目录

[一、实验要求 2](#_Toc156228856)

[二、协议原理 2](#_Toc156228857)

[2.1 Ping工作原理 2](#_Toc156228858)

[2.2 Ping命令作用 3](#_Toc156228859)

[2.3 IP协议 3](#_Toc156228860)

[2.4 IP报文格式 5](#_Toc156228861)

[2.5 ICMP协议 6](#_Toc156228862)

[2.6 ICMP协议的作用 8](#_Toc156228863)

[2.6 IP协议与ICMP协议 8](#_Toc156228864)

[2.7 套接字编程 9](#_Toc156228865)

[三、程序功能与流程 10](#_Toc156228866)

[3.1 ping命令程序功能 10](#_Toc156228867)

[3.2  使用ping命令常见问题 12](#_Toc156228868)

[3.3 ping程序开发流程 12](#_Toc156228869)

[四、分析程序代码 13](#_Toc156228870)

[4.1 导入python库 13](#_Toc156228871)

[4.2 构建GUI画面 14](#_Toc156228872)

[4.3 调用Ping方法 16](#_Toc156228873)

[4.4 connect()函数 20](#_Toc156228874)

[4.5 checksum()函数 21](#_Toc156228875)

[4.6 request()函数 21](#_Toc156228876)

[4.7 reply()函数 22](#_Toc156228877)

[五、总结 22](#_Toc156228878)

实验报告

# 一、实验要求

1. 基本要求：理解 ping 工作原理，编程实现 ping 程序。

2. 设计语言：Python、C/C++。

3. 原理：发送 ICMP 的回送请求报文，接收 ICMP 报文，并分析该报文（可能收到差错报告报文，也可能正常收到 ICMP 回送回答报文）。

4. 技术难点：构造 ICMP 回送请求报文并封装到 IP 数据报中，将该 IP 数据报发送出去并分析收到的结果，输出结果中一些统计信息的计算与呈现（图形化界面）。带有不同参数运行程序，其结果不同，即实现 windows 中 ping 命令部分选项功能，例如，发送 ICMP 报文数量、IP 中 TTL 设置等。

# 二、协议原理

## 2.1 Ping工作原理

ping命令是一种常用的网络诊断工具，可用于测试主机之间的连通性和测量往返时间（RTT）。您可以在命令行界面中使用ping命令，并指定目标主机的IP地址或域名来执行以下操作：

1. 发送ICMP Echo请求：当您在命令行中输入ping命令并指定目标主机的IP地址或域名时，您的计算机将创建一个Internet控制消息协议（ICMP）Echo请求数据包，并将其发送到目标主机。

2. 目标主机响应：目标主机接收到ICMP Echo请求后，将生成一个ICMP Echo回复数据包，并将其发送回源主机。

3. 测量往返时间（RTT）：源主机接收到ICMP Echo回复后，将记录下往返时间（RTT），即从发送请求到接收回复所经过的时间。

4. 显示结果：Ping命令将显示目标主机的响应时间和成功率。通常，它还会显示往返时间的统计信息，如最小、最大和平均往返时间。

值得注意的是，Ping命令使用ICMP协议来进行通信。ICMP是在互联网协议（IP）层上运行的协议，它用于在IP网络上传输有关网络状况和错误的信息。Ping命令利用ICMP Echo请求和回复消息来测试主机之间的连通性。

Ping命令的工作原理有助于确定主机是否可达、网络是否正常运行以及网络延迟的情况。它是一种简单而有效的网络诊断工具，广泛用于网络故障排除和性能测试。

## 2.2 Ping命令作用

1. 检测主机连通性：通过发送ICMP Echo请求消息到目标主机，ping命令可以确定目标主机是否可达。如果目标主机响应请求并发送回ICMP Echo回复消息，那么ping命令将显示成功的结果。如果目标主机未响应，ping命令将显示超时错误。

2. 测量往返时间（RTT）：ping命令还可以测量从发送请求到接收回复所经过的时间，即往返时间（RTT）。它会发送多个ICMP Echo请求，并记录每个请求的往返时间。通常，ping命令会显示每个请求的RTT，并提供有关最小、最大和平均RTT的统计信息。

3. 检测数据包丢失：ping命令还可以检测数据包丢失的情况。如果目标主机未能响应所有的ICMP Echo请求，ping命令将显示丢失的数据包百分比。这可以帮助确定网络中存在的丢包问题。

4. 追踪路由路径：在某些操作系统上，ping命令还提供了一种追踪路由路径的功能，称为"traceroute"或"tracert"。它通过发送一系列的ICMP Echo请求，逐跳地追踪到目标主机的路径。每个中间路由器都会响应一个ICMP回复，显示其IP地址和往返时间，从而帮助您确定数据包传输的路径。

总而言之，ping命令是一种简单而有用的网络诊断工具，可用于测试主机之间的连通性、测量往返时间和检测数据包丢失。它可以帮助网络管理员诊断和排除网络问题，并提供有关网络性能的基本信息。

## 2.3 IP协议

IP协议（Internet Protocol）是互联网上最基础和最核心的网络协议之一。它定义了在网络中进行数据传输的标准规范。IP协议负责将数据分割成称为数据包（或数据报）的小块，并在网络中进行路由和传输。

以下是IP协议的一些关键特点：

1. 数据包格式：IP协议定义了数据包的格式，包括头部和有效载荷。IP头部包含了源IP地址和目标IP地址，以及其他用于数据传输和路由的控制信息。有效载荷是要传输的实际数据。

2. 数据包路由：IP协议使用IP地址来标识和定位主机和网络。每个设备在网络中都有一个唯一的IP地址。当数据包从源主机发送到目标主机时，IP协议使用路由表来确定数据包的下一个跳，并通过网络中的路由器进行转发，直到达到目标主机。

3. 分组和重组：IP协议将数据分割成小块，称为数据包。这种分割允许大量的数据在网络中进行分段传输，并在接收端重新组装为完整的数据。这种分组和重组的机制使得IP协议能够适应各种网络环境和传输介质。

4. 无连接性：IP协议是一种无连接的协议，这意味着每个数据包都是独立传输的，没有与之相关的状态信息。每个数据包在传输时都是独立决策的，这使得网络更加灵活和可扩展。

5. 不可靠性：IP协议本身是不可靠的，它不提供数据包传输的可靠性保证。如果在传输过程中发生数据包丢失、错误或乱序，IP协议不会自动进行重传或纠错。这种可靠性需要在更高层的协议（如TCP）中实现。

IP协议是互联网的核心协议之一，它为互联网上的各种应用和服务提供了基础的数据传输能力。它的灵活性和可扩展性使得互联网能够支持大规模的数据通信，并连接了全球各种类型的设备和网络。

当涉及到IP协议时，还有一些重要的概念和特性需要了解：

1. IP地址版本：目前广泛使用的IP协议有两个主要版本，即IPv4和IPv6。IPv4使用32位地址表示，通常以四个十进制数（例如192.168.0.1）的形式表示。而IPv6采用128位地址，通常以八组16进制数（例如2001:0db8:85a3:0000:0000:8a2e:0370:7334）的形式表示。由于IPv4地址空间的枯竭问题，IPv6被设计为IPv4的后继版本，以提供更大的地址空间。

