Linux基础

## 内存模型

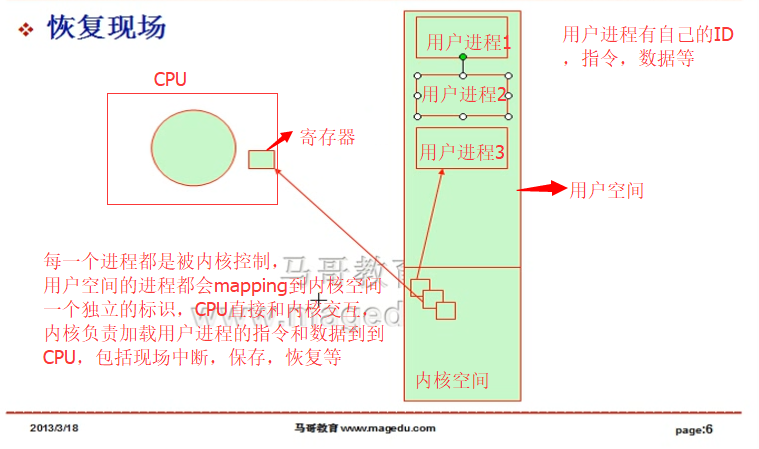
线性内存(逻辑地址)

物理内存

**进程切换，进程中断，现场保护，现场恢复，等都是由内核来管理完成**

内核的数据结构：

task structure



如果执行进程种的指令被执行完，内核就进行销毁，回收空间。

x86的架构

用户进程和系统进程多个同时存在,对于CPU来说,每个进程轮流来使用CPU硬件资源,对于每一个用户进程, 如果需要向磁盘写入,就会进行系统调用,内核就会使用系统进程,对磁盘进行IO操作.

在同一时刻只有一个进程使用CPU资源,进程切换的时候会进行现场保护,进程被执行的时候会根据线程保护进行线程恢复,使用CPU资源

用户进程保存在内存中用户进程区域,用户进程的标识被保存在内存中的系统进程区域中

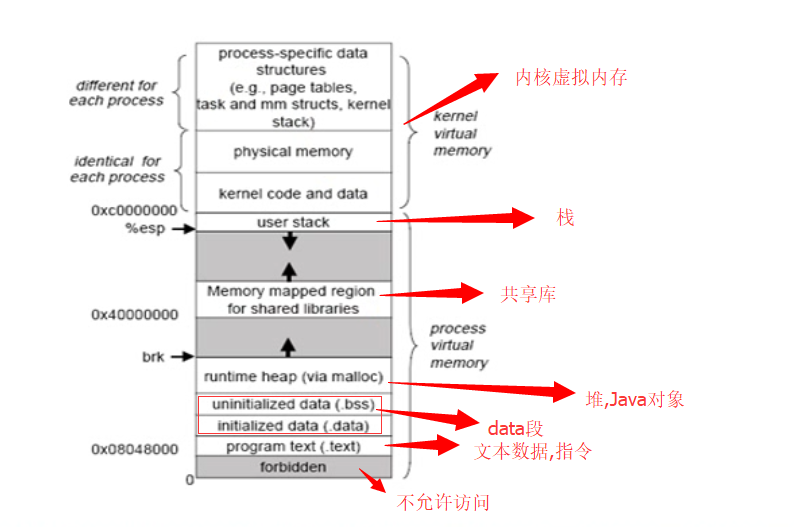
程序执行的时候,系统进程区域的标识保存到CPU的寄存器中

## 内存的逻辑地址和物理地址

内存的物理地址是连续的。

逻辑地址要求内存分页，一个进程使用的内存空间在物理上是不连续的，但是逻辑上是连续的，就像链表数据结构，**最重要就是那张物理地址和逻辑地址的mapping表**

每一个进程都认为自己有4G内存, 内存分为若干页面



program text : 指令是只读的

initiailized data, uninitialized data 数据段

heap，动态变化，需要了就申请，内核就会把一些空闲的地址（内存页）进行mapping,

不用了就进行回收，比如打卡一个world，文本文件，都需要heap来存放，关闭文件的时候，内核进行回收，所以内核帮助完成这一系列动作，还有比如java种的heap

