**01.列举10个常见的Linux命令**

cd pwd touch ls mkdir rm help sudo ssh date clear vim ps cat more scp cp find mv grep echo

**Linux**

/bin /home /dev /proc /mnt /usr /etc /lib

/bin:存储系统所使用命令的可执行文件

/home：普通用户的家目录 (cd home)；

/dev:外部设备（存的是设备的接口 通过接口能访问设备）；

/proc：虚拟目录 以进程为单位存储内存的映射--------一些统计信息；

/mnt：临时挂载点（将接口挂载到临时挂载点进行操作）；

/usr：第三方软件的一些文档；

/etc：系统配置目录；

/lib：库文件（静态库 共享库（动态库））

冯\*诺依曼 五大部件

（计算器 控制器）cpu (存储器)内存 主存------缓存数据（为了使cpu效率提高）（输入设备 输出设备）I/O 硬盘 显示器 键盘

数据从硬盘读到内存 再读到cpu去执行

Linux上一切皆文件：不以文件扩展名区分文件类型（将文件分为五大类）

          普通文件：-（.c     .h     .cpp   .java       .class       .txt     .pdf都属于普通文件）  
        目录文件：d   文件夹  
        链接文件:l(L)  
        管道文件:p （进程间通讯的专用文件）  
        设备文件 ：字符设备(c)           块设备(b)           套接字(s)

ls -l

文件提供给不同用户不同的权限（属主（创建者） 属组（组用户的名称） 其他用户

三种用户 ：三组权限

          u             g         o

r： 读权限

w：写权限

x：执行权限 普通文件（前提是他是可执行文件）

是否可进入   目录文件         针对文件不同，x的含义不同

文件操作命令：

文件创建：普通文件 （touch） 目录文件----文件夹（mkdir）

文件删除：普通文件（rm） 目录文件(rmdir-----删除空目录 rm -r 删除非空目录)