2. IP数据包分片：当数据包的大小超过网络上的最大传输单元（MTU）时，IP协议会对数据包进行分片。分片是将一个大的数据包分割成更小的片段，以适应网络传输。接收端的主机会重新组装这些分片，以还原原始的数据包。分片和重组是IP协议中的重要机制，但也会增加网络延迟和处理负担。

3. IP地址分类：在IPv4中，IP地址根据其网络部分和主机部分的位数分为不同的类别。最常见的分类是A类、B类和C类地址。每个类别具有不同的网络位数和主机位数，用于满足不同规模和需求的网络。然而，由于IP地址的有限性和地址分配的不均衡，CIDR（无类别域间路由）的引入使得地址分类的重要性逐渐减弱。

4. 子网划分和子网掩码：为了更好地管理IP地址和划分网络，子网划分的概念被引入。子网划分允许将一个大的IP网络划分为多个较小的子网，每个子网具有自己的网络标识和主机范围。子网掩码是用于确定IP地址中哪些位是网络部分和主机部分的一种掩码。它与IP地址进行逻辑与运算，以确定网络标识和主机标识。

5. 路由和路由表：IP协议使用路由表来决定数据包的下一跳。路由表是一个网络设备（如路由器）上的表格，其中包含了目标网络和相应的下一跳信息。当数据包到达路由器时，它会根据目标IP地址和路由表进行路由转发，将数据包发送到正确的目标网络。

6. 网络地址转换（NAT）：由于IPv4地址的有限性，网络地址转换（NAT）成为一种常见的技术，用于将私有IP地址转换为公共IP地址。NAT允许多个设备共享单个公共IP地址，通过修改IP数据包中的源IP地址和目标IP地址，以实现内部网络和外部网络之间的通信。

这些是IP协议的一些关键概念和特性。理解这些概念有助于更好地理解IP协议的工作原理和网络通信的基础知识。

7. IP包的生存时间（TTL）：IP数据包中的头部字段之一是TTL字段（Time to Live），它表示数据包在网络中允许存在的最长时间（以秒为单位）。每当数据包经过一个路由器时，TTL值都会减少。当TTL值达到0时，数据包将被丢弃，并向源主机发送一个ICMP超时消息。TTL的目的是防止数据包在网络中无限循环。

8 IP多播（Multicast）：IP协议支持多播通信，它使得一台主机可以将数据包发送给一组特定的目标主机，而不是所有的主机。多播允许有效地在网络中传输流媒体、视频会议和分布式应用等。多播使用特殊的IP地址范围（例如224.0.0.0到239.255.255.255）来标识多播组。

9. IP安全性：IP协议本身没有提供加密和安全性保护。然而，其他协议如IPsec（IP Security）可以在IP层提供加密、认证和完整性校验等安全功能。IPsec可以用于建立虚拟专用网络（VPN）连接，以保护数据在公共网络上的安全传输。

10. IPv6的优点：IPv6相对于IPv4具有一些优点。首先，IPv6地址空间更大，可以提供更多的IP地址，解决了IPv4地址枯竭的问题。其次，IPv6在协议设计中考虑了安全性和QoS（Quality of Service）等方面的改进。此外，IPv6还具有内置的支持IP多播和移动性的特性。

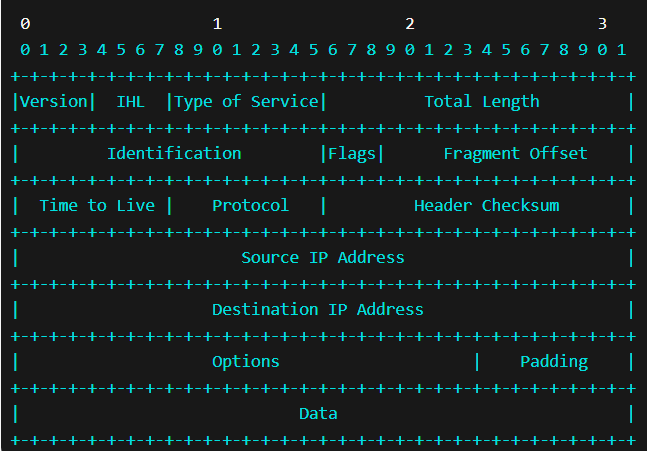
11. IP协议与传输层协议：IP协议位于网络层，而传输层协议如TCP（Transmission Control Protocol）和UDP（User Datagram Protocol）位于IP协议之上。TCP提供面向连接的、可靠的数据传输，而UDP则提供面向无连接的、不可靠的数据传输。IP协议负责将数据包从源主机传递到目标主机，而传输层协议负责在源主机和目标主机之间建立通信会话。

12. IP协议的扩展：IP协议支持各种扩展，例如IP选项字段和IP协议扩展头部。IP选项字段允许在IP头部中添加额外的信息，如时间戳、记录路由和安全性选项。IP协议扩展头部允许在IP数据包中添加额外的协议头部，以支持特定的应用需求。

这些是有关IP协议的一些补充信息。IP协议是构建互联网的基础，它提供了可靠的数据传输和网络连接的基本功能。理解IP协议的工作原理和相关特性对于网络管理和故障排除非常重要。

## 2.4 IP报文格式

IP（Internet Protocol）报文是在网络层使用的数据包格式，用于在互联网上进行数据传输。IP报文格式定义了数据包的结构和字段的含义。下面是对IP报文格式的详细解释：IP报文的基本结构如下：



以下是每个字段的解释：

1. Version（4 bits）：指定IP协议的版本号。IPv4的版本号为4，IPv6的版本号为6。

2. IHL（Internet Header Length）（4 bits）：指定IP报文头部的长度，以32位字长为单位。由于IPv4报文头部的长度是可变的，因此此字段指示报文头部占用的32位字长数。

3. Type of Service（8 bits）：指定服务质量和差异化服务（Quality of Service, QoS）相关的标志和参数。它可以用于指定数据包的优先级、延迟、吞吐量和可靠性需求。

4. Total Length（16 bits）：指定整个IP报文的长度，包括报文头部和数据部分。以字节为单位。

5. Identification（16 bits）：在数据包进行分片时，用于唯一标识原始数据报的标识符。在分片重组中使用。

6. Flags（3 bits）：用于指示分片的状态。包括 "DF"（Don't Fragment，不分片）和 "MF"（More Fragments，还有更多分片）标志。

7. Fragment Offset（13 bits）：在分片重组中用于指示分片在原始数据报中的位置。

## 2.5 ICMP协议

ICMP（Internet Control Message Protocol）是一种与IP协议紧密相关的网络协议。它用于在IP网络中传递控制消息和错误报告。ICMP的主要功能是提供网络诊断、错误检测和报告功能，以及支持网络设备之间的通信。

以下是ICMP协议的一些关键特点：

1. 错误报告：ICMP协议用于报告在IP网络中发生的错误情况。例如，当一个数据包无法到达目的地时，路由器或目标主机可以使用ICMP消息向源主机发送相应的错误报告。这些错误报告可以包括目标不可达、超时、重定向等。

