套接字联网API

大佬说这本书只用看前8章，相信大佬好了，多了其实也看不完。。

看了《UNIX高级环境编程》之后，知道了套接字其实也是一种IPC的方式，而且套接字可以与文件描述符关联。也可以实现通过文件描述符来输入和输出内容，因此，我对这本书《套接字编程》的理解是详细介绍套接字编程，讲解IPC中套接字的具体实现和使用。所以看来先看《高级环境编程》是正确的。

第一章：简介

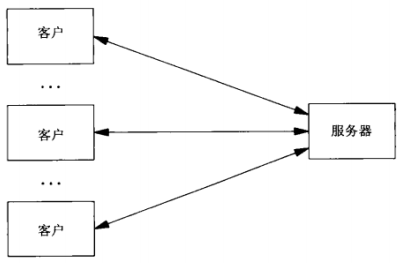
1、要编写通过计算机网络通信的程序，首先要确定这些程序相互通信所使用的的协议

2、一般认为Web服务器程序是一个长时间运行的程序（守护程序，记忆中好像只用设置一些方式就可以了），它只在响应来自网络的请求时才发送网络消息。协议的另一端是Web客户程序，如某种浏览器，与服务器进程的通信总是由客户进程发起的。

3、一般按照客户端和服务器来组织

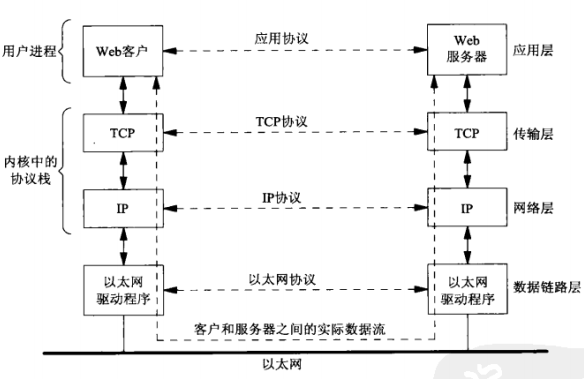


通常客户每次只与一个服务器通信，但一个服务器会与多个客户进行通信

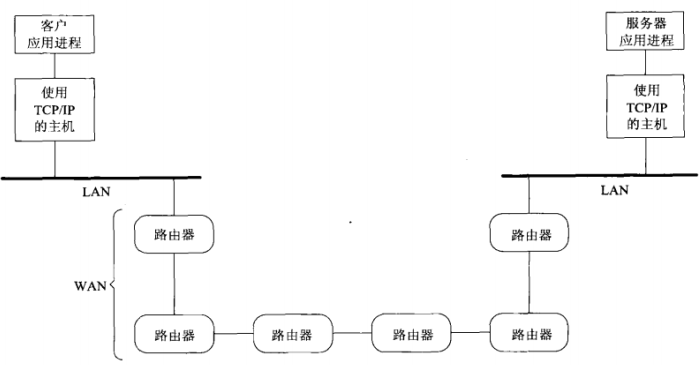


客户机和服务器使用某个网络协议通信的，本书的焦点是TCP/IP。实际上会一层一层的进行通讯

一层一层的逐渐下降



同一网络应用的客户和服务器无需处于同一个局域网。如图所示



4、接下来介绍了一个获取服务器时间的客户程序：

包含头文件

命令行参数

创建TCP套接字

指定服务器的IP地址和端口

建立与服务器的连接

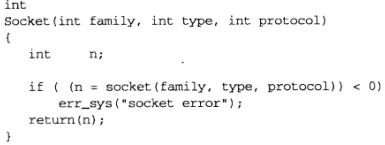
读入并输出服务器的应答

终止程序

协议无关性：程序兼容不同协议，例如上面的程序可以使用IPv4或IPv6

5、错误处理：程序必须检查每个函数调用是否返回错误。发生错误时就用我们的err\_quit或err\_sys函数输出一个出错消息并终止程序的运行。既然发生错误时终止程序的运行是非常普遍的情况，我们可以定义包裹函数