文件拷贝：普通文件（cp原文件路径+文件名 拷贝的目的地） 目录文件(cp -r 原文件路径+文件名 拷贝目的地的路径)把目录里面的文件都拷贝过去

文件剪切/重命名：普通文件（mv） 目录文件（mv）

边移动边重命名：如下图 stu3目录下本来就有一个project1所以剪切过来要重命名。这里的project1和project2是相同的文件

mv /home/stu1/project1 (这里中间要用空格隔开)/home/stu1/project2 单纯的重命名

修改属性：（文件类型一旦创建就不能修改，权限可以修改，链接数不能改，是随着操作变化的；属主可以改；属组可以修改；文件大小不能改，是你往里面加了东西才修改）

修改属主：只能root用户修改，chown newuser（改成谁） filename（所改的文件）

修改属组：只能root用户修改，chgrp newgroup（新的组，原来在那个组不用关心） filename

修改权限：chomd

1、用字符方式修改 chmod a（all所有的）/u/g/o +/-/=(赋值) r/w/x/rw/rx/wx/rwx

2、用数字方式修改：

每个n代表一个用户的权限，一共有3个n,每个n都是0-7的数字

chmod nnn filename

linux基本命令

cd 跟路径 ：切换目录

ls -l ：显示文件详细信息

ls -a ： 显示所有文件 （包含隐藏文件文件名以“.”开头的都是隐藏文件）

cd .. ：可以退回上级目录

cd - ：两个目录之间来回切换

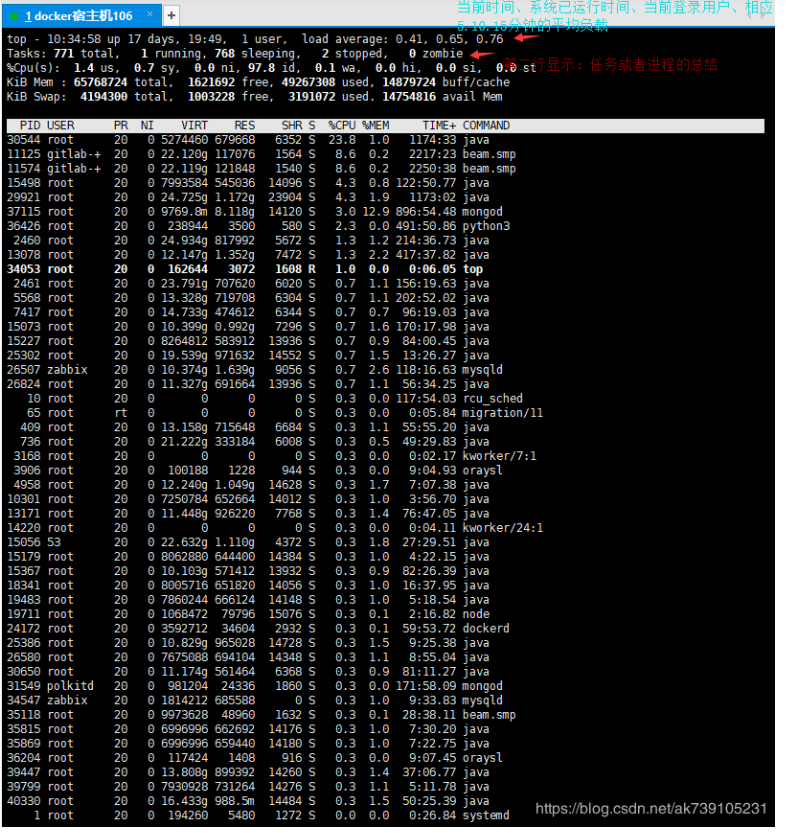
cd ~ ：直接返回家目录

pwd：查看当前目录的绝对路径/查看当前位置

分类: [Docker](https://www.cnblogs.com/minseo/category/1207246.html), [Linux](https://www.cnblogs.com/minseo/category/1000755.html), [Ubuntu](https://www.cnblogs.com/minseo/category/1376975.html)

**1.top**

**top是linux中自带的系统监控命令，实时监控系统各项指标**

****

**第一行各字段含义：**

**这些字段显示：**

**当前时间**

**系统已运行的时间**

**当前登录用户的数量**

**相应最近5、10和15分钟内的平均负载。**

**Ps：1核cpu饱满负载为1，1以下均正常不会出现拥堵情况**

**# 总核数 = 物理CPU个数 X 每颗物理CPU的核数**

**# 总逻辑CPU数 = 物理CPU个数 X 每颗物理CPU的核数 X 超线程数**

**# 查看物理CPU个数cat /proc/cpuinfo| grep "physical id"| sort| uniq| wc -l**

**# 查看每个物理CPU中core的个数(即核数)cat /proc/cpuinfo| grep "cpu cores"| uniq**

**# 查看逻辑CPU的个数cat /proc/cpuinfo| grep "processor"| wc -l**

**1.物理cpu数：主板上实际插入的cpu数量，可以数不重复的 physical id 有几个（physical id）**

**2.cpu核数：单块CPU上面能处理数据的芯片组的数量，如双核、四核等 （cpu cores）**

**3.逻辑cpu数：一般情况下，逻辑cpu=物理CPU个数×每颗核数，如果不相等的话，则表示服务器的CPU支持超线程技术（HT：简单来说，它可使处理器中的1 颗内核如2 颗内核那样在操作系统中发挥作用。这样一来，操作系统可使用的执行资源扩大了一倍，大幅提高了系统的整体性能，此时逻辑cpu=物理CPU个数×每颗核数x2）**

**第二行字段含义：**

**第二行显示的是任务或者进程的总结。进程可以处于不同的状态。这里显示了全部进程的数量。除此之外，还有正在运行、睡眠、停止、僵尸进程的数量（僵尸是一种进程的状态）。**

**第三行字段含义：**

**第三行主要显示cpu信息：**

**us, user： 运行(未调整优先级的) 用户进程的CPU**

**sy，system: 运行内核进程的CPU**

**ni，niced：运行已调整优先级的用户进程的CPU**

**Id，空闲cpu**

**wa，IO wait: 用于等待IO完成的CPU**

**hi：处理硬件中断的CPU**

**si: 处理软件中断的CPU**

**st：这个虚拟机被hypervisor偷去的CPU（译注：如果当前处于一个hypervisor下的vm，实际上hypervisor也是要消耗一部分CPU处理时间的）。**

**第四行和第五行字段含义：**

**第四行是物理内存使用：**

**物理内存显示如下:全部可用内存、已使用内存、空闲内存、缓冲内存。**

**第五行是虚拟内存使用(交换空间)：**

**相似地：交换部分显示的是：全部、已使用、空闲和缓冲交换空间。**

**第四行中使用中的内存总量（used）指的是现在系统内核控制的内存数，空闲内存总量（free）是内核还未纳入其管控范围的数量。纳入内核管理的内存不见得都在使用中，还包括过去使用过的现在可以被重复利用的内存，内核并不把这些可被重新使用的内存交还到free中去，因此在linux上free内存会越来越少，但不用为此担心。**