2. Ping和Traceroute：ICMP协议提供了执行网络诊断和测量的工具，如Ping和Traceroute。Ping用于测试与目标主机之间的连通性，发送ICMP回显请求并等待回复。Traceroute用于确定数据包从源主机到目标主机经过的路由路径，发送一系列的ICMP探测报文并检测每个报文的回应时间。

3. MTU探测：ICMP协议还用于进行最大传输单元（MTU）探测。MTU是网络链路上能够传输的最大数据包大小。当数据包的大小超过链路的MTU时，数据包将被分片。ICMP协议中的路径MTU探测机制允许主机发现沿途链路的最小MTU，以便发送合适大小的数据包。

4. ICMP消息类型：ICMP协议定义了不同类型的消息，每个消息都有特定的目的和功能。一些常见的ICMP消息类型包括回显请求和回显应答（用于Ping操作）、目标不可达消息（用于指示数据包无法到达目标）、超时消息（用于指示数据包在传输过程中超时）等。

5. ICMP重定向：ICMP协议还支持重定向消息，用于告知主机发送数据包时更好的下一跳路由。当一个主机发送数据包到一个错误的下一跳路由时，路由器可以发送ICMP重定向消息给源主机，提供一个更好的下一跳路由选择。

ICMP协议在网络中起着重要的作用，它为网络设备之间的通信提供了控制和错误报告的机制。通过使用ICMP，网络管理员可以进行网络故障排除、性能测量和连通性测试。

当涉及ICMP协议时，以下是一些进一步的信息：

1. ICMP消息格式：ICMP消息由包含固定格式的报头和可选的数据字段组成。报头包括类型字段、代码字段、校验和字段等。类型字段指示消息的类型，代码字段提供更详细的信息，校验和字段用于检测消息的完整性。

2. ICMP消息类型：ICMP定义了多种消息类型，每种类型都有不同的目的和功能。一些常见的ICMP消息类型包括：

- 回显请求和回显应答（Echo Request和Echo Reply）：用于执行Ping操作，测试与目标主机之间的连通性。

- 目标不可达（Destination Unreachable）：用于报告数据包无法到达目标的原因，如网络不可达、主机不可达等。

- 超时（Time Exceeded）：用于指示数据包在传输过程中超时，常见的有传输超时和生存时间超时。

- 重定向（Redirect）：用于告知主机发送数据包时更好的下一跳路由。

3. Ping工具：Ping是基于ICMP协议的一种常用工具，用于测试与目标主机之间的连通性。Ping发送ICMP回显请求消息到目标主机，并等待目标主机返回ICMP回显应答消息。通过检查回显应答消息，可以确定是否成功与目标主机进行通信。

4. Traceroute工具：Traceroute（或称为tracert）是另一种基于ICMP协议的工具，用于确定数据包从源主机到目标主机经过的路由路径。Traceroute发送一系列的ICMP探测报文，每个报文具有不同的生存时间（TTL）值。每个报文在经过一个路由器时，该路由器会返回一个ICMP超时消息，从而确定路由路径。

5. ICMP错误报告：当IP数据包在传输过程中发生错误时，接收方的设备会生成一个相应的ICMP错误报告消息，并将其发送回数据包的源主机。这些错误报告提供了有关数据包无法到达目标或发生其他问题的信息，帮助进行故障排除和网络问题的诊断。

6. ICMP攻击：由于ICMP协议的开放性，它也可能成为网络攻击的目标。一些常见的ICMP攻击类型包括洪水攻击（如ICMP洪水攻击）、Ping洪水攻击和ICMP重定向攻击。这些攻击利用ICMP消息的特性来使网络设备过载、降低网络性能或导致服务不可用。

## 2.6 ICMP协议的作用

ICMP协议在网络中具有重要的作用，它提供了网络设备之间的控制消息传递、错误报告和网络诊断功能。理解ICMP协议对于网络管理和故障排除是至关重要的。

ICMP协议的主要作用是提供网络设备之间的控制消息传递、错误报告和网络诊断功能。以下是一些关键的作用：

1. 控制消息传递：ICMP协议允许网络设备之间传递控制消息。这些消息用于通信和协调网络设备的操作。例如，当一个路由器无法传递一个数据包时，它可以使用ICMP消息向源主机发送目标不可达的错误报告，以便源主机采取适当的措施。

2. 错误报告：ICMP协议用于报告在IP网络中发生的错误情况。当数据包无法到达目的地或发生其他问题时，目标主机或路由器可以使用ICMP消息向源主机发送相应的错误报告。这些错误报告可以包括目标不可达、超时、重定向等，帮助诊断网络故障和问题。

3. 网络诊断：ICMP协议提供了用于网络诊断的工具和功能。例如，Ping是基于ICMP的工具，可以测试与目标主机之间的连通性。通过发送ICMP回显请求并等待回复，Ping可以确定是否成功与目标主机进行通信。Traceroute是另一个基于ICMP的工具，用于确定数据包从源主机到目标主机经过的路由路径。

4. MTU探测：ICMP协议支持最大传输单元（MTU）探测。MTU是网络链路上能够传输的最大数据包大小。当数据包的大小超过链路的MTU时，数据包将被分片。ICMP协议中的路径MTU探测机制允许主机发现沿途链路的最小MTU，以便发送适当大小的数据包。

总而言之，ICMP协议在网络中起着重要的作用，它提供了网络设备之间的控制消息传递、错误报告和网络诊断功能。通过使用ICMP，网络管理员可以进行网络故障排除、性能测量和连通性测试，确保网络的稳定运行。

## 2.6 IP协议与ICMP协议

ICMP（Internet Control Message Protocol）是与IP协议紧密相关的协议，它在IP网络中传递控制消息和错误报告。它通过在IP数据包的数据部分中封装特定的ICMP消息来实现通信。

ICMP协议的主要目的是提供网络设备之间的通信和网络诊断功能。它可以用于报告网络中发生的错误情况，如目标不可达、超时、重定向等。ICMP还支持Ping和Traceroute等工具，用于测试与目标主机之间的连通性和确定数据包的路径。

ICMP消息是在IP数据包的数据部分中封装的，它们包含特定的类型和代码字段，用于指示消息的目的和功能。常见的ICMP消息类型包括回显请求和回显应答（用于执行Ping操作）、目标不可达消息（用于指示数据包无法到达目标）、超时消息（用于指示数据包在传输过程中超时）等。

总而言之，ICMP协议与IP协议密切相关，它提供了网络设备之间的控制消息和错误报告功能，用于网络诊断和故障排除。

## 2.7 套接字编程

套接字编程通常用于实现网络应用程序，包括客户端和服务器端。下面是套接字编程的基本步骤：

1. 创建套接字：首先，需要创建一个套接字对象。在大多数编程语言中，都提供了相应的套接字库或模块，可以使用这些库或模块创建套接字对象。

2. 绑定地址和端口：服务器端需要绑定一个特定的地址和端口，以便客户端能够连接到它。地址可以是IP地址或主机名，端口是一个数字，用于标识服务器上的特定服务。客户端也可以选择绑定一个本地地址和端口。