如下，Socket就是socket的包裹函数



本书中首字母大写的函数名，就是定义的包裹函数，调用的实际函数的名字与包裹函数名相同，只是以对应的小写字母开头。

本书的例子中，除非必须检查某个确定的错误是否发生，并以不同于终止进程的其他某种方式处理它，否则就是用这些包裹函数。（看UNIX高级环境编程时，发现对错误的处理确实会让代码很乱，包裹函数确实让代码简洁了很多。）

7、UNIX errno值：只要一个Unix函数中有错误发生，全局变量errno就被设置为一个指明该错误类型的正值，函数本身则返回-1，全书中使用例如“connect函数返回ECONNREFUSED”这样的句子简明表达以下意思：该函数返回一个错误，同时errno被置为指定的常值。

8、服务器程序：（socket bind listen这三步是任何TCP服务器准备所谓的监听描述符的正常步骤）

创建TCP套接字（socket）

把服务器的众所周知端口捆绑到套接字（时间获取服务是13）（bind）

把套接字转换成监听套接字（listen）

接受客户连接，发送应答（accept，服务器在这一步等待，TCP采用三次握手来建立连接，握手完毕返回的是一个已连接描述符的新描述符）

终止连接（调用close关闭与客户的连接，引发正常的TCP连接终止序列。 三次握手，四次挥手。）

（与版本无关的程序在11章，这是一个迭代式的服务器，一次只能响应一个客户端的请求，并发式的最简单的是使用Unix的fork函数。为每个客户创建一个子进程，其他技术包括线程代替fork，或在服务器启动时预先fork一定数量的子进程

9、如果在shell命令行启动本例这样的一个服务器，我们也许想要它运行很久，这要求我们在服务器中添加代码，让它作为一个Unix守护进程（daemon）

10、OSI网络七层模型需要知道：

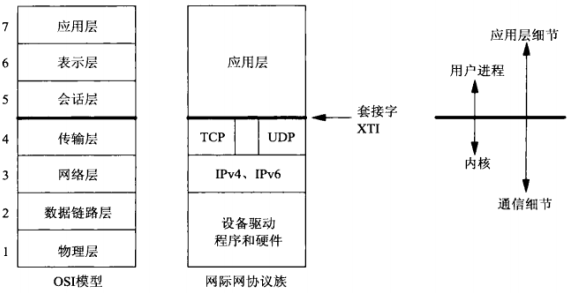
底下两层是随系统提供的设备驱动程序和网络硬件。通常除了需要知道数据链路的某些特性外（如MTU为1500字节），不必关心这两层的具体情况

网络层由IPv4和IPv6这两个协议处理，传输层有UDP和TCP，网络应用绕过传输层直接使用IPv4或IPv6是可能的。也就是所谓的原始套接字

OSI顶上3层被合并为一层，称为应用层。

11、本书的焦点是如何使用套接字编写使用TCP或UDP的网络应用程序。

为什么套接字提供从OSI模型的顶上三层进入传输层的接口？：顶上三层处理具体网络应用的所有细节，对通信细节了解很少；底下四层对具体网络应用了解不多，却处理所有的通信细节；其次是顶上三层构成所谓用户进程，底下四层却通常作为操作系统内核的一部分提供。UNIX与其他现代操作系统都提供分隔用户进程与内核的机制，由此可见，第四层和第五层之间的接口是构建API的自然位置



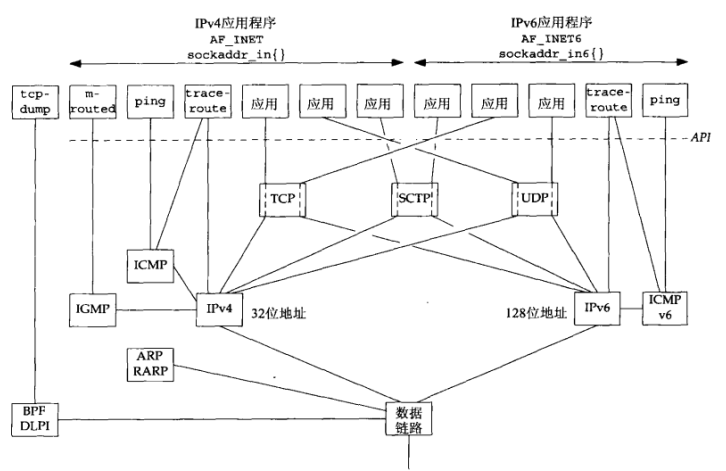
12、网络拓扑：大多数UNIX系统都提供了可用于发现某些网络细节的两个基本命令，netstat和ifconfig。Netstat -i提供网络接口的信息，-n标志指定以输出数值地址，netstat -r展示路由表，也是另一种确定接口的方法。有了各个网络接口的名字，执行ifconfig就可以获得每个接口的详细信息。对本地接口的广播地址执行ping就可以获得本地网络中众多主机的IP地址

