Linux其他网络知识

2019年8月30日 11:41

通过主机名查看IP：

bash

host www.baidu.com

返回的结果中有该域名的IP地址。

远程拷贝：

从本机拷贝数据到远程的服务器上

要求：必须知道对方的账户和密码，且具备相应的权限。

语法：scp [-r] [path]/fie | dir {UserName}@Host_IP:/[path]

-r	该选项用于传输文件夹的时候使用。

案例：

将本机的文件拷贝到远程服务器上

bash

scp /root/install.log root@192.168.89.128:/home

注意， 如果是第一次访问该服务器，那么会询问，是否要继续连接。每次访问都需要输入远程服务器的密码。

从远程服务器上拷贝数据到本机：

要求：必须知道对方的账户和密码，且具备相应的权限。

语法：scp {UserName}@Host_IP:/[path]/file /[path]

案例：

将远程服务器中/home目录下的install.log拷贝到本地的root目录下

bash

scp root@192.168.89.128:/home/install.log /root

通过主机名进行远程copy数据

1.修改/etc/sysconfig/network将HOSTNAME修改为主机名 注：这个修改是永久修改主机名需要重启才能生效

2.临时修改：hostname 主机名

3、想让服务器直到其他服务器的ip以及主机名的对用关系，需要配置/etc/hosts文件。将ip地址和主机名做对应即可

登录远程服务器：

语法：ssh {UserName}@Host_IP

回车之后，如果首次访问，会提示是否继续连接。接下来要求输入远程服务器的密码。

案例：

```
# bash
    ssh root@192.168.89.128
```

如果想退出当前登录

```
# bash
    exit
```

Linux系统下，ssh服务的默认端口是22。如果在访问是没有指明端口，默认按照22端口访问，如果远程服务器，提供的端口不是22，那么就需要在访问的时候指定远程服务器的端口：

```
# bash
    ssh [-p port] {UserName}@Host_IP
```

SSH免密登录

2019年8月30日 12:05

Linux免密登录使用的RSA算法。

RSA本身是一种非对称加密算法，会生成公钥和私钥。

公钥	使用公钥对内容进行加密	天王盖地虎
私钥	持有私钥的PC才能正常访问公钥加密的内容	曹洋一米五

只要持有私钥就能访问公钥加密的内容，这种事情本身就是存在风险的。一旦私钥丢失，那么服务器上的数据就存在被窃取的风险。

但是Linux生成公钥和私钥的时候**支持**对私钥证书文件添加密码。

证书使用场景：

场景一：只是单纯的使用证书来登录服务器。

使用证书的登录方式可以避免密码遗忘、泄漏的问题。

使用证书登录服务器的方式也是服务器加固(服务器安全相关问题)的方式。

服务器可以设置不允许使用密码进行远程登录。只允许证书的方式登录。

证书本身支持加密，就算证书丢失，再不知道证书密码的情况，证书属于无效文件。

场景二：集群中使用证书进行免密登录。

因为但凡设计到集群的时候，一般都不会是小数目的服务器数量。众多的服务器之间进行互相访问，频繁的输入密码的事情将会成为开发工程师噩梦。

所以，使用证书管理集群的时候，可以免除集群中的服务器互相访问时工程师手工输入密码的问题。

证书的生成：

```
# bash
    ssh-keygen
```

第一次提示：你的证书文件存放位置

第二次提示：对私钥加密，输入密码。如果不需要输入密码，直接回车。

第三次提示：私钥证书的密码确认操作。

证书文件会存放在当前账户的家目录下的隐藏目录".ssh"目录下，在该目录下会有以下4个文件：

id_rsa	私钥	执行证书生成命令才会有
id_rsa.pub	公钥	执行证书生成命令才会有
known_hosts	曾经访问过的服务器信	每次ssh、scp、ssh-copy-id到远程服务器时就会保存记录到此文件中，以后再此访问该服务器时就不会再提示那一句"你

	息	确定要继续访问吗 yes/NO？"
authorized_keys	记录来访服务器的公钥文件内容	该文件会记录访问本机的远程服务器的公钥证书文件内容，只有对应的私钥才能进行验证。