3. 监听连接：服务器端调用一个监听函数，开始监听来自客户端的连接请求。这样，服务器就可以接受客户端的连接并进行处理。

4. 接受连接：当服务器端接收到来自客户端的连接请求时，它调用一个接受连接的函数，建立与客户端之间的连接。这样，服务器和客户端之间就可以进行数据交换。

5. 数据交换：一旦建立了连接，服务器和客户端之间可以通过套接字发送和接收数据。可以使用读取和写入操作来进行数据交换。

6. 关闭连接：当通信结束时，服务器和客户端都可以调用关闭连接的函数，关闭套接字连接。这样，释放网络资源并结束通信。

套接字编程提供了一种灵活的方式来实现网络通信。不同的编程语言和操作系统提供了各种套接字库和API，例如Python的socket库、Java的Socket类、C语言的socket函数等。开发人员可以根据自己的需求选择适合的套接字库和函数来实现网络应用程序。

当涉及套接字编程时，以下是一些进一步的信息：

1. 套接字类型：套接字可以分为两种类型：流套接字（Stream Socket）和数据报套接字（Datagram Socket）。

- 流套接字（也称为面向连接的套接字）提供可靠的、基于字节流的通信。它们使用TCP（传输控制协议）作为传输协议，确保数据的可靠传输，但可能引入一定的延迟。

- 数据报套接字提供不可靠的、无连接的通信。它们使用UDP（用户数据报协议）作为传输协议，适用于需要低延迟和较少数据验证的应用，但传输过程中可能丢失或重复数据包。

2. 客户端和服务器端：套接字编程通常涉及客户端和服务器端之间的通信。客户端是发起连接的一方，它连接到服务器并发送请求。服务器端则接受来自客户端的连接请求，并提供相应的服务。

3. IP地址和端口号：在套接字编程中，IP地址用于标识网络中的主机，而端口号用于标识主机上的特定服务。IP地址由四个数字组成，用点分十进制表示（例如，192.168.0.1），而端口号是一个16位的数字（范围从0到65535）。

4. 阻塞和非阻塞模式：套接字可以在阻塞模式和非阻塞模式下运行。在阻塞模式下，套接字操作将一直阻塞（即暂停执行），直到操作完成或发生错误。在非阻塞模式下，套接字操作将立即返回，无论操作是否完成。非阻塞模式可以使程序同时处理多个套接字连接或执行其他任务。

5. 错误处理：在套接字编程中，可能会出现各种错误情况，例如连接失败、数据包丢失、超时等。程序员需要适当地处理这些错误，以确保网络通信的可靠性和稳定性。错误处理通常涉及检查返回的错误代码或异常，并采取适当的措施，例如重新连接、重新发送数据或记录错误日志。

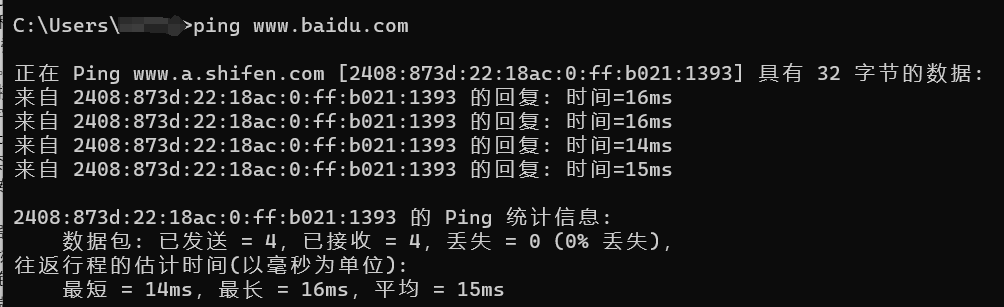
6. 安全性考虑：在套接字编程中，网络安全性是一个重要的考虑因素。通过使用加密和身份验证技术，可以确保通信的机密性和完整性。常见的安全协议包括TLS（传输层安全）和SSL（安全套接字层），它们提供了加密和认证的功能。

套接字编程是实现网络通信的重要技术，它使开发人员能够创建各种网络应用程序，包括Web服务器、聊天程序、文件传输工具等。了解套接字编程的基本概念和技术是进行网络应用程序开发的关键。

# 三、程序功能与流程

## 3.1 ping命令程序功能

ping [www.baidu.com](http://www.baidu.com) 连接百度，测试本机与指定网站服务器之间的网络连通性



这里面的几个参数：

* icmp\_seq=1 　ping序列，从1开始；
* bytes值： 数据包大小，也就是字节。
* time值：响应时间，这个时间越小，说明你连接这个地址速度越快。
* TTL值：Time To Live,表示DNS记录在DNS服务器上存在的时间，它是IP协议包的一个值，告诉路由器该数据包何时需要被丢弃。

TTL就是说ping的数 据包 能在网络上存在多少时间。当我们对网络上的主机进行ping操作的时候，我们本地机器会发出一个数据包，数据包经过一定数量的路由器传送到目的主机，但是由于很多的原因，一些数据包不能正常传送到目的主机，那如果不给这些数据包一个生存时间的话，这些数据包会一直在网络上传送，导致网络开销的增大。当数据包传送到一个路由器之后，TTL就自动减1，如果减到0了还是没有传送到目的主机，那么就自动丢失。

Linux中，有如下语法：

ping [-dfnqrRv][-c<完成次数>][-i<间隔秒数>][-I<网络界面>][-l<前置载入>][-p<范本样式>][-s<数据包大小>][-t<存活数值>][主机名称或IP地址]

**参数说明**：

-d 使用Socket的SO\_DEBUG功能。

-c <完成次数> 设置完成要求回应的次数。

-f 极限检测。

-i<间隔秒数> 指定收发信息的间隔时间。

-I<网络界面> 使用指定的网络接口送出数据包。

-l<前置载入> 设置在送出要求信息之前，先行发出的数据包。

-n 只输出数值。

-p<范本样式> 设置填满数据包的范本样式。

-q 不显示指令执行过程，开头和结尾的相关信息除外。

-r 忽略普通的Routing Table，直接将数据包送到远端主机上。

-R 记录路由过程。

-s<数据包大小> 设置数据包的大小。

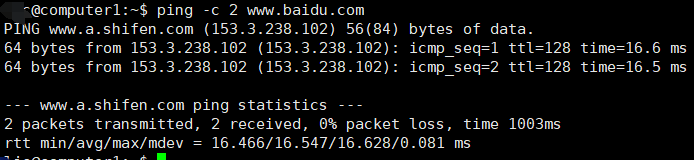
-t<存活数值> 设置存活数值TTL的大小。

-v 详细显示指令的执行过程。

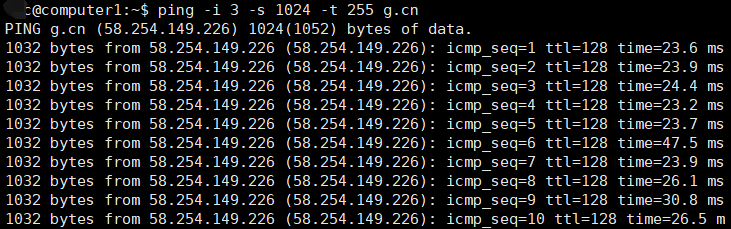
-w <deadline> 在 deadline 毫秒后退出。

-W <timeout> 在等待 timeout 毫秒后开始执行。

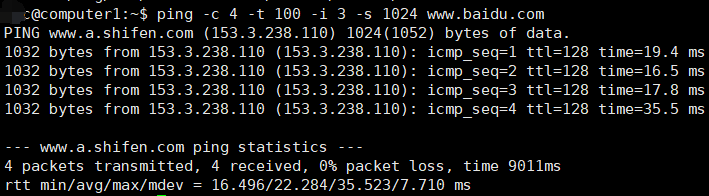
如：ping -c 2 [www.baidu.com](http://www.baidu.com) //-c 指定接收包的次数