13、size\_t在32位系统中是32位，但在64位中就会变为64位，同样还有long ，有滴地方不需要变化，因此在不需要变化的地方设计特有的数据类型

第二章：传输层 TCP UDP SCTP

本章介绍传输层的协议的概貌。从网络编程角度提供足够的细节理解如何使用这些协议

1、总图：右边的网络应用使用IPv6，随sockaddr\_in6结构讲解AF\_INET6常值，左边的使用IPv4，使用sockaddr\_in结构，值为AF\_INET



（应该记住各种协议的名称）

2、用户数据报协议（UDP）：应用程序向UDP套接字中写入一个消息，该消息随后被封装到一个UDP数据报，该UDP数据报随后又被封装到一个IP数据报，然后发送到目的地。UDP不保证UDP数据报回到达其最终目的地，不保证每个数据报的先后顺序跨网络保持不变，也不保证每个数据报只到达一次。UDP缺乏可靠性，如果一个数据报到达了最终目的地，但是校验和检测有错误，或者数据报被中途丢弃了，它就无法被投递给UDP套接字，也不会被源端自动重传。如果想要确保一个数据报到达目的地，可以往应用程序中添加一大堆的特性：来自对端的确认，本端的超时和重传等。每个UDP数据报都有一个长度，如果数据报正确的到达目的地，该数据报的长度将随数据一道传递给接收端应用进程，而TCP是一个字节流协议，没有任何记录边界。UDP提供无连接的服务，因为UDP客户和服务器之间不必存在任何长期的关系，例如一个UDP客户可以创建一个套接字发送一个数据报给一个给定的服务器，然后立即用同一个套接字发送给另一个数据报给另一个服务器，同样地，一个UDP服务器可以用同一个UDP套接字从若干个不同的客户接受数据报，每个客户一个数据报。

3、传输控制协议（TCP）：

TCP提供客户与服务器之间的连接，TCP客户先与某个给定服务器建立一个连接，再跨该连接与那个服务器交换数据，然后终止这个连接。

TCP提供了可靠性，当TCP向另一端发送数据时，它要求对端返回一个确认，如果没有收到确认，TCP就自动重传数据并等待更长时间。在数次重传失败后，TCP才放弃。重传一般花费4~10分钟

TCP有动态估算客户和服务器之间的往返时间RTT的算法，以便它知道等待一个确认要多少时间。

TCP通过每个字节关联一个序列号对所发送的数据进行排序。如果TCP接收到来自对端的重复数据，它可以判定数据是重复的，从而丢弃重复数据

TCP提供流量控制，TCP总是告知对端在任何时刻它一次能够从对端接受多少字节的数据，这称为通告窗口，窗口指出接受缓冲区中当前可用的空间量，从而确保发送端发送的数据不会使接受缓冲区溢出。该窗口动态变化，当接受到来自发送端的数据时，窗口大小就减小。

TCP是全双工的，在一个给定的连接上应用可以在任何时刻在进出两个方向上既发送数据又接收数据。

4、流控制传输协议（SCTP ）

SCTP提供类似UDP和TCP的服务

SCTP在客户端和服务器之间提供关联，并像TCP那样给应用提供可靠性、排序、流量控制以及全双工的数据传送。一个关联指代两个系统之间的一次通信，它可能因为SCTP支持多宿而涉及不止两个地址。

