NAMES 回复中有三个用户（foobar1，foobar2 和 foobar3）。

PRIVMSG 和频道通知

扩展你之前的 PRIVMSG 和 NOTICE 实现，以支持向频道发送消息。

在实施模式前，不需要支持 PRIVMSG 中的任何其他回复。 但是，请考虑以下因素：

- 当指定不存在的通道时，ERR_NOSUCHNICK 也是适当的回复。
- 用户无法向未加入的频道发送 PRIVMSG 和 NOTICE 消息。发生这种情况时，必须发回 ERR_CANNOTSENDTOCHAN 回复（仅在 PRIVMSG 消息的情况下）。
- 实现模式后，可能存在其他情况：如果用户没有足够的权限在频道上发言，则将拒绝该消息。

PART

实现 PART 命令，如[RFC2812§3.2.2]中所述，但有以下例外：

- 该命令接受一个参数（频道名称）或两个参数（频道名称和 part 消息）。
- 只支持 ERR_NOTONCHANNEL，ERR_NOSUCHCHANNEL 和 ERR_NEEDMOREPARAMS 回复。

考虑以下因素：

- 一旦某个频道中的所有用户都离开了，就必须销毁该频道。

TOPIC

实现 TOPIC 命令，如[RFC2812 §3.2.4]中所述，但有以下例外：

- 只需要支持 ERR_NOTONCHANNEL，RPL_NOTOPIC，RPL_TOPIC 和 ERR_NEEDMOREPARAMS 回复。

- 在实现模式之前，不需要支持 ERR_CHANOPRIVSNEEDED 回复。

用户和频道模式

在 IRC 中，可以为用户分配某些模式。模式由单个字符标识，并且是二进制的：用户要么具有模式，要么不具有模式。[RFC2812 §3.1.5]中描述了可能的用户模式，我们将仅实现以下模式：

- a - 离开模式。具有此模式的用户被视为“离线”。除了在 IRC 客户端上显示之外，它还将影响如何处理指向用户的某些消息。
- o - 操作员模式。具有此模式的用户在 IRC 服务器上具有管理权限，并且可以访问仅供操作员使用的某些命令。

以上两种模式是全局模式：它们在整个服务器上都有效。用户还可以具有频道特定模式（或成员状态模式，请参阅[RFC2811 §4.1]）。我们将实现以下成员状态模式：

- o – 频道操作员模式。在频道上使用此模式的用户在该频道上具有特殊权限。
- v - 语音模式。具有此模式的用户可以将消息发送到仲裁频道（如下所述）。

最后，频道本身也可以有模式（参见[RFC2811 §4]）。 我们将实现以下模式：

- m - 审核模式。当频道具有此模式时，仅允许某些用户向频道发送消息。
- t - 主题模式。 当频道具有此模式时，只有频道操作员可以设置频道的主题。

使用 OPER，MODE 和 AWAY 命令管理这些模式。目前，我们将重点关注前两个。

你必须按照[RFC2812 §3.1.4]中的描述实现 OPER 消息，但以下情况除外：

- 只支持 RPL_YOUREOPER，ERR_PASSWDMISMATCH 和 ERR_NEEDMOREPARAMS。

你会用到<user>参数，但忽略其内容; OPER 命令需要的密码是 chirc 可执行文件的-o 命令行参数中指定的密码。

你必须按[RFC2812 §3.1.5]（针对用户模式）和[RFC2812§3.2.3]（针对成员状态和通道模式）中所述实现 MODE 消息，但以下情况除外：

- 对于用户模式：
  - 只需要支持两个（也就是两个）参数：nick 和模式字符串。模式字符串总是两个字符长：一个加号或减号字符，后跟一个字母。
  - 只需要支持 ERR_UMODEUNKNOWNFLAG 和 ERR_USERSDONTMATCH 回复。
  - 如果没有错误，则对 MODE 消息的回复将是消息的中继，以用户的 nick 为前缀，并在长参数中使用模式字符串。因此，如果用户发送此消息：

    ```
    MODE jrandom -o
    ```

    回复应该是：

    ```
    :jrandom MODE jrandom :-o
    ```

- 对于频道模式：
  - 当仅使用单个参数（频道名称）时，你必须支持的错误条件是 ERR_NOSUCHCHANNEL 回复（尽管这不包括在 MODE 的规范中）。 如果命令成功，则返回 RPL_CHANNELMODEIS 回复（在此回复中，<mode>参数必须是加号后跟频道模式;必须省略<mode params>参数）。
  - 使用两个参数（频道名称和模式字符串）时，必须支持以下错误答复：ERR_NOSUCHCHANNEL，ERR_CHANOPRIVSNEEDED 和