证书注册：

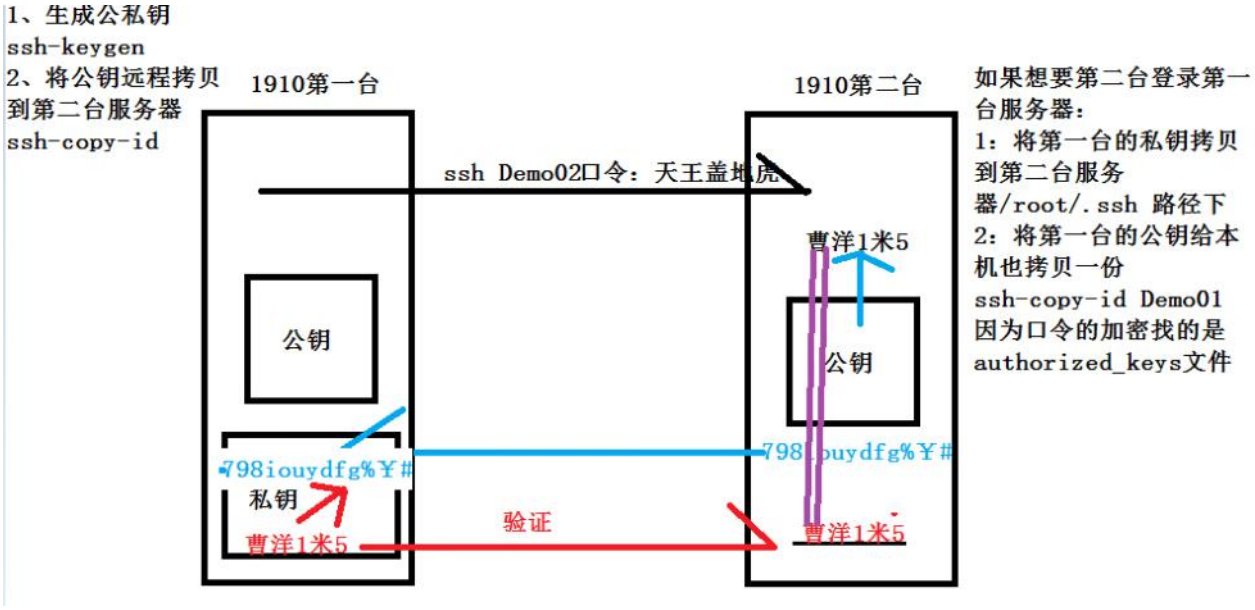
将本机的公钥证书文件注册到远程服务器上，此后就可以使用私钥证书进行登录。

bash

ssh-copy-id {UserName}@Host_IP

执行此命令，会要求输入远程服务器的对应账户的密码。

这一步就是向远程服务器注册本机的id_rsa.pub文件(公钥)内容。此后只有本机上与公钥文件共同生成的私钥才能够进行免密登录。



Linux的自带下载工具

2019年8月30日 15:02

wget

用于从网络上下载资源，没有指定目录，下载资源默认存储到当前目录。

wget [参数] [URL地址]

–支持断点下载功能

–同时支持FTP和HTTP下载方式

–支持代理服务器

使用wget下载单个文件

wget http://www.tedu.cn

使用wget -O下载并以不同的文件名保存

wget -O NewName.new http://www.tedu.cn

使用wget --limit-rate限速下载（单位，byte/秒）

wget --limit-rate=300k http://mirrors.hust.edu.cn/apache/httpd/httpd-2.2.34.tar.bz2

使用wget -c断点续传

wget -c http://mirrors.hust.edu.cn/apache/httpd/httpd-2.4.25.tar.bz2

使用wget -b后台下载

wget -b http://mirrors.hust.edu.cn/apache//httpd/mod_fcgid/mod_fcgid-2.3.9.tar.gz