SCTP面向消息，提供各个记录的按序递送服务。和UDP一样，由发送端写入的每条记录的长度随数据一道传递给接收端应用。

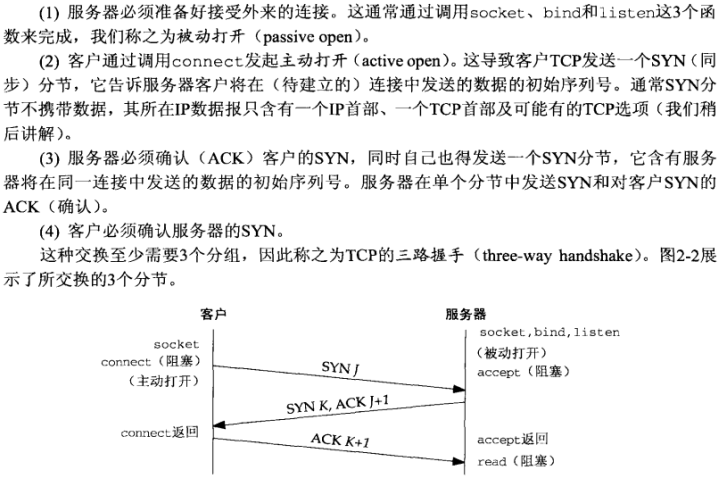
SCTP能够在所连接的端点之间提供多个流，每个流各自可靠地按序递送消息。一个流上某个消息的丢失不会阻塞同一关联其他流上消息的投递。与TCP正好相反

SCTP还提供多宿特性，使得单个SCTP端点能够支持多个IP地址。该特性可以增强应对网络故障的健壮性。一个端点可能有多个冗余的网络连接，每个网络又可能有各自介入因特网基础设施的连接。SCTP可以通过切换到使用已与该关联相关的另一个地址来规避所发生的的故障。

5、TCP的建立和终止

这有助于理解connect、accept、close这三个函数并使用netstat程序调试TCP应用

三次握手：建立一个TCP连接时发生下述情形。



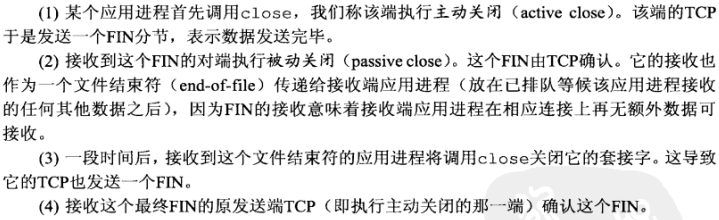
SYN报文可以含有多个TCP选项，

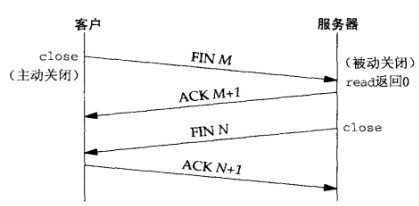
如MSS选项：最大分节大小，MSS，也就是它在本连接的每个TCP分节中愿意接受的最大数据量。

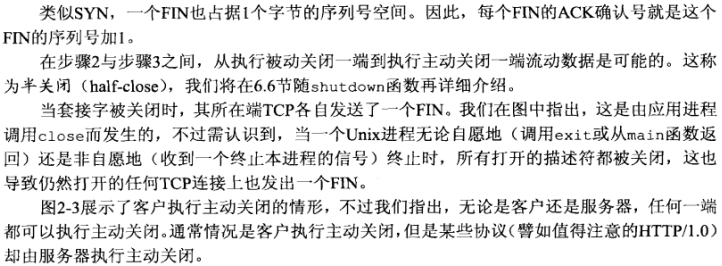
窗口规模选项：TCP连接任何一端能够通告对端的最大窗口大小是65535，因为在TCP首部中相应的字段占16位。新选项指定TCP首部中的通告窗口必须扩大的位数是0~14，因此所提供的最大窗口65535\*214 约为1GB