如：ping -i 3 -s 1024 -t 255 g.cn //ping主机多参数使用



ping 一个主机名为 baidu.com 的主机，发送 4 个回声请求消息，可以使用以下命令。ping -c 4 192.168.0.1



我们之后进行程序编写实现上述命令程序功能。

## 3.2  使用ping命令常见问题

在ping的过程中，经常会遇到以下的情况，我们根据情况的反馈来判断具体的问题。

* No Answer: ，对方主机没工作，双方网络配置不正确，路由问题等。
* Request Time Out: 对方主机已关机，路由问题或对端防火墙设置禁止ping
* Destination Net Unreachable: 双方没有建立连接，或对方主机不存在
* Unknown Host Name: DNS设置问题，或者对方主机不存在
* transmit failed，error code: 网卡驱动问题
* Bad IP Address: IP地址不存在或IP不能被DNS服务器解析
* pining 127.0.0.1 如果ping不通，表明本地机TCP/IP协议不能正常工作
* no rout to host 网卡工作不正常

使用ping来辅助判断网络的连通性

## 3.3 ping程序开发流程

创建套接字：使用套接字API创建一个套接字，指定IP协议和ICMP协议。

构造Ping消息：创建ICMP Echo请求消息，包括目标IP地址、标识符（用于匹配请求和响应）和序列号（用于标识每个请求的顺序）等字段。

发送Ping消息：使用套接字发送构造的Ping消息到目标主机的IP地址。

接收响应消息：等待目标主机发送的ICMP Echo响应消息。使用套接字接收和解析响应消息，验证标识符和序列号是否匹配。

最后，使用python中的tkinter库的GUI图像画出我们的界面，使得可以让我们进行人机交互

分析结果：根据接收到的响应消息，计算并显示Ping的结果，包括往返时间（Round-Trip Time，RTT）和丢包率等统计信息。

# 四、分析程序代码

## 4.1 导入python库

这三个库的作用如下：

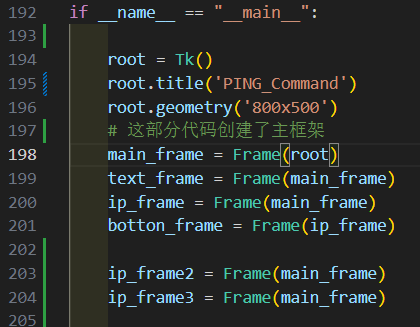
1. `struct` 库：`struct` 是 Python 标准库中的一个模块，用于处理二进制数据和 C 结构体之间的转换。它提供了一组函数，用于将数据打包成二进制字符串（`pack()` 函数）或从二进制字符串中解包数据（`unpack()` 函数）。`struct` 库通常用于与底层的二进制数据进行交互，例如在网络编程、文件处理或与其他语言交互时。它提供了一种便捷的方式来处理不同字节顺序、数据类型和对齐方式的数据。

2. `socket` 库：`socket` 是 Python 标准库中的一个模块，用于进行网络编程。它提供了一组函数和类，用于创建和操作网络套接字。通过 `socket` 库，你可以在 Python 中创建客户端和服务器应用程序，进行网络通信，发送和接收数据。它支持各种网络协议，包括 TCP、UDP 和原始套接字。`socket` 库提供了一种跨平台的方式来处理网络操作，使得编写网络应用程序变得更加简单和便捷。

3. `select` 库：`select` 是 Python 标准库中的一个模块，用于多路复用 I/O 操作。它提供了一种机制，使得可以同时监视多个文件描述符（如套接字）的可读性、可写性和异常情况，从而实现并发的 I/O 操作。`select` 库通常用于构建高效的网络服务器，能够同时处理多个客户端连接而无需为每个连接创建单独的线程或进程。它提供了一种异步 I/O 的解决方案，可以在一个线程中同时处理多个 I/O 事件，提高了应用程序的性能和可伸缩性。

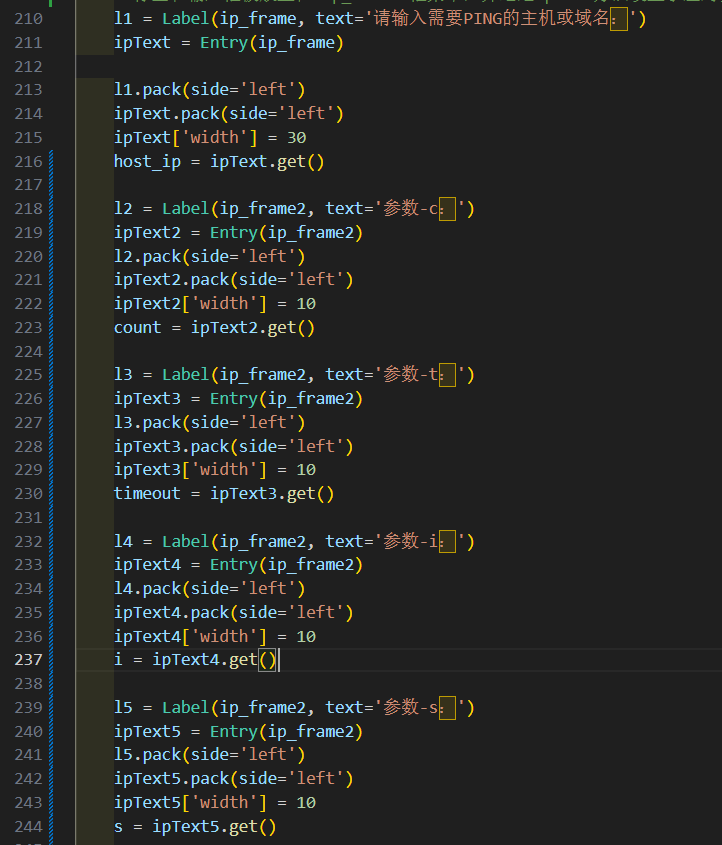
通过结合使用 `struct`、`socket` 和 `select` 这三个库，你可以处理二进制数据、进行网络编程，以及实现高效的并发 I/O 操作，从而构建各种复杂的网络应用程序。