wget http://mirrors.hust.edu.cn/apache//httpd/mod_fcgid/mod_fcgid-2.3.9.tar.gz

使用wget -i下载多个文件

wget -i urlfile.txt　　# urifile文件名称仅仅为了见名知意。

urlfile.txt内容为

http://www.tedu.cn

http://big.tedu.cn/index.html

进程

2019年8月30日 15:06

概念：

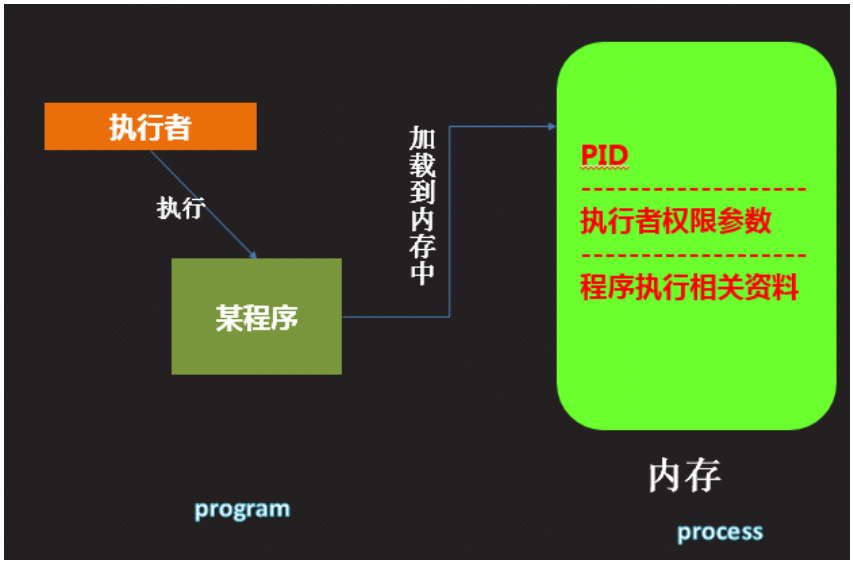
进程通俗来说，运行中的程序，

在linux下，知道程序要运行，首先就是将磁盘中相应的可执行文件加载到内存中，那么我们怎么知道他在内存中哪呢？这个时候就需要我们通过一个叫做进程标识符的东西找到它。类似于我们自己的身份证

进程分为临时进程和持久进程

临时进程	执行完命令，自动结束
持久进程/守护进程	程序运行后，需要手动结束。

程序被加载为进程的示意图：



1、用户执行程序。

2、程序加载到内存中。

2.1、给程序一个临时的pid

2.2、查看执行的权限，如果用户没有执行权限，那么拒绝操作，如果有，开始加载程序执行的相关资料（内存指针开始扫描相应的数据或者代码）

2.3、确认临时的PID。

如何查看进程：

在Linux系统中，不像windows那样方便，可以通过快捷键调出图形化的任务管理器来管理进程。

静态查询：

- ps:

  将某个时间点的程序运作情况截取下来

  常用组合选项 -aux

a	关联的所有 process，通常与x一起使用，列出完整信息。
x	后台进程
u	有效使用者的相关联的进程
ajxf	可以让ps的结果以树状的格式显示出来。

举例：当我们新开一个通道编辑一个文件的时候，我们通过ps -aux可以查看到

查询特定进程用

ps -aux | grep sshd

查询sshd服务

ps查询结果各项解释：

USER	用户
PID	进程ID
%CPU	cpu占用率
%MEM	内存使用率
VSS	虚拟内存使用量
RSS	物理内存使用量
TTY	tty1-tty6 是本机上面的登入者程序。 pts/0 等等的,则表示为由网络连接进主机的程序。 如果显示？则表示与终端机无关。
STAT	进程的状态
START	进程启动的时间
TIME	累计消耗CPU的时间
COMMAND	表示哪个命令/程序运行的该进程

状态标识：

R	正在运行，或在队列中的进程
S	处于休眠状态
I	多进程
Z	僵尸进程
T	停止或者被追踪
<	高优先级
N	低优先级

s	包含子进程
+	位于后台的进程组

僵尸进程：

由于该进程已经执行完毕，但是父进程没有终止或其他原因导致该进程并没有真正的结束，所形成的进程称之为僵尸进程。

此进程对服务器的危害在于它会持续的消耗服务器资源，消耗量会越来越大。最终导致其他的进程无资源可用，服务器崩溃。

pstree：

选项：

A	各程序之间的连接以ASCII字符来连接
U	各程序之间的连接以UTF-8的字符来连接
u	列出每个process的所属账号名称
p	同时列出每个程序的进程的ID