ERR_UNKNOWNMODE。如果命令成功，则将消息发回此用户和通道中的所有用户。

- 对于成员状态模式：
  - 只需要支持三个参数：频道，模式字符串和 nick。
  - 必须支持以下错误回复：ERR_NOSUCHCHANNEL, ERR_CHANOPRIVSNEEDED, ERR_UNKNOWNMODE 和 ERR_USERNOTINCHANNEL。
  - 如果命令成功，则将消息发回用户和通道中的所有用户。

在处理模式时，必须遵守以下规则：

- OPER 消息是用户获得操作员状态的唯一方式（o 用户模式）。如规范中所示，应忽非管理员的+ o 请求。
- 无法使用 MODE 命令切换用户模式。只有 AWAY 消息才能操作该模式。应该忽略更改它的请求。
- 创建频道时，第一个进入该频道的用户将自动被授予频道操作员模式。
- 在频道中，只有频道管理员员才能更改频道模式。
- 在频道中，只有频道管理员才能更改该频道中用户的成员状态模式。
- 当频道具有 m 模式时，只有具有 v 成员状态的用户和频道管理员才能向该频道发送 PRIVMSG 和 NOTICE 消息。其他用户将收到 ERR_CANNOTSENDTOCHAN 回复。
- 当频道具有 t 模式时，只有频道管理员可以更改频道的主题。其他用户将收到 ERR_CHANOPRIVSNEEDED 回复。
- 在许可方面，服务器管理员（具有 o 用户模式）具有与频道管理员相同的特权。但是，服务器管理员在加入频道时不会明确地具有 o 成员状态（用户将隐含地具有与频道管理员相同的特权）。

AWAY

实现 AWAY 命令，如[RFC2812 §4.1]中所述。

Names

实现 NAMES 命令，如[RFC2812 §3.2.5]中所述，但有以下例外：

- 我们不支持隐藏，私有或秘密频道，因此你可以认为发送 NAMES 命令的用户可以看到所有频道。
- 只需要支持没有参数或单个参数的 NAMES 消息。
  - 如果未指定任何参数，则必须为每个频道返回 RPL_NAMREPLY 回复。由于我们不支持不可见用户，因此最终的 RPL_NAMREPLY 必须包含不在任何频道上的所有用户的名称。如果所有已连接的用户都在一个通道中，则省略该最终的 RPL_NAMREPLY。
  - 指定单个参数时，该参数为一个频道。
- 不需要支持 ERR_TOOMANYMATCHES 和 ERR_NOSUCHSERVER 回复。

考虑以下因素：

- 频道和 nick 不需按任何特定顺序列出。
- 实现模式时，频道上具有频道管理员权限的 nick 必须在 RPL_NAMREPLY 回复中以@为前缀。具有“语音”权限的 nick 必须加上前缀+。

LIST

实现 LIST 命令，如[RFC2812 §3.2.6]中所述，但有以下例外：

- 只需要支持没有参数的 LIST 消息（列出所有频道）或单个参数（仅列出指定的频道）。
- 不需要支持 ERR_TOOMANYMATCHES 和 ERR_NOSUCHSERVER 回复。

考虑以下因素：

- 频道不需要按任何特定顺序列出。
- 在 RPL_LIST 回复中，<#visible>是指该频道上的用户总数（因为我们不支持不可见用户，可见用户数等于频道中的用户总数）。

WHO

实施 WHO 命令，如[RFC2812§3.6.1]所述，但以下情况除外：

- 如果指定了掩码，则只需支持掩码是频道名称的情况。如果存在此频通道，则必须为该频道中的每个用户返回 RPL_WHOREPLY。
- 我们不支持不可见客户端，因此，如果未指定掩码（或者指定 0 或*作为掩码），则必须为服务器中与请求客户端没有公共频道的每个用户返回 RPL_WHOREPLY。
- 不需要支持 o 参数。
- 不需要支持 ERR_NOSUCHSERVER 回复。

考虑以下因素：

- 未指定频道时，RPL_WHOREPLY 回复中的<channel>字段必须设置为*。
- 在 RPL_WHOREPLY 回复中，<hopcount>应为 0。
- RPL_WHOREPLY 必须返回一系列标志，指定为（“H”/“G”> [“*”] [（“@”/“+”）]没有解释（此外，>是一个拼写错误，并且 应该是一个右括号）。必须按照以下顺序构造标志：
  - 如果用户没有离开，请包含 H（“here”）。
  - 如果用户离开，请加 G（“gone”）。
  - 如果用户是管理员，请包含*。
  - 如果用户是频道管理员，请包含@。
  - 如果用户在频道中具有语音模式，请包括+。

