server:

1: 客户端应该首先建立与服务器的 TCP 连接。

2: 假设连接成功，客户机应该提示用户输入用户名。用户名应该发送到服务器。

3: 服务器应该检查凭据文件(credentials.txt)是否匹配。

如果用户名存在，服务器将向客户端发送确认消息。

4: 客户端提示用户输入密码。

5: 密码被发送到服务器，

服务器检查与存储的密码是否匹配。

如果密码匹配，服务器发送确认信息;

如果密码不匹配，则发送错误消息。

6: 向用户显示适当的消息(欢迎或错误)。

7: 如果不匹配，则要求客户端再次输入密码(请参阅后面关于阻止的讨论)。

如果用户名不存在，则假设用户正在创建一个新帐户，服务器将向客户端发送适当的消息。

客户端提示用户输入新密码。您可以假设密码格式如上所述(不需要检查)。

密码被发送到服务器。服务器在凭据文件中创建一个新的用户名和密码条目(将其作为最后一个条目追加)。

向客户端发送确认信息。客户端向用户显示适当的欢迎消息。

您应该确保为 credentials.txt 文件启用了写权限(在服务器当前工作目录的终端输入“chmod +w credentials.txt”)。

身份验证成功后，客户端被视为已登录(即在线)。

如上所述，在输入无效密码时，会提示用户重试。在对特定用户名连续尝试 3 次失败后，该用户将被阻塞一段持续时间(阻塞持续时间是提供给服务器的命令行参数)，并且在此期间无法登录。在这个实例中，客户端应该退出。

设计:

Timeout - 服务器应该检查所有登录的用户都是活动的。如果服务器检测到用户在一段超时时间内(timeout 是提供给服务器的命令行参数)没有发出任何有效的命令来与服务器进行交互或进行点对点消息传递，那么服务器应该自动将该用户注销。收到消息或输入无效命令不计算在内。有几种方法可以实现超时机制。我们将把设计留给你。

Presence Broadcasts - 服务器应该通知其他登录到服务器的用户的存在/缺席，即，当用户登录和退出时向所有在线用户发送广播通知。

List of online users - 服务器应该提供当前在线的用户列表，以响应用户的此类查询。

Online history - 服务器应该提供在过去某一用户指定时间内登录的用户列表(例如，在过去 15 分钟内登录的用户)。

Message Forwarding - 服务器应该将每个即时消息转发给正确的收件人，假设他们是在线的。

Offline Messaging --当消息的接收者没有登录(即脱机)，消息将被服务器保存。当收件人下一次登录时，服务器将发送为该用户存储的所有未读消息(不需要时间戳)

Message Broadcast——服务器应该允许用户向所有在线用户广播消息。

Blacklisting - 服务器应该允许一个用户阻止/解除阻止任何其他用户。例如，如果用户 A 阻止了用户 B，用户 B 就不能再向用户 A 发送消息，即服务器应该拦截这些消息，并通知用户 B 该消息不能转发。被屏蔽的用户也不会收到到场通知。每次 A 登录或退出时，不会通知 B。被屏蔽用户也无法查看被屏蔽用户的在线状态，即。B 将不能看到 A 当前或过去是否在线(即在线历史)。

在标记时，服务器将首先执行，并在整个测试期间保持在线。我们不会突然终止服务器或客户端进程(CTRL-C)。服务器不需要维护以前执行的状态。在执行时，假定没有用户是活动的。