**如果出于习惯去计算可用内存数，这里有个近似的计算公式：第四行的free + 第四行的buffers + 第五行的cached，按这个公式此台服务器的可用内存：**

**对于内存监控，在top里我们要时刻监控第五行swap交换分区的used，如果这个数值在不断的变化，说明内核在不断进行内存和swap的数据交换，这是真正的内存不够用了。**

**第7行各字段含义：**

**PID：进程ID，进程的唯一标识符**

**USER：进程所有者的实际用户名。**

**PR：进程的调度优先级。这个字段的一些值是'rt'。这意味这这些进程运行在实时态。**

**NI：进程的nice值（优先级）。越小的值意味着越高的优先级。负值表示高优先级，正值表示低优先级**

**VIRT：进程使用的虚拟内存。进程使用的虚拟内存总量，单位kb。VIRT=SWAP+RES**

**RES：驻留内存大小。驻留内存是任务使用的非交换物理内存大小。进程使用的、未被换出的物理内存大小，单位kb。RES=CODE+DATA**

**SHR：SHR是进程使用的共享内存。共享内存大小，单位kb**

**S：这个是进程的状态。它有以下不同的值:**

**D - 不可中断的睡眠态。**

**R – 运行态**

**S – 睡眠态**

**T – 被跟踪或已停止**

**Z – 僵尸态**

**%CPU：自从上一次更新时到现在任务所使用的CPU百分比。**

**%MEM：进程使用的可用物理内存百分比。**

**TIME+：任务启动后到现在所使用的全部CPU时间，精确到百分之一秒。**

**COMMAND：运行进程所使用的命令。进程名称（命令名/命令行）**

**Top其他快捷命令：**

**1.按1显示所有逻辑cpu的情况**

**2.按e显示各进程占用资源单位单位**

**3.按E显示整体占用资源单位**

**4.进程排序默认按%cpu进程，shift+”<”或”>”改变排序字段**

**5.d 指定每两次屏幕信息刷新之间的时间间隔。当然用户可以使用s交互命令来改变之。**

**6.p 通过指定监控进程ID来仅仅监控某个进程的状态。**

**7.i 使top不显示任何闲置或者僵死进程。**

**8.c 显示整个命令行而不只是显示命令名**

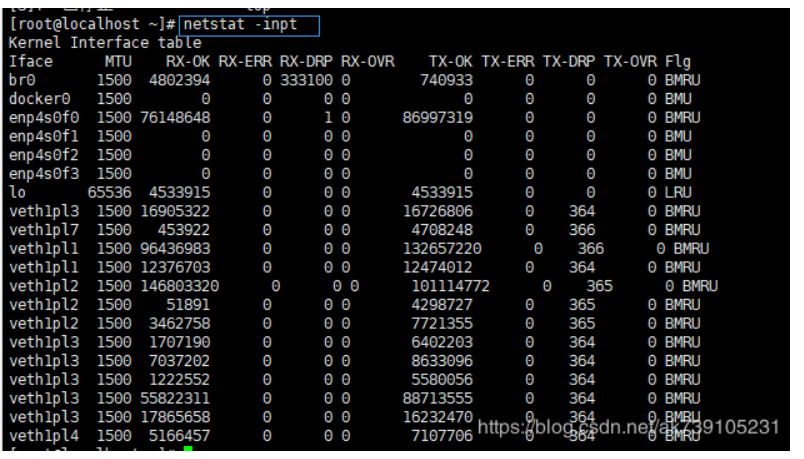
**2.查看进程的几种方式**

**1.ps -ef  查看当前运行的所有进程**

**2.ps -aux  查看正在内存中的进程**

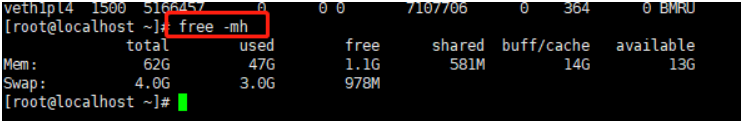
**3.查看系统端口**

**netstat  -lnpt**

****

**4.查看内存**

**free -mh**

****

**5.重定向**

**1.追加重定向 >>**

**2.覆盖重定向  >**

**6.查看磁盘空间**

**df -Th  统计磁盘空间**

**du -sh 统计当前目录大小**

**du -sh \*  列出当前目录下所有文件和文件夹大小**

**7.kill**

**kill是linux结束进程的命令，发送指定的信号到相应进程。**

**Kill使用：kill [option] [pid]**

**kill  [pid]          进程在退出之前可以清理并释放资源**

**Kill -9 [pid]  强制杀死该进程，可能会出现数据丢失情况**

**Linux**

/bin /home /dev /proc /mnt /usr /etc /lib

/bin:存储系统所使用命令的可执行文件

/home：普通用户的家目录 (cd home)；

/dev:外部设备（存的是设备的接口 通过接口能访问设备）；

/proc：虚拟目录 以进程为单位存储内存的映射--------一些统计信息；

/mnt：临时挂载点（将接口挂载到临时挂载点进行操作）；

/usr：第三方软件的一些文档；

/etc：系统配置目录；

/lib：库文件（静态库 共享库（动态库））

冯\*诺依曼 五大部件

（计算器 控制器）cpu (存储器)内存 主存------缓存数据（为了使cpu效率提高）（输入设备 输出设备）I/O 硬盘 显示器 键盘

数据从硬盘读到内存 再读到cpu去执行