举例：

#bash

- pstree -up

注：使用哪个账户运行此命令，那么与其相关的进程则不会显示用户名

动态查询：

★ top：

动态查询系统的进程状态。默认是3秒一更新。

选项：

-d	跟时间，可以修改top默认更新(刷新)的时间
-b：	以批次的方式执行 top ,还有更多的参数可以使用,通常会搭配数据流重导向来将批次的结果输出成为档案;
-n Number：	与 -b 搭配,意义是需要进行几次 top 的输出结果;
-p：	指定某些个 PID 来进行观察监测而已;

案例：

每秒刷新一次top

bash

top -d 1

每2秒刷新一次top，以批次输出2次。

#bash

top -d 2 -n 2

每秒刷新一次top，以批次输出5次。

bash

top -d 1 -b -n 5 >> top.log # >>表示以追加的方式输出，>表示以覆盖的方式输出

交互式按键：(并不常用)

? :	显示在 top 当中可以输入的按键指令
P :	以CPU的使用资源排序显示
M :	以Memory的使用资源排序显示
N :	以PID来排序
T :	由该Process使用的CPU时间累积 (TIME+) 排序
q :	离开top软件的按键

★ 进程的管理：

单进程的管理：

kill：结束某个进程

语法：kill 信号量 PID

信号量：

-15:	以正常的程序方式终止一个进程！！！
-9 :	立刻强制终止一个进程！！！（ ！！不能强制结束系统级别的进程 ）
-2 :	代表由键盘输入 [ctrl] + c 同样的动作;
-1 :	对于sshd这样的守护进程，重新读取一次参数的配置文件 (类似 reload)，如果进程为非守护进程，默认为终止进程; ！！

多进程的管理：

killall：结束基于某个程序运行进程。

语法：killall 信号量 程序名/命令名

信号量：

-15:	以正常的程序方式终止一个进程！！！
-9 :	立刻强制终止一个进程！！！（ ！！不能强制结束系统级别的进程 ）
-2 :	代表由键盘输入 [ctrl] + c 同样的动作;
-1 :	对于sshd这样的守护进程，重新读取一次参数的配置文件 (类似 reload)，如果

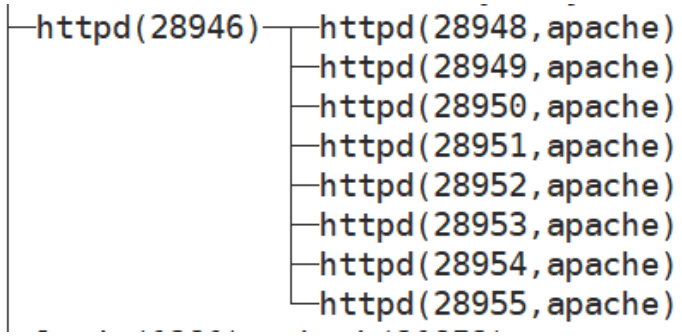
	进程为非守护进程，默认为终止进程; ！！

案例：

结束所有httpd的进程（如果没有可以先安装一下yum install -y httpd）

bash

killall -9 httpd



结束所有java的进程

bash

killall -9 java

系统资源监控(自带)

2019年8月30日 16:29

free：内存监控（可能监控不精准）

选项：

-b	bytes
-k	kb
-m	mb
-g	gb
-t	统计总量

free -m

free -mt

清理缓存的命令：echo 3 > /proc/sys/vm/drop_caches

在真实的公司中有第三方，甚至公司自己开发的监控工具。不属于我们负责

uname:查阅系统与核心相关信息

选项：

-a	所有系统相关的信息,包括以下的数据都会被列出来;
-s	系统内核名称
-r	内核版本
-m	本系统的硬件名称,例如 i686或x86_64 等;
-p	CPU 的类型,与 -m 类似,是显示的是CPU的类型;
-i	硬件的平台(ix86);

系统 32 64

CPU 32 64

主板 32 64

uptime：观察系统启动时间与工作负载

00:29:43 up 1 day, 7:20, 2 users, load average: 0.00, 0.00, 0.00

00:29:43	系统当前的时间
Up 1 day, 7:20	系统运行时间
2 users	当前有两个用户登录
load average: 0.00, 0.00, 0.00	系统过去的1,5,15分钟的平均负载

netstat：网络监控

-a	将目前系统上所有的已经连接、监听、Socket数据都列出来
-t	列出tcp网络包的信息
-u	列出udp网络包的信息
-n	以端口(port number)方式来显示（不以程序的服务名称）
-l	列出目前正在监听(listen)的服务;

