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

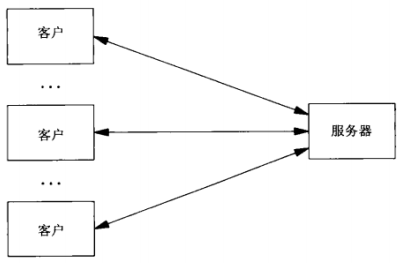
要编写通过计算机网络通信的程序，首先要确定这些程序相互通信所使用的的协议

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。

一般按照客户端和服务器来组织

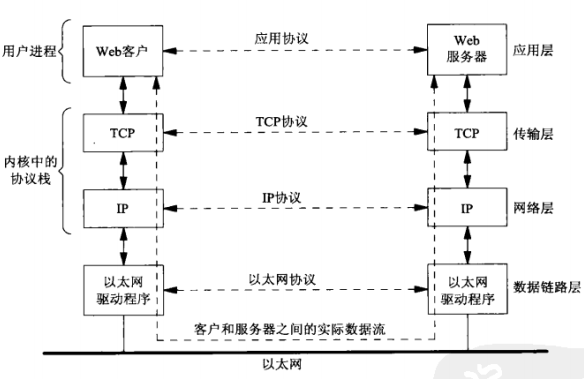


通常客户每次只与一个服务器通信，但一个服务器会与多个客户进行通信

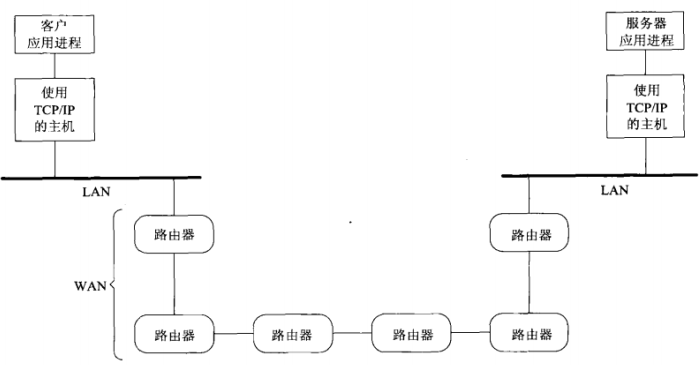


客户机和服务器使用某个网络协议通信的，本书的焦点是TCP/IP。实际上会一层一层的进行通讯

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



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

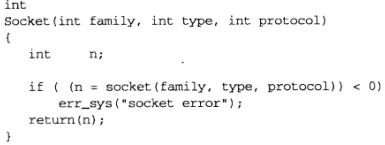
读入并输出服务器的应答

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

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。）

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。

服务器程序：

创建TCP套接字

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）

把套接字转换成监听套接字

接受客户连接，发送应答

终止连接（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

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）

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。

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