Linux上一切皆文件：不以文件扩展名区分文件类型（将文件分为五大类）

          普通文件：-（.c     .h     .cpp   .java       .class       .txt     .pdf都属于普通文件）  
        目录文件：d   文件夹  
        链接文件:l(L)  
        管道文件:p （进程间通讯的专用文件）  
        设备文件 ：字符设备(c)           块设备(b)           套接字(s)

ls -l

文件提供给不同用户不同的权限（属主（创建者） 属组（组用户的名称） 其他用户

三种用户 ：三组权限

          u             g         o

r： 读权限

w：写权限

x：执行权限 普通文件（前提是他是可执行文件）

是否可进入   目录文件         针对文件不同，x的含义不同

文件操作命令：

文件创建：普通文件 （touch） 目录文件----文件夹（mkdir）

文件删除：普通文件（rm） 目录文件(rmdir-----删除空目录 rm -r 删除非空目录)

文件拷贝：普通文件（cp原文件路径+文件名 拷贝的目的地） 目录文件(cp -r 原文件路径+文件名 拷贝目的地的路径)把目录里面的文件都拷贝过去

文件剪切/重命名：普通文件（mv） 目录文件（mv）

边移动边重命名：如下图 stu3目录下本来就有一个project1所以剪切过来要重命名。这里的project1和project2是相同的文件

mv /home/stu1/project1 (这里中间要用空格隔开)/home/stu1/project2 单纯的重命名

修改属性：（文件类型一旦创建就不能修改，权限可以修改，链接数不能改，是随着操作变化的；属主可以改；属组可以修改；文件大小不能改，是你往里面加了东西才修改）

修改属主：只能root用户修改，chown newuser（改成谁） filename（所改的文件）

修改属组：只能root用户修改，chgrp newgroup（新的组，原来在那个组不用关心） filename

修改权限：chomd

1、用字符方式修改 chmod a（all所有的）/u/g/o +/-/=(赋值) r/w/x/rw/rx/wx/rwx

2、用数字方式修改：

每个n代表一个用户的权限，一共有3个n,每个n都是0-7的数字

chmod nnn filename

linux基本命令

cd 跟路径 ：切换目录

ls -l ：显示文件详细信息

ls -a ： 显示所有文件 （包含隐藏文件文件名以“.”开头的都是隐藏文件）

cd .. ：可以退回上级目录

cd - ：两个目录之间来回切换

cd ~ ：直接返回家目录

pwd：查看当前目录的绝对路径/查看当前位置

linux怎么查看端口 答：支支吾吾

linux怎么查看内存占用情况 答：支支吾吾

ping这个命令的底层协议是什么 答：tcp。

使用的是[ICMP](http://www.so.com/s?q=ICMP&ie=utf-8&src=internal_wenda_recommend_text)协议，是“Internet Control Message Protocol”（Internet控制消息协议）的缩写，是[TCP/IP协议族](http://www.so.com/s?q=TCP%2FIP%E5%8D%8F%E8%AE%AE%E6%97%8F&ie=utf-8&src=internal_wenda_recommend_text)的一个子协议，用于在IP主机、[路由器](http://www.so.com/s?q=%E8%B7%AF%E7%94%B1%E5%99%A8&ie=utf-8&src=internal_wenda_recommend_text)之间传递控制消息。控制消息是指网络通不通、主机是否可达、路由是否可用等网络本身的消息。这些控制消息虽然并不传输用户数据，但是对于用户数据的传递起着重要的作用。  
它是用来检查网络是否通畅或者网络连接速度的命令。它所利用的原理是这样的：利用网络上机器IP地址的唯一性，给目标IP地址发送一个数据包，再要求对方返回一个同样大小的数据包来确定两台网络机器是否连接相通，时延是多少。