## 4.2 构建GUI画面



这部分代码创建了一个顶级窗口，并设置了窗口的标题为 'PING\_Command'，以及窗口的初始大小为 800x500 像素。这些代码创建了一些框架 (Frame)。框架是 Tkinter 中的容器，可以用来组织和布局其他的 GUI 元素，如标签、按钮、文本框等。这里创建了四个框架：main\_frame、text\_frame、ip\_frame 和 botton\_frame。这部分代码创建了另外两个框架 ip\_frame2 和 ip\_frame3，它们也是被添加到 main\_frame 框架中的。

总体而言，以上代码创建了一个包含多个框架的 GUI 窗口。这些框架可以用来添加其他的 GUI 元素，并通过布局管理器进行排列和组织，以创建一个具有特定功能和外观的应用程序界面。



这部分代码创建了一个标签 (Label) 和一个文本输入框 (Entry)。标签的文本是 '请输入需要PING的主机或域名：'，而输入框是用于用户输入主机名或域名的地方。这两个 GUI 元素都被添加到了 ip\_frame 框架中。这些代码将标签和输入框添加到应用程序的窗口中，并设置输入框的宽度为 30 个字符。pack() 方法用于将 GUI 元素放置在父容器中的适当位置。这里使用 'left' 参数将它们水平排列在 ip\_frame 框架中的左侧。最后，ipText.get() 是用于获取用户在输入框中输入的文本内容，并将其赋值给变量 host\_ip。其他设定同理。



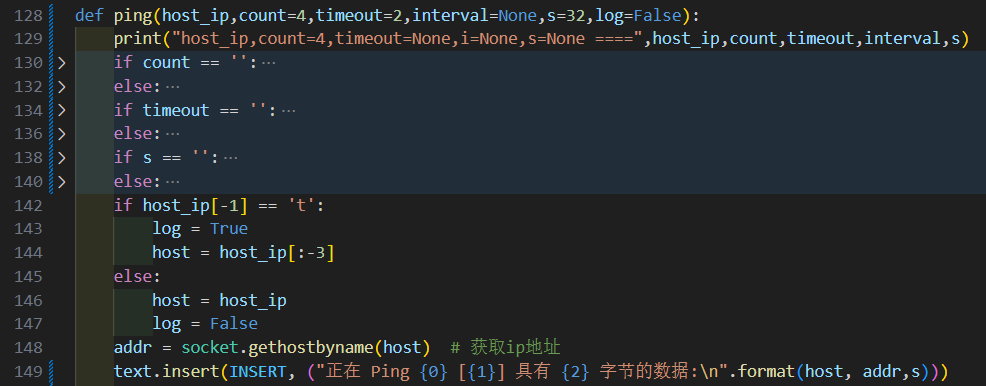
这部分代码创建了一个按钮 (Button)，按钮上显示的文本为 '发送'。按钮的点击事件与一个匿名函数绑定，这个匿名函数调用了一个名为 ping 的函数，并将 ipText.get()、ipText2.get()、ipText3.get()、ipText4.get() 和 ipText5.get() 作为参数传递给它。这些 get() 调用是用于获取之前创建的输入框 (Entry) 中用户输入的值。这些代码设置了按钮的宽度为 5 个字符，高度为 1 行。然后使用 pack() 方法将按钮添加到 ip\_frame 框架中，并通过设置 side='left' 参数使其靠左排列。通过以上代码，创建了一个带有文本为 '发送' 的按钮，并与一个点击事件绑定。按钮被添加到 ip\_frame 框架中，并设置了适当的宽度和高度。 这部分代码还创建了一个文本框 (Text)，用于显示文本内容。文本框的宽度设置为 100 个字符，高度设置为 30 行。然后使用 pack() 方法将文本框添加到 text\_frame 框架中。这些代码使用 pack() 方法将之前创建的各个框架 (main\_frame、ip\_frame、ip\_frame2、ip\_frame3 和 text\_frame) 添加到应用程序的窗口中，并设置它们的布局。main\_frame.pack() 将主框架添加到窗口中，并根据默认的布局方式进行排列。ip\_frame.pack(side='top', pady='5') 将 ip\_frame 框架添加到窗口中，通过设置 side='top' 参数使其在顶部排列，并使用 pady='5' 参数设置了上方的垂直间距为 5 像素。类似地，ip\_frame2 和 ip\_frame3 也被添加到窗口中，并按照相同的方式进行布局。text\_frame.pack() 将文本框所在的 text\_frame 框架添加到窗口中。

这样大致的就可以画出GUI界面，如下图所示：

## 4.3 调用Ping方法

调用在main()方式中的按钮中的Ping方法

    b = Button(ip\_frame, text='发送', command=lambda: ping(ipText.get(),ipText2.get(),ipText3.get(),ipText4.get(),ipText5.get()))



上述代码是一个用于执行Ping操作的函数。让我来为你解释一下每个部分的作用：

1. `def ping(host\_ip, count=4, timeout=None, interval=None, s=32, log=False):`

- 这是定义了一个名为 `ping` 的函数，接受多个参数，其中 `host\_ip` 是必需的参数，其他参数都有默认值。

2. `print("host\_ip,count=4,timeout=None,s=None,i=None ====", host\_ip, count, timeout, interval, s)`

- 这是一个打印语句，用于在控制台输出函数的参数值。

3. `count = int(count)`

- 这一行将参数 `count` 转换为整数类型。

4. `if host\_ip[-1] == 't':`

- 这是一个条件语句，检查 `host\_ip` 的最后一个字符是否为 't'。

5. `log = True`

- 如果 `host\_ip` 的最后一个字符是 't'，则将变量 `log` 设置为 `True`。

6. `host = host\_ip[:-3]`

- 如果 `host\_ip` 的最后一个字符是 't'，则将变量 `host` 设置为去掉最后3个字符的 `host\_ip`。

7. `else:`

- 如果 `host\_ip` 的最后一个字符不是 't'，则执行以下代码块。

8. `host = host\_ip`

- 在没有 't' 结尾的情况下，将变量 `host` 设置为 `host\_ip`。

9. `log = False`

- 在没有 't' 结尾的情况下，将变量 `log` 设置为 `False`。

10. `addr = socket.gethostbyname(host)`

- 使用 `socket.gethostbyname()` 函数获取主机的 IP 地址。

11. `text.insert(INSERT, ("正在 Ping {0} [{1}] 具有 {2} 字节的数据:\n".format(host, addr, s)))`