时间戳选项：防止失而复现的网络分组造成数据损坏。网络编程人员可以不考虑

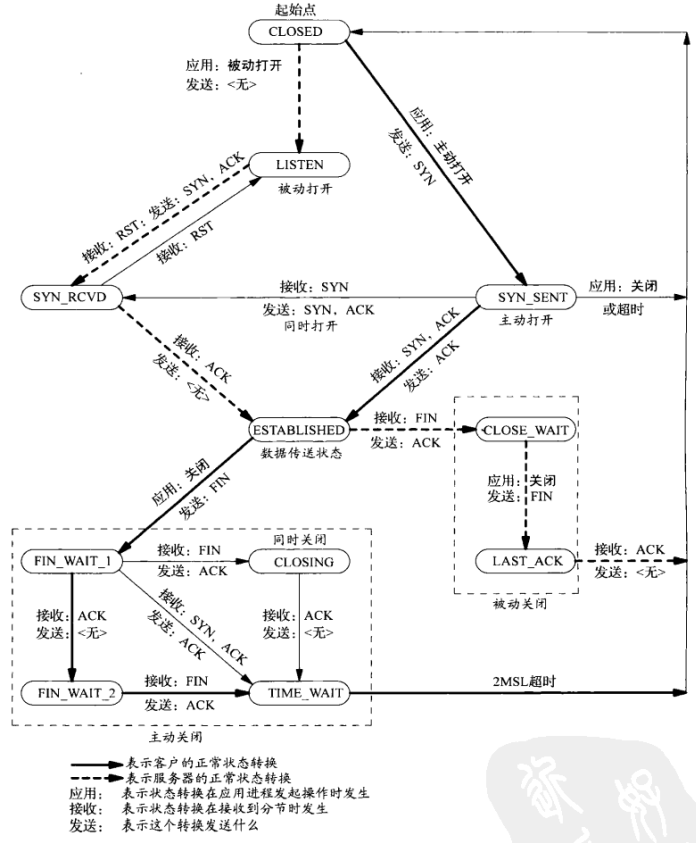
TCP终止连接的4次挥手：







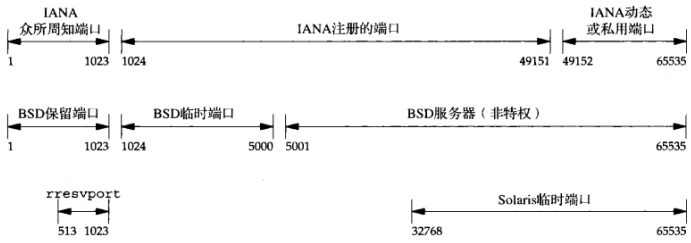
6、TCP状态转换图：TCP涉及连接建立和连接终止的操作可以用状态转换图来说明，提供有11种状态。这些状态可以使用netstat显示。



7、TIME\_WAIT状态存在的理由：可靠地实现TCP全双工连接的终止；允许老的重复分节在网络中消逝。2MSL

8、任何时候，多个进程使用TCP UDP SCTP传输层协议，3个协议都使用16位整数的端口号来区分这些进程。

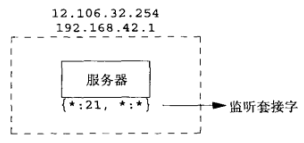
众所周知的端口为0~1023。已登记的端口为1024-49151，49151~65535是动态的或私用的端口。



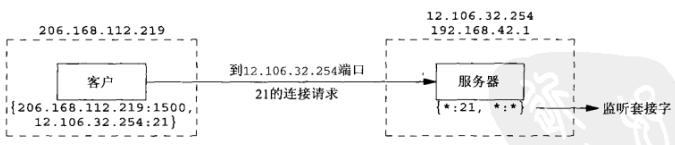
9、套接字对：一个TCP连接的套接字对是一个定义该连接的两个端点的四元组：本地IP地址、本地TCP端口号、外地IP地址、外地TCP端口号。

10、TCP端口号与并发服务器：过程图解

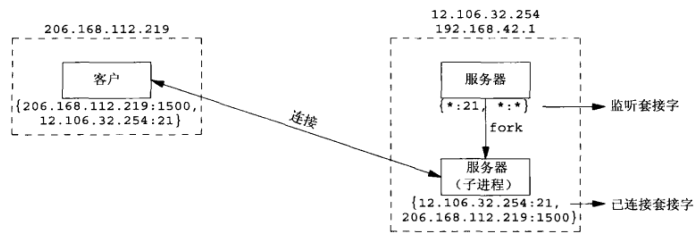
服务器本地任意接口的端口21上等待监听，然后监听套接字对为｛\*：21，\*：\*｝