tcp建立连接 答：三次握手四次挥手

tcp的回执是什么包 答：不记得了。

传输控制协议（Transmission Control Protocol, TCP）

TCP协议主为了在主机间实现高可靠性的包交换传输协议。本文将描述协议标准和实现的一些方法。因为计算机网络在现代社会中已经是不可缺少的了，TCP协议主要在网络不可靠的时候完成通信，对军方可能特别有用，但是对于政府和商用部门也适用。  
  TCP是面向连接的端到端的可靠协议。它支持多种网络应用程序。TCP对下层服务没有多少要求，它假定下层只能提供不可靠的数据报服务，它可以在多种硬件构成的网络上运行。下面的图是TCP在层次式结构中的位置，它的下层是IP协议，TCP可以根据IP协议提供的服务传送大小不定的数据，IP协议负责对数据进行分段，重组，在多种网络中传送。

TCP的上面就是应用程序，下面是IP协议，上层接口包括一系列类似于操作系统中断的调用。对于上层应用程序来说，TCP应该能够异步传送数据。下层接口我们假定为IP协议接口。为了在并不可靠的网络上实现面向连接的可靠的传送数据，TCP必须解决可靠性，流量控制的问题，必须能够为上层应用程序提供多个接口，同时为多个应用程序提供数据，同时TCP必须解决连接问题，这样TCP才能称得上是面向连接的，最后，TCP也必须能够解决通信安全性的问题。

网络环境包括由网关（或其它设备）连接的网络，网络可以是局域网也可以是一些城域网或广域网，但无论它们是什么，它们必须是基于包交换的。主机上不同的协议有不同的端口号，一对进程通过这个端口号进行通信。这个通信不包括计算机内的I/O操作，只包括在网络上进行的操作。  
  网络上的计算机被看作包传送的源和目的结点。特别应该注意的是：计算机中的不同进程可能同时进行通信，这时它们会用端口号进行区别，不会把发向A进程的数据由B进程接收的。

进程为了传送数据会调用TCP，将数据和相应的参数传送给TCP，于是TCP会将数据传送到目的TCP那里，当然这是通过将TCP包打包在IP包内在网络上传送达到的。  
  接收方TCP在接收到数据后会通信上层应用程序，TCP会保证接收数据顺序的正确性。虽然下层协议可能不会保证顺序是正确的。这里需要说明的是网关在接收到这个包后，会将包解开，看看是不是已经到目的地了，如果没有到，应该走什么路由达到目的地，在决定后，网关会根据下一个网络内的协议情况再次将TCP包打包传送，如果需要，还要把这个包再次分成几段再传送。  
  这个落地检查的过程是一个耗时的过程。从上面，我们可以看出TCP传送的基本过程，当然具体过程可能要复杂得多。

在实现TCP的主机上，TCP可以被看成是一个模块，和文件系统区别不大，TCP也可以调用一些操作系统的功能，TCP不直接和网络打交道，控制网络的任务由专门的设备驱动模块完成。  
  TCP只是调用IP接口，IP向TCP提供所有TCP需要的服务。通过下图我们可以更清楚地看到TCP协议的结构。

上面已经说过了，TCP连接是可靠的，而且保证了传送数据包的顺序，保证顺序是用一个序号来保证的。响应包内也包括一个序列号，表示接收方准备好这个序号的包。  
  在TCP传送一个数据包时，它同时把这个数据包放入重发队列中，同时启动记数器，如果收到了关于这个包的确认信息，将此包从队列中删除，如果计时超时则需要重新发送此包。请注意，从TCP返回的确认信息并不保证最终接收者接收到数据，这个责任由接收方负责。

每个用于传送TCP的通道都有一个端口标记，因为这个标记是由每个TCP终端确定的，因此TCP可能不唯一，为了保证这个数值的唯一，要使用网络地址和端口号的组合达到唯一标识的目的，我们称这个为了套接字（Socket），一个连接由连接两端的套接字标识，本地的套接字可能和不同的外部套接字通信，这种通信是全双工的。