- 将一条信息插入到文本框中，表示正在 Ping 的主机信息。



12. `lost = 0`

- 初始化丢失的数据包数量为 0。

13. `accept = 0`

- 初始化接收的数据包数量为 0。

14. `sumtime = 0.0`

- 初始化往返时间总和为 0。

15. `times = []`

- 创建一个空列表用于存储每个数据包的往返时间。

16. `i = 0`

- 初始化一个计数器变量 `i` 为 0。

17. `req\_count = count`

- 将初始的请求数量设置为 `count`。

18. `while count:`

- 进入一个循环，当 `count` 不为 0 时继续执行。

19. `if log:`

- 如果 `log` 为 `True`，执行以下代码块。

20. `count = count + 1`

- 将 `count` 的值增加 1。

21. `else:`

- 如果 `log` 不为 `True`，执行以下代码块。

22. `count = count - 1`

- 将 `count` 的值减少 1。

23. `i += 1`

- 将计数器变量 `i` 的值增加 1。

24. `icmp\_packet = request(i)`

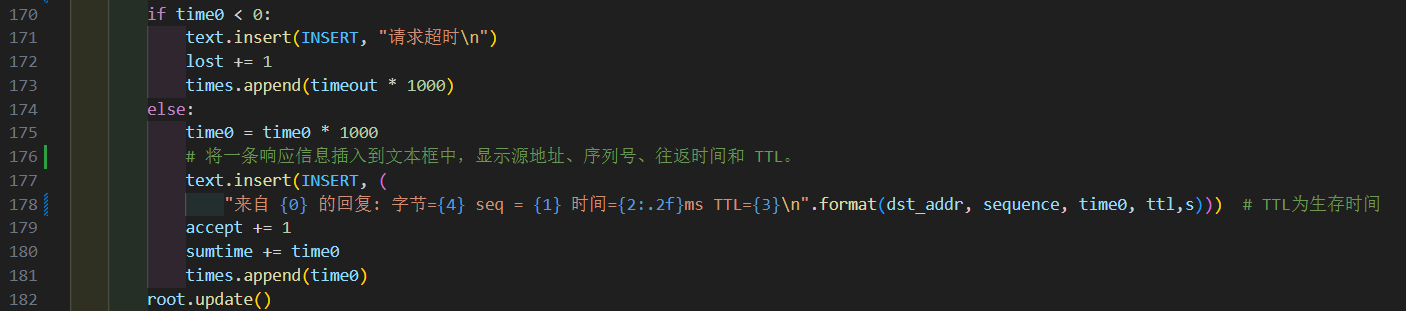
- 调用 `request()` 函数生成 ICMP 请求数据包。

25. `rawsocket, dst\_addr = connect(addr, icmp\_packet)`

- 调用 `connect()` 函数建立与目标地址的连接，并返回原始套接字和目标地址。

26. `time0, sequence, ttl = reply(rawsocket, time.time(), timeout)`

- 调用 `reply()` 函数发送 ICMP 请求并等待接收响应，返回往返时间、序列号和 TTL。



27. `if time0 < 0:`

- 如果往返时间小于 0，表示请求超时。

28. `text.insert(INSERT, "请求超时\n")`

- 将一条信息插入到文本框中，表示请求超时。

29. `lost += 1`

- 丢失的数据包数量的计数增加 1。

30. `times.append(timeout \* 1000)`

- 将超时时间乘以 1000（转换为毫秒）后，将其加入到 `times` 列表中。

31. `else:`

- 如果往返时间大于等于 0，表示收到了响应。

32. `time0 = time0 \* 1000`

- 将往返时间乘以 1000（转换为毫秒）。

33. `text.insert(INSERT, ("来自 {0} 的回复: 字节=32 seq = {1} 时间={2:.2f}ms TTL={3}\n".format(dst\_addr, sequence, time0, ttl)))`

- 将一条响应信息插入到文本框中，显示源地址、序列号、往返时间和 TTL。

34. `accept += 1`

- 接收的数据包数量计数增加 1。

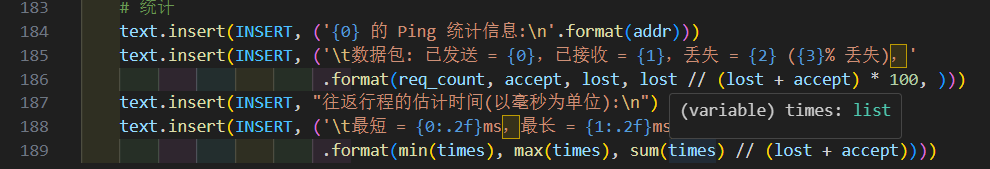
35. `sumtime += time0`

- 将往返时间累加到总和中。

36. `times.append(time0)`

- 将往返时间加入到 `times` 列表中。

37. `root.update()`



- 更新界面，以确保信息显示在文本框中。

38. `text.insert(INSERT, ('{0} 的 Ping 统计信息:\n'.format(addr)))`

- 将一条信息插入到文本框中，表示 Ping 统计信息的标题。

39. `text.insert(INSERT, ('\t数据包: 已发送 = {0}，已接收 = {1}，丢失 = {2} ({3}% 丢失)，'.format(req\_count, accept, lost, lost // (lost + accept) \* 100, )))`

- 将一条信息插入到文本框中，显示发送的数据包数量、接收的数据包数量、丢失的数据包数量和丢包率。

40. `text.insert(INSERT, "往返行程的估计时间(以毫秒为单位):\n")`

- 将一条信息插入到文本框中，表示往返行程时间的标题。

41. `text.insert(INSERT, ('\t最短 = {0:.2f}ms，最长 = {1:.2f}ms，平均 = {2:.2f}ms\n'.format(min(times), max(times), sum(times) // (lost + accept))))`