-p	列出该网络服务的进程id（PID）、程序名

案例：

列出当前系统中正在监听的TCP服务。

bash

netstat -lt

列出当前系统中正在监听的TCP服务，并且显示进程ID。

bash

netstat -ltp

列出当前系统中正在监听的TCP服务，并且显示进程ID、端口号。

bash

netstat -lntp

列出当前系统中已连接的TCP服务，并显示进程ID、端口号。

bash

netstat -tnp

监听udp一般监听不出来

各项含义：

Proto	协议名
Recv-Q	接收消息缓冲区
Send-Q	发送消息缓冲区
Local Address	本地地址和端口号
Foreign Address	远程地址和端口号
State	状态。连接、监听
PID/Program name	进程ID和程序名

vmstat :侦测系统资源变化

统计目前主机CPU状态,每秒一次,共计四次

[root@tedu ~]# vmstat 1 4

```
procs -----------memory---------- ---swap-- -----io---- --system-- -----cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 0  0      0 662856  54876 151320    0    0     2     1   10   10  0  0 100  0  0
 0  0      0 662816  54876 151320    0    0     0     0   17   10  0  0 100  0  0
 0  0      0 662816  54876 151320    0    0     0     0   20   12  0  0 100  0  0
 0  0      0 662816  54876 151320    0    0     0     0   13   10  0  0 100  0  0
```

- procs（进程字段 ）

r :	等待运行的进程数量; cup处理不过来
b:	不可被唤醒的进程数量

这两个项目越多,代表系统越忙碌 (因为系统太忙,所以很多进程就无法被执行或一直在等待而无法被唤醒)

- memory (内存字段)

swpd:	虚拟内存被使用的容量;
free:	未被使用的内存容量;
buff/cache:	用于缓冲的内存;

- swap（交换分区字段） （重点记忆下si和so）

si:	每秒从交换分区写到内存的数据量大小，由磁盘->内存;
so:	每秒写入交换分区的内存数据量大小，由内存->磁盘。

如果si/so的数值太大,表示内存内的数据常常得在磁盘与主存储器之间传来传去,系统效能会很差

- io(磁盘读写字段)

bi:	从块设备读入数据的总量（读磁盘）（每秒kb）;
bo:	从块设备写入数据的总量（写磁盘）（每秒kb）。

如果这部份的值越高,代表系统的I/O非常忙碌

- system（系统字段 ）

in:	每秒被中断的进程次数; 发生在cup争抢的过程中
cs:	每秒钟进行的事件切换次数。发生在cup争抢的过程中

这两个数值越大,代表系统与接口设备的通信非常频繁

- CPU (cpu字段)

us:	(user)非内核态的（用户进程） CPU 使用情况;
sy:	(system）内核态所使用（系统进程）的 CPU 情况;
id:	(idle ）闲置的CPU情况;
wa:	(wait)等待I/O所耗费的CPU;
st:	被虚拟机(virtual machine)所盗用的CPU(2.6.11 以后才支持)

Linux防火墙：

它具备一定的防护功能，比如说端口的开放和禁止，也可做数据的转发(类似路由功能)，策略及其他功能。

临时处理防火墙：如果系统重启，那么防火墙将恢复到之前的状态。

开启	service iptables start or /etc/init.d/iptables start
关闭	service iptables stop or /etc/init.d/iptables stop
重启	service iptables restart or /etc/init.d/iptables restart
查看	service iptables status or /etc/init.d/iptables status

永久处理防火墙：(需重启系统后才能生效)

开启：	chkconfig iptables on
查看状态	chkconfig iptables --list
关闭：	chkconfig iptables off

任务管理

2019年8月30日 17:07

概念：

前台任务：可以控制与执行命令的bash环境称为前台。

后台任务：在操作系统中自行运行,你无法使用[ctrl]+c终止称为后台。

管理：

将前台任务放置后台暂停：

Ctrl + z就可以将前台的任务放置后台

如何运行任务时，使其在后台运行：

在运行命令之前加上"&"