通过向本地端口发送OPEN命令及外部套接字参数建立连接，TCP返回一个标记这个连接的名称，以后如果用户需要使用这个名称标记这个连接。为了保存这个连接的信息，我们假设有一个称为传输控制块（Transmission Control Block，TCB）的东西来保存。  
  OPEN命令还指定这个连接的建立是主动请求还是被动等待请求。下面我们要涉及具体的功能了，TCP段以internet数据报的形式传送。IP包头传送不同的信息域，包括源地址和目的地址。TCP头跟在internet包头后面，提供了一些专用于TCP协议的信息。  
  下图是TCP包头格式图：

源端口：16位；

目的端口：16位

序列码：32位，当SYN出现，序列码实际上是初始序列码（ISN），而第一个数据字节是ISN+1；

确认码：32位，如果设置了ACK控制位，这个值表示一个准备接收的包的序列码；

数据偏移量：4位，指示何处数据开始；

保留：6位，这些位必须是0；

控制位：6位；

窗口：16位；

校验位：16位；

优先指针：16位，指向后面是优先数据的字节；

选项：长度不定；但长度必须以字节记；选项的具体内容我们结合具体命令来看；

填充：不定长，填充的内容必须为0，它是为了保证包头的结合和数据的开始处偏移量能够被32整除；

我们前面已经说过有一个TCB的东西了，TCB里有存储了包括发送方，接收方的套接字，用户的发送和接收的缓冲区指针等变量。  
  除了这些还有一些变量和发送接收序列号有关：

发送序列变量

SND。UNA - 发送未确认

SND。NXT - 发送下一个

SND。WND - 发送窗口

SND。UP - 发送优先指针

SND。WL1 - 用于最后窗口更新的段序列号

SND。  
  WL2 - 用于最后窗口更新的段确认号

ISS - 初始发送序列号

接收序列号

RCV。NXT - 接收下一个

RCV。WND - 接收下一个

RCV。UP - 接收优先指针

IRS - 初始接收序列号

下图会帮助您了解发送序列变量间的关系：

当前段变量

SEG。  
  SEQ - 段序列号

SEG。ACK - 段确认标记

SEG。LEN - 段长

SEG。WND - 段窗口

SEG。UP - 段紧急指针

SEG。PRC - 段优先级

连接进程是通过一系列状态表示的，这些状态有：LISTEN，SYN-SENT，SYN-RECEIVED，ESTABLISHED，FIN-WAIT-1，FIN-WAIT-2，CLOSE-WAIT，CLOSING，LAST-ACK，TIME-WAIT和 CLOSED。  
  CLOSED表示没有连接，各个状态的意义如下：

LISTEN - 侦听来自远方TCP端口的连接请求；

SYN-SENT - 在发送连接请求后等待匹配的连接请求；

SYN-RECEIVED - 在收到和发送一个连接请求后等待对连接请求的确认；

ESTABLISHED - 代表一个打开的连接，数据可以传送给用户；

FIN-WAIT-1 - 等待远程TCP的连接中断请求，或先前的连接中断请求的确认；

FIN-WAIT-2 - 从远程TCP等待连接中断请求；

CLOSE-WAIT - 等待从本地用户发来的连接中断请求；

CLOSING - 等待远程TCP对连接中断的确认；

LAST-ACK - 等待原来发向远程TCP的连接中断请求的确认；

TIME-WAIT - 等待足够的时间以确保远程TCP接收到连接中断请求的确认；

CLOSED - 没有任何连接状态；

TCP连接过程是状态的转换，促使发生状态转换的是用户调用：OPEN，SEND，RECEIVE，CLOSE，ABORT和STATUS；传送过来的数据段，特别那些包括以下标记的数据段SYN，ACK，RST和FIN；还有超时，上面所说的都会时TCP状态发生变化。

下面的图表示了TCP状态的转换，但这图中没有包括错误的情况和错误处理，不要把这幅图看成是总说明了。

3。3。 序列号

请注意，我们在TCP连接中发送的字节都有一个序列号。因为编了号，所以可以确认它们的收到。对序列号的确认是累积性的，也就是说，如果用户收到对X的确认信息，这表示在X以前的数据（不包括X）都收到了。  
  在每个段中字节是这样安排的：第一个字节在包头后面，按这个顺序排列。我们需要认记实际的序列空间是有限的，虽然很大，但是还是有限的，它的范围是0到2的32次方减1。我想熟悉编程的一定知道为什么要在计算两个段是不是相继的时候要使用2的32次方为模了。  
  TCP必须进行的序列号比较操作种类包括以下几种：