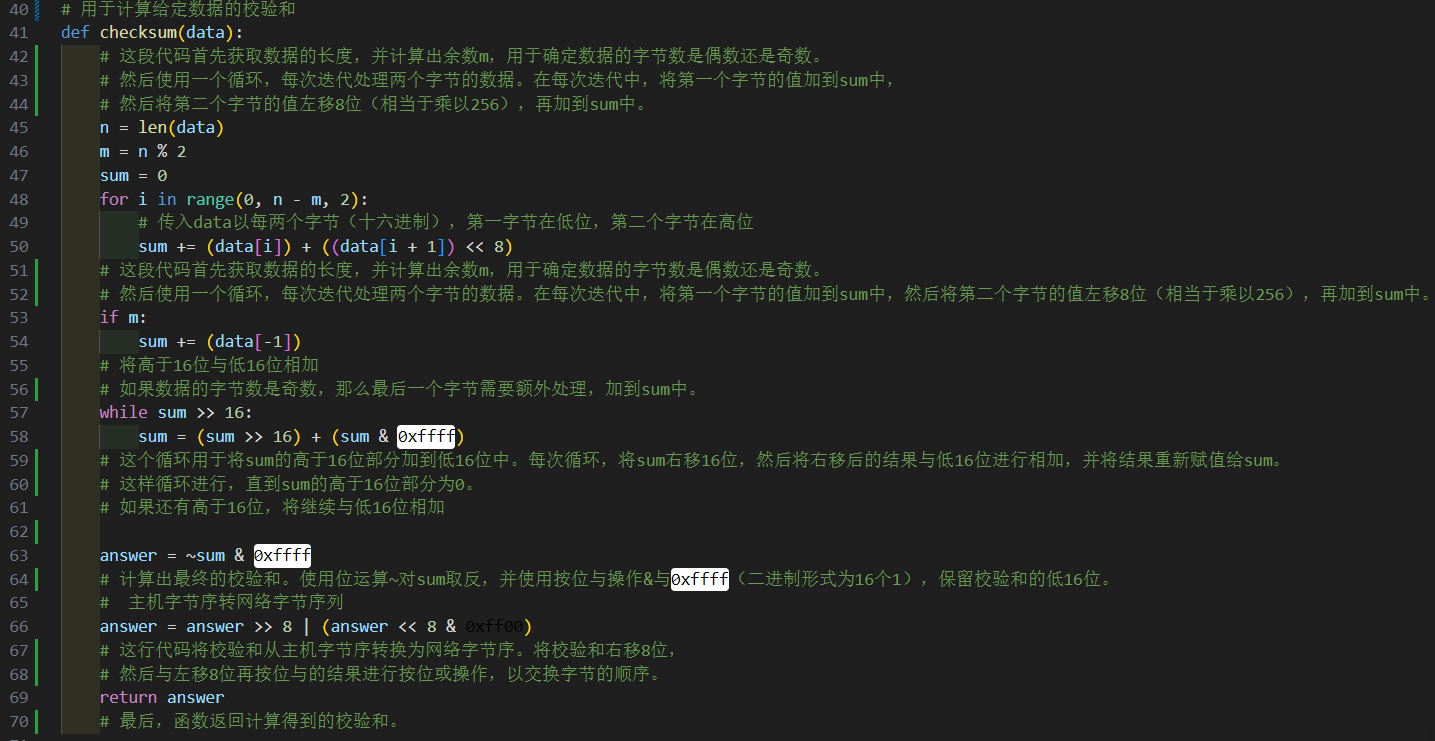
- 将一条信息插入到文本框中，显示往返行程时间的最短值、最长值和平均值。

这段代码执行了一个简单的 Ping 操作，向指定的主机发送 ICMP 请求并接收响应，然后计算并显示统计信息。

## 4.4 connect()函数

# 实例化一个socket对象，ipv4，原始套接字，分配协议端口 # 这一行代码创建了一个套接字对象rawsocket，使用IPv4地址族和原始套接字类型。 # socket.getprotobyname("icmp")返回ICMP协议的协议号，# 并将其作为第三个参数传递给socket.socket函数，以指定使用ICMP协议。 # 发送数据到网络 # 这一行代码使用sendto方法将icmp\_packet发送到指定的地址(addr, 800)。icmp\_packet是一个包含ICMP数据的字节序列。最后返回数据

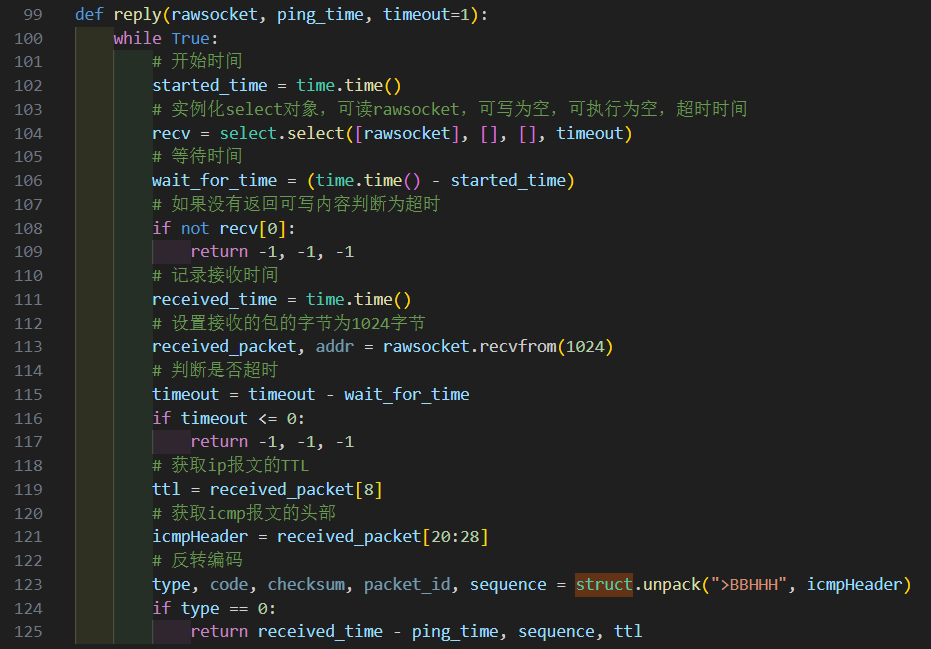
## 4.5 checksum()函数



## 4.6 request()函数



## 4.7 reply()函数



# 五、总结

以下是对实验内容的总结：

实验目标：

本实验的目标是理解并编程实现 Ping 程序的工作原理。具体包括构造 ICMP 回送请求报文并封装到 IP 数据报中，发送数据报并分析收到的结果，计算和呈现一些统计信息，并在图形界面中显示。

实验步骤和实现：

1. 理解 Ping 的工作原理：Ping 是一种网络工具，用于测试主机之间的连通性。Ping 使用 ICMP 协议发送回送请求报文，并等待目标主机返回回送应答报文。通过测量往返时间和检查数据包丢失情况，可以评估网络的稳定性。

2. 构造 ICMP 回送请求报文和封装 IP 数据报：使用编程语言（如 Python）创建一个程序，可以构造 ICMP 回送请求报文，并将其封装到 IP 数据报中。ICMP 报文包括类型、代码、校验和等字段，用于在网络上进行通信。

3. 发送数据报和分析结果：使用网络套接字库（如 Python 的 `socket` 模块）发送构造的 IP 数据报。发送数据报后，等待目标主机的回送应答报文，并接收和分析收到的结果。可以使用套接字的超时设置来控制等待时间。

4. 统计信息的计算与呈现：根据收到的回送应答报文，计算并呈现一些统计信息，如往返时间（RTT）的平均值、最小值、最大值，以及丢包率等。这些统计信息可以在图形界面中以易于理解的方式显示。

5. 支持不同参数的运行：实现支持不同参数的运行，例如指定发送的 ICMP 报文数量、设置 IP 数据报中的 TTL 等。这样可以模拟 Windows 中 Ping 命令的部分选项功能，使程序更加灵活和可定制。

实验结果和收获：

通过完成这个实验，对 Ping 工具的工作原理有了更深入的理解，并成功编程实现了一个 Ping 程序。通过构造 ICMP 报文和封装 IP 数据报，我们能够发送数据报并分析收到的结果，计算和呈现一些统计信息，并在图形界面中显示。此外，通过支持不同参数的运行，增加了程序的灵活性和可定制性。

这个实验有助于加深对网络协议和工具的理解，并提高编程能力。通过实践，我们能够更好地理解网络通信的细节，并掌握如何使用编程语言来实现网络工具。同时，图形界面的设计也提升了用户体验，使 Ping 程序更加易用和友好。

总而言之，通过完成这个实验，我们学习了 Ping 工具的工作原理，掌握了编程实现 Ping 程序的方法，并实现了一些常见的功能和选项。这对于网络和系统管理、网络性能评估以及网络故障排除等方面都具有重要意义。