  如果未指定频道，则不包括@和+标志（无论用户在其所属的用户组中有哪种频道模式）。

更新先前任务中的命令

更新以下命令的实现：

- NICK：当用户发送此消息并且成功更改 nick 时，必须将其转发到用户所在的所有频道。
- QUIT：当用户发送此消息时，必须将其转发给用户所在的所有频道。应考虑到 QUIT 意味着该用户离开他所在的所有频道。
- WHOIS：添加对 RPL_WHOISOPERATOR，RPL_WHOISCHANNELS 和 RPL_AWAY 回复的支持。这些回复仅在用户是 IRC 管理员、至少在一个频道上或在离开时发送。所有回复的顺序为：RPL_WHOISUSER，RPL_WHOISCHANNELS，RPL_WHOISSERVER，

RPL_AWAY，RPL_WHOISOPERATOR，RPL_ENDOFWHOIS。

- LUSERS：需要更新回复以显示正确的 IRCops 数和频道数。

九. 测试你的实现

请用以下两种方式测试你的实现：

1. 使用所提供的一组自动测试。这些提供了一种方便的机制来验证特定命令或命令集是否正确实现，但是交互式调试比较困难。
2. 手动登录你的 IRC 服务器。可以用交互方式测试你的实现。

使用自动测试

chirc 包含一套全面的自动测试，可以让你测试你的实现是否正确。要运行自动测试，只需运行以下命令：

```
make tests
```

这将调用一个名为 py.test 的测试工具，该测试工具将运行一系列单独的测试，并将提供正确运行次数和失败次数的摘要（包括所有失败的测试的输出和错误消息）。它还将生成一个名为 report.html 的 HTML 文件，其中包含测试结果的摘要。

运行类别测试

make 测试将运行所有测试（当你开始处理项目时几乎所有测试都会失败），输出可能不太有用。测试分为几个类别，对应于三个任务的部分，你可以只运行一个类别的测试：

```
TEST_ARGS="-C CATEGORY" make tests
```

CATEGORY 是你要运行的测试类别。所有类别如下：

- Assignment 1

```
  TEST_ARGS="-C BASIC_CONNECTION" make tests
```

- Assignment 2

```
TEST_ARGS="-C CONNECTION_REGISTRATION" make tests

TEST_ARGS="-C PRIVMSG_NOTICE" make tests

TEST_ARGS="-C PING_PONG" make tests

TEST_ARGS="-C MOTD" make tests

TEST_ARGS="-C LUSERS" make tests

TEST_ARGS="-C WHOIS" make tests
```

```
TEST_ARGS="-C ERR_UNKNOWN" make tests
```

```
TEST_ARGS="-C ROBUST" make tests
```

- Assignment 3

```
TEST_ARGS="-C CHANNEL_JOIN" make tests

TEST_ARGS="-C CHANNEL_PRIVMSG_NOTICE" make tests

TEST_ARGS="-C CHANNEL_PART" make tests

TEST_ARGS="-C CHANNEL_TOPIC" make tests

TEST_ARGS="-C MODES" make tests

TEST_ARGS="-C AWAY" make tests

TEST_ARGS="-C OPER" make tests

TEST_ARGS="-C NAMES" make tests

TEST_ARGS="-C LIST" make tests

TEST_ARGS="-C WHO" make tests

TEST_ARGS="-C UPDATE_ASSIGNMENT2" make tests
```

运行个别测试

如果你想专注于调试单个的失败测试，你还可以使用-k 参数运行单个测试：

```
TEST_ARGS="-k test_connect_both_messages_at_once" make tests
```

可以在测试输出中的 FAILURES 行之后找到每个失败测试的名称。 例如：

```
================================== FAILURES ========================

________ TestBasicConnection.test_connect_both_messages_at_once __________
```

运行单个测试时，有时查看客户端和服务器之间交换的确切消息会很有用。可以使用 tcpdump 和 Wireshark 等网络嗅探器。Wireshark 的控制台版本 tshark 可用于调试自动测试。考虑到像 Wireshark 这样的 tshark 需要特殊权限，因此你可能无法在学校的计算机上运行它，而必须在自己的计算机上运行。

要从单个测试中捕获网络流量，你需要手动运行 py.test（而不是 make tests）来强制测试使用特定的 TCP 端口。例如，要运行 test_connect_simple 测试：

```
py.test -k test_connect_simple --chirc-port=7776 --chirc-exe=./chirc
```

请注意，我们使用端口 7776 来避免与标准 IRC 端口冲突（6667）。

在一个单独的终端上，像这样运行 tshark：

```
tshark -i lo \

      -d tcp.port==7776,irc -R irc -V -O irc -T fields -e irc.request -e
 irc.response \ tcp port 7776
```