(a) 决定一些发送了的但未确认的序列号；

(b) 决定所有的序列号都已经收到了；

(c) 决定下一个段中应该包括的序列号。

对于发送的数据TCP要接收确认，处理确认时必须进行下面的比较操作：

SND。  
  UNA = 最老的确认了的序列号；

SND。NXT = 下一个要发送的序列号；

SEG。ACK = 接收TCP的确认，接收TCP期待的下一个序列号；

SEG。SEQ = 一个数据段的第一个序列号；

SEG。LEN = 数据段中包括的字节数；

SEG。  
  SEQ+SEG。LEN-1 = 数据段的最后一个序列号。

请注意下面的关系：

SND。UNA 0

RCV。NXT =0

0

不可接受

>0

>0

RCV。NXT = B SYN 本方序列号是X

2) A B ACK 确认对方序列号

上面的第2步和第3步可以合并，这时可以成为3阶段，所以我们可以称它为三消息握手。  
  这个过程是必须的，因为序列号不和全局时钟关联，TCP也可以有不同的机制选择ISN。接收到第一个SYN的接收方不可能知道这个数据段是不是被延时，除非它记住了在连接上使用的最近的序列号（这通常是不可能的），因此它必须要求发送者确认。

为了保证TCP获得的确认是刚才发送的段产生的，而不是仍然在网络中的老数据段产生的，因此TCP必须在MSL时间之内保持沉默。  
  在本文中，我们假设MSL=2小时，这是出于工程的需要，如果用户觉得可以，他可以改变MSL。请注意如果TCP重新初始化，而内存中的序列号正在使用，不需要等待，但必须确认使用的序列号比当前使用的要大。

如果一台主机在未保留任何序列号的情况下失败，那么它应该在MSL时间之内不发出任何数据段。  
  下面将会这一情况进行说明。TCP的实现可以不遵守这个规定，但是这会造成老数据被当成新数据接收，而新数据被当成老数据拒绝的情况。

每当数据段形成并进入输出队列，TCP会为它指定序列空间中的一个值。TCP中多复本检测和序列算法都依赖于这个地址空间，在对方发送或接收之前不会超过2的32次方个包存在于输出队列中。  
  所有多余的数据段都会被删除。如果没有这个规定，会出现多个数据段被指定同一个序列号的情况，会造成混乱。数据段中序列号的多少和数据段中的字节数一样多。

在通常情况下，TCP保留下一个要发送的序列号和还未确认的最老的序列号，不要在没有确认的时候就再次使用，这样会有些风险，也正是因为这样的目的，所以序列空间很大。  
  对于2M的网络，要4。5小时来耗尽序列空间，因为一个数据段可能的最大生存时间也不过十几分之一秒，这就留下了足够的空间；而在100M的网络上需要5。4分钟，虽然少了点，但也可以了。

如果在实现TCP时没有为保存序列号留下空间，那清除多余的包可能就不能实现了，因此推荐这种类型的TCP实现最好在失败后等待MSL时间，这样保证多余的包被删除。  
  这种情况有时候也可能会出现在保留序列号的TCP实现中。如果TCP在选择一个另一个TCP连接正在使用的序列号时，这台主机突然失败了，这就产生了问题。这个问题的实质在于主机不知道它失败了多久，也不知道多余的复本是不是还在网络中。

处理这种问题的方法是等待MSL时间，如果不这样就要冒着对方错误接收数据的危险，要等待的时间也就称为“沉默时间”。  
  实现者可以让用户选择是不是等待，但是无论用户如何也不见得非要等待MSL时间。

3。4。 建立一个连接

建立连接应用的是三消息握手。如果双方同时都发送SYN也没有关系，双方会发现这个SYN中没有确认，于是就知道了这种情况，通常来说，应该发送一个"reset"段来解决这种情况。  
  三消息握手减少了连接失败的可能性。下面就是一个例子，在尖括号是的就是数据段中的内容和标记。其它的就不多说了。