例如：

cp file1 file2 &

不是所有的任务都能够在后台运行的，比如需要与用户进行交互的程序或命令就不允许在后台运行， 比如vi文本编辑器

用来查看**后台任务**：

jobs

选项：

-r	仅查看后台运行的任务
-s	仅查看后台暂停的任务
-l	查看后台的任务，并显示其PID

如果将后台任务调至前台：

fg命令+jobnumber来把后台任务调至前台。(无论在后台是暂停还是运行)

fg命令不加jobnumber也是可以调后台的任务，但是默认就会调取后台带有+号的那个任务。最后放置后台的任务就会带有+号。

+	表示最近一次放置后台的任务

案例：

```
[root@bogon ~]# jobs
[1]   Stopped                 vi 1.txt
[2]-  Stopped                 vi 2.txt
[3]+  Stopped                 vi 3.txt
```

调取2号任务：

bash

fg 2

如何将后台任务修改为运行状态：

bg命令 + jobnumber 可以将后台任务的暂停状态修改为运行状态。(交互式的应用无法修改为运行状态)

bg命令不加jobnumber也是可以调后台的任务，但是默认就会调取后台带有+号的那个任务。（最后放置后台的任务就会带有+号。）

案例：

```
[root@bogon ~]# jobs
[1]   Stopped                 vi 1.txt
[2]-  Stopped                 vi 2.txt
[3]+  Stopped                 vi 3.txt
```

修改2号任务的后台状态：

bash

bg 2

失败案例！！

```
[root@bogon ~]# bg 2
[2]- vi 2.txt &
[root@bogon ~]#

[2]+  Stopped                 vi 2.txt
[root@bogon ~]#
```

```
[root@bogon ~]# jobs
[1]   Stopped                 vi 1.txt
[2]-  Stopped                 vi 2.txt
[3]   Stopped                 vi 3.txt
[4]+  Stopped                 cp -i -r /home/cdrom/ /root/home/
```

修改4号任务在后台的工作状态。

bash

bg

```
[root@bogon ~]# bg
[4]+ cp -i -r /home/cdrom/ /root/home/ &
[root@bogon ~]# jobs
[1]-  Stopped                 vi 1.txt
[2]+  Stopped                 vi 2.txt
[3]   Stopped                 vi 3.txt
[4]   Running                 cp -i -r /home/cdrom/ /root/home/ &
[root@bogon ~]#
```

终止job:

jobs -l查询出ID，之后通过kill -9 PID 结束

VIM

2019年8月30日 17:41

什么是VIM：

是一个类似vi的文本编辑器，不过在vi的基础上增加了很多新特性，vim被公认为类vi编辑器中最好用的一个。

为什么要学习VIM，vi不够？

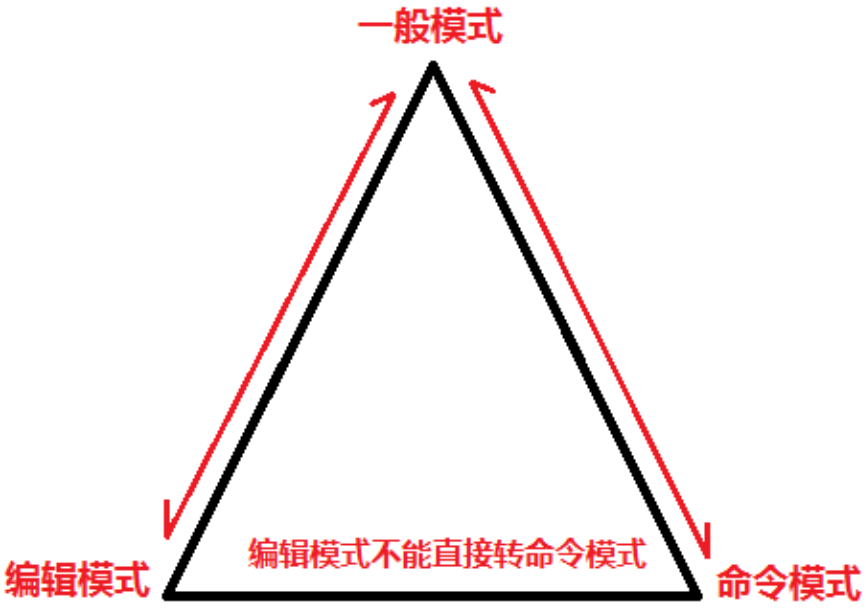
vim在vi的基础之上增加了很多的小功能。可以有效的帮助程序员快速的排查问题。

很多系统都内建vi编辑器，其他的文本编辑器不一定有，很多软件都会主动调用vi的接口。

vim的三种模式：

一般模式、编辑模式、命令模式

三种模式的转换图：



一般模式(默认模式)的快捷键：

h或←光标左移一个字符。如果是20h，表示左移20个字符。

j或→光标下移一个字符　同上

k或↑光标上移一个字符　同上

l或↓光标右移一个字符　同上

[Ctrl]+[f]屏幕向下移动一页　Page Down!!

[Ctrl]+[b]屏幕向上移动一页　Page Up !!

0或[Home]移动到此行最前面字符处!!

$或[End]移到光标所在行的行尾!!

H 光标移到当前屏幕最上方行的第一个字符!!

M光标移到当前屏幕中间行的第一个字符!!

L光标移动到当前屏幕最下方行第一个字符!!

G移到此文件最后一行!!!

nG移到第n行

gg相当于1G，即移到第一行!!!

n[Enter]光标下移n行

/word向下查找单词“word”（！!!）

？word向上查找单词“word”（！!!）

n表示重复前一个查找操作

N与n相反（反向查找）

yy复制光标所在行（！！）

nyy复制光标所在向下n行(n为数字)

y1G复制光标所在行到第一行所有数据

yG复制光标所在行到最后一行所有数据

y$复制光标所在处到同行最后一个字符

y0复制光标所在处到同行第一个字符

p将已复制的数据粘贴到光标所在下一行

P将已复制的数据粘贴到光标所在上一行

u复原前一个操作（类似于windows中的ctrl+z）!!!

Ctrl+r恢复一个操作。

x向后删除一个字符

nx向后删除n个字符(n为数字)

X向前删除一个字符

dd删除光标所在行（！！！）

ndd删除光标所在行以下n行(n为数字,包含当前行在内)

d1G删除光标所在行到第一行所有数据（包括所在的行）

dG删除光标所在行到最后一行（！！）

d$或d end删除光标所在处到同行最后一个字符（！！）

d0或d home删除光标所在处到同行第一个字符。（!!）

编辑模式：

进入编辑模式：

i从光标所在处插入(!!!)

I从光标所在行第一个非空白字符处插入(!!)

a从光标所在处下一个字符处插入

A从光标所在行最后一个字符处插入（！！）

o在光标所在处下一行插入新的一行(!!)

O在光标所在处上一行插入新的一行（！！）

r替换光标所在处字符一次

R一直替换光标所在处文字直到按下Esc(!!!)

命令模式：

如何进入命令模式：

: ? /	三个符号任意都可以进入命令模式

:w [filename] 另存为filename

:r [filename] 读取filename指定文件中的内容到光标所在的行。

:n1,n2 w [filename] 将n1到n2行另存为filename

:! command 临时切换到命令行模式下执行command命令。

例如 ":!find / -name Helllo.java" 即可在vim当中执行命令。

:wq 保存后离开

:q 不保存离开(未改可以离开，如果修改了需要q!强制离开)

:q! 不保存强制离开

:set nu 显示行号 （number）

:set nonu 取消显示行号 （noNumber）

:s/word1/word2/g 在当前行将word1替换成word2（！！）

:%s/word1/word2/g 在当前文件将word1替换成word2（！！）

:n1,n2s/word1/word2/g在n1到n2行查找word1替换成word2 (n1、n2为数字)

:10,$ s/word1/word2/g从第一行到最后一行查找word1替换成word2

:%s/word1/word2/gc 同上，在替换前确认是否替换。（！！！）只能单行确认，需要逐个确认。

替换为 b (y/n/a/q/l/^E/^Y)？

y表示yes，n表示no，a表示all(限光标当前到最后一行)，q表示quit，l表示替换后移动光标到行首，^E(Ctrl+E)表示向下翻，^y(Ctrl+Y)表示向上翻。