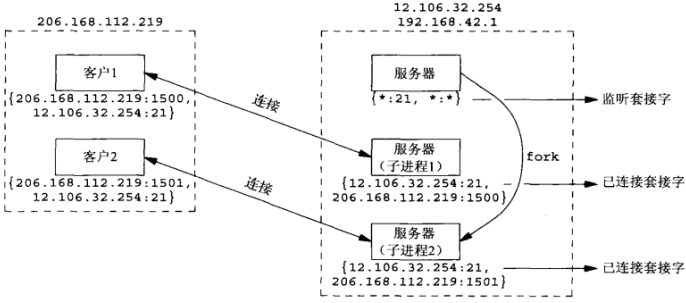
客户端主动打开，



客户和服务器进行连接，服务器将fork一个自身的副本，让子进程来处理该客户的请求。



假设另一个客户请求连接到同一个服务器，则产生第二个套接字对



TCP无法仅仅通过查看目的端口号来分离外来的分节到不同的端点。它必须查看套接字对的所有4个元素才能确定哪个端点接受某个到达的分节。

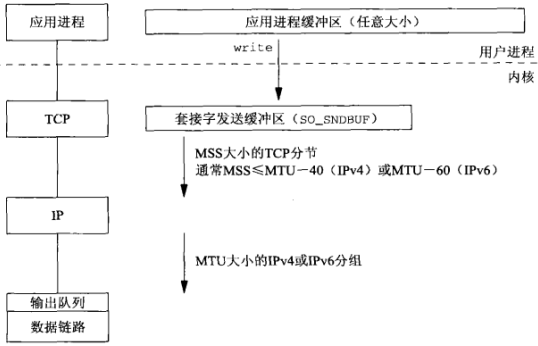
11、缓冲区大小和限制

IPv4 65535字节 包含首部

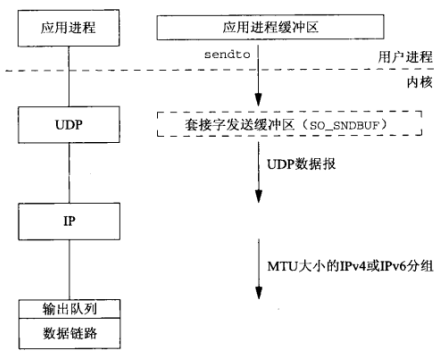
以太网 MTU 1500字节

当一个IP数据报从某个接口送出时，如果大小超过相应链路MTU，IPv4和IPv6都将执行分片

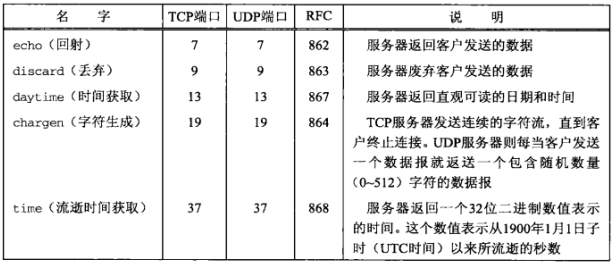
12、TCP输出：设定套接字发送缓冲区选项SO\_SNDBUF来设定缓冲区大小，然后应用进程调用write时，内核从该应用进程缓冲区中复制所有数据到所写套接字的发送缓冲区，



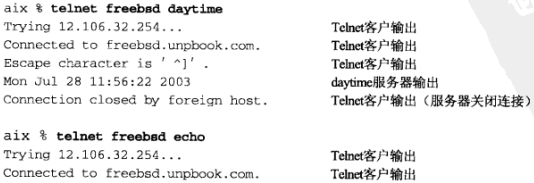
UDP输出：



13、标准因特网服务：如下图



提供标准的Telnet客户程序完成简易的测试机制，如：





14、常见互联网应用的协议使用：



第二部分：基本套接字编程

第三章 套接字编程简介