在第2行，TCP A发送SYN初始化序列号，表示它要使用序列号100；第3行中，TCP B给出确认，并且期待着A的带有序列号101的数据段；第4行，TCP A给出确认，而在第5行，它也给出确认，并发送了一些数据，注意第4行的序列号与第5号的一样，因为ACK信息不占用序列号空间内的序列号。  
  同时产生请求的情况如下图所示，只复杂一点。

使用三消息握手的主要原因是为了防止使用过期的数据段。为了这个目的，必须引入新的控制消息，RESET。如果接收TCP处理非同步状态，在接收到RESET后返回到LISTEN状态。如果TCP处理下面几种状态ESTABLISHED，FIN-WAIT-1，FIN-WAIT-2，CLOSE-WAIT，CLOSING，LAST-ACK，TIME-WAIT时，放弃连接并通过用户。  
  我们下面就详细说明后一种情况。

通过上面的例子，我们可以看出TCP连接是如何从过期数据段的干扰下恢复的。请注意第4行和第5行中的RST（RESET信号）。

半开连接和其它非正常状态

如果一方在未通过另一方的情况下关闭连接，或双方虽然失败而不同步的情况我们称为半开连接状态。  
  在一方试图发送数据时连接会自动RESET。然而这种情况毕竟属于不正常情况。应该做出相应的处理。如果A处的连接已经关闭，B处并不知道。当B希望发送数据到A时，就会收到RESET信号，表示这个TCP连接有误，要中止当前连接。

假设A和B两个进程相互通信的时候A的TCP发生了失败，A依靠操作系统支持TCP的存在，通常这种情况下会有恢复机制起作用，当TCP重新恢复的时候，A可能希望从恢复点开始工作。  
  这样A可能会试图OPEN连接，然后在这个它认为还是打开的连接上传送数据，这时A会从本地（也就是A的）TCP上获得错误消息“未打开连接”。A的TCP将发送包括SYN的数据段。下面的例子将显示这一过程：

上面这个例子中，A方收到的信息并没有确认任何东西，这时候A发现出了问题，于是发送了RST控制信息。  
  另一种情况是发生在A失败，而B方仍然试图发送数据时，下面的例子可以表示这种情况，请注意第2行中A对B发送来的信息不知所云。

在下面的例子中，A方和B方进行的被动连接，它们都在等待SYN信息。过期的包传送到B方使B回应了，而收到回应的A却发现不对头，传送RST控制信息，B方返回被动LISTEN状态。

现实中的情况太多了，我们列举一些产生RST控制信息的规则如下：通常情况下，RST在收到的信息不是期待的信息时产生。如果在不能确定时不要轻易发送RST控制信息。下面有三类情况：

如果连接已经不存在，而发送来的消息又不是RST，那么要返回RST。  
  如果想拒绝对不存在的连接进行SYN，可以使用这种办法。如果到达的信息有一个ACK域，返回的RST信息可以从ACK域中取得序列号，如果没有这个域，就把RST的序列号设置为0，ACK域被设备为序列号和到达段长度之和。连接仍然处于CLOSE状态。

如果连接处于非同步状态（LISTEN，SYN-SENT，SYN-RECEIVED），而且收到的确认是对未发出包的确认或是接收到数据段的安全级别与不能连接要求的相一一致时，就发送RST。  
  如果SYN未被确认时，而且收到的数据段的优先级比要求的优先级要高，那么要么提高本地优先级（得事先征得用户和系统的许可）要么发送RST；如果接收数据段的优先级比要求的优先级低，就算是匹配了，当然如果对方发现优先级不对提高了优先级，在下一个包中提高了优先级，这就不算是匹配了。  
  如果连接已经进入SYN，那么接收到数据段的优先级必须和本地优先级一样，否则发送RST。如果到达的信息有一个ACK域，返回的RST信息可以从ACK域中取得序列号，如果没有这个域，就把RST的序列号设置为0，ACK域被设备为序列号和到达段长度之和。  
  连接仍然处于与原来相同的状态。

如果连接处于同步状态（ESTABLISHED，FIN-WAIT-1，FIN-WAIT-2，CLOSE-WAIT，CLOSING，LAST-ACK，TIME-WAIT），任何超出接收窗口的序列号的数据段都产生如下结果：发出一个空确认数据段，此段中包括当前发送序列号，另外还包括一个确认指出希望接收的下一个数据段的序列号，连接仍然保存在原来的状态。  
  如果因为安全级，优先级之类的问题，那就发送RST信号然后进入CLOSED状态