然后运行测试，tshark 应该打印出以下内容（假设是 chirc 的完整实现）：

```
NICK user1

USER user1 * * :User One

   :haddock 001 user1 :Welcome to the Internet Relay Network user1!user1@
localhost

   :haddock 002 user1 :Your host is haddock, running version chirc-0.3.9

   :haddock 003 user1 :This server was created 2016-01-03 10:46:01

   :haddock 004 user1 haddock chirc-0.3.9 ao mtov

   :haddock 251 user1 :There are 1 users and 0 services on 1 servers

   :haddock 252 user1 0 :operator(s) online

   :haddock 253 user1 0 :unknown connection(s)

   :haddock 254 user1 0 :channels formed

   :haddock 255 user1 :I have 1 clients and 1 servers

   :haddock 422 user1 :MOTD File is missing
```

考虑到自动测试在测试通过后立即关闭连接，这意味着有时一些消息不会被发送。例如，在这个特定的测试中，tshark 就不会被发送。

制作成绩报告

运行所有测试后，可以运行以下命令来生成已通过的测试数量的摘要，以及每个测试类别的得分：

```
make grade
```

注意：上面的命令只会在运行 make tests 后生成有意义的输出。

一个完整的 chirc 实现会产生如下总结：

```
       Assignment 1
=====================================================================
Category                          Passed / Total      Score  / Points
---------------------------------------------------------------------
Basic Connection                   15     / 15         50.00  / 50.00
---------------------------------------------------------------------
                                              TOTAL = 50.00  / 50
=====================================================================


Assignment 2
=====================================================================
Category                          Passed / Total      Score  / Points
---------------------------------------------------------------------
Connection Registration            5      / 5          35.00  / 35.00
PRIVMSG and NOTICE                 10     / 10         30.00  / 30.00
PING and PONG                      6      / 6          2.50   / 2.50
MOTD                               2      / 2          5.00   / 5.00
LUSERS                             7      / 7          10.00  / 10.00
WHOIS                              2      / 2          10.00  / 10.00
ERR_UNKNOWN                        3      / 3          2.50   / 2.50
Robustness                         9      / 9          5.00   / 5.00
---------------------------------------------------------------------
                                              TOTAL = 100.00 / 100
=====================================================================
```

```
Assignment 3
========================================================================
Category                          Passed / Total      Score  / Points
------------------------------------------------------------------------
JOIN                             5      / 5          15.00  / 15.00
PRIVMSG and NOTICE to channels     6      / 6          15.00  / 15.00
PART                             13     / 13         10.00  / 10.00
TOPIC                            10     / 10         10.00  / 10.00
User and channel modes             57     / 57         25.00  / 25.00
AWAY                             6      / 6          5.00   / 5.00
NAMES                            11     / 11         5.00   / 5.00
LIST                             5      / 5          5.00   / 5.00
WHO                              6      / 6          5.00   / 5.00
Update Assignment 2                5      / 5          5.00   / 5.00
------------------------------------------------------------------------
                                              TOTAL = 100.00 / 100
========================================================================
```

注意：分配给每个类别的分数可能与上面显示的不同。这些分数可由教师配置，教师可决定以不同方式分配分值。

手动登录 IRC 服务器

自动测试可以帮助你了解项目的哪些部分正常工作，以及哪些部分可能需要一些修改。但是，即使使用 tshark，调试也很麻烦，因为你受到测试执行（及检查）的特定操作的限制。

在服务器中调试特定问题时，可以使用标准 telnet 客户端手动连接到服务器，以交互方式调试。只需像这样运行你的服务器：

```
./chirc -o foobar -p 7776
```

像这样登录：

```
telnet localhost 7776
```

这提供了 IRC 协议的直接接口。因此，如果要注册为用户，你必须在 telnet 客户端中输

入以下内容：

```
NICK user1
```

按 Enter 键将发送\ r \ n 终止符。 接下来，输入：

```
USER user1 * * :User One
```

然后按 Enter 键。如果你的服务器实现正确，telnet 客户端将打印出服务器对 NICK 和 USER 命令发送的欢迎回复。一旦你以这种方式登录，你可以手动测试其他 IRC 命令。

你还可以使用现有的 IRC 客户端测试你的实现。我们建议使用 irssi（http://irssi.org/），它提供了一个简单的基于终端的界面。这将允许你与 IRC 协议进行更高级别的交互（另外，如果你的服务器与标准 IRC 客户端正常工作，这表明你的实现非常好）。但是，考虑到像 irssi 这样的客户端不允许你直接输入 IRC 命令（就像 telnet 会话允许的那样）。你将需要使用 IRC 客户端中定义的命令（IRC 客户端将转换为实际的 IRC 命令通过 TCP 连接发送到服务器）。

IRC 协议官方文档：https://tools.ietf.org/html/rfc2812#