stack 栈，存放本地变量

memory mapped region for shared libraries : 共享库，每个进程都可以使用的

MMU 对逻辑地址和物理地址转换,负责mapping 转换

TLB MMU缓存 调优使用

进程切换也叫上下文切换或者是环境切换 进程切换是很耗费时间的

每一个进程都有一个task structer

**VSZ: 虚拟内存集 要算共享库**

**RSS: 常驻内存集 不能被交换出去的,如指令,变量等核心数据，heap的数据可以被交换出去**

CPU是随着时间的流逝提供的计算能力的

一个进程可以运行在多个CPU上，但是不能同时运行在多个CPU上，也就是说，指令是有顺序的执行，不能同时执行一前一后两条指令，因为第二条指令执行的结果是要依赖第一条指令执行的结果。

## 线程

一个进程如果想要运行在多个CPU上，可以采用并行编程模型来实现。合理均衡的使用CPU硬件资源。将一个进程内部分成多个执行实体，或者多个执行流。是比进程更小的单位，就是线程。这些线程就可以分别执行在各个CPU上。

好处： 可以共享资源文件，相对节省内存空间

合理使用，提高效率，如果太多线程，时间耗费在线程间切换上，同步（锁）上，有死锁的情况。

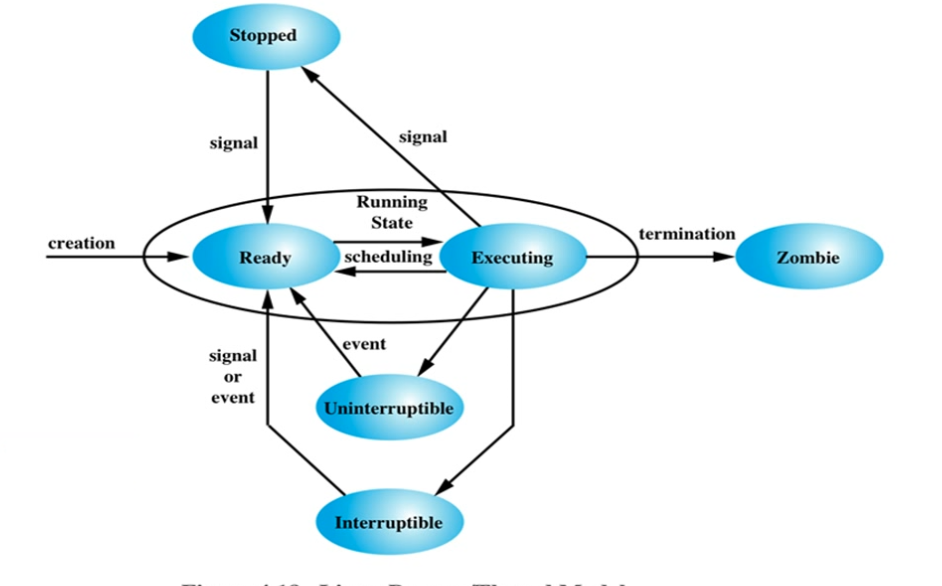
## 进程状态模型

多核CPU是,一个进程只能运行在一个核上,但是也可以运行在多个核上,这需要程序提供多线程的能力,使用并行编程模型(java中的多线程模型)

单进程多线程模型更省内存, 自选锁控制 ,有死锁的情况

（线程内部有竞争条件，和线间的切换）

多进程模型



stopped 停止状态

ready 就绪状态

executing 执行状态

uninterruptible: 不可中断的睡眠 外部IO调用,到磁盘读取数据的这段时间

interruptible: 可中断的睡眠

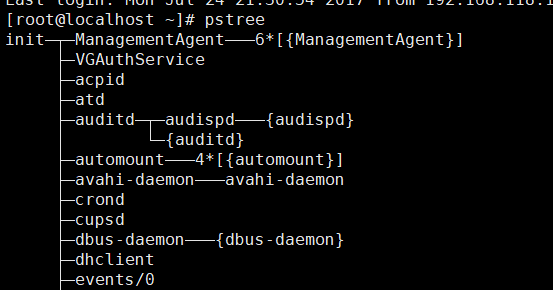
zombie: 僵尸状态,进程不在了,内存就是不释放

## 进程的父子关系

kernel:

init : 上帝进程 所有的进程都是他克隆的

父进程为子进程收尸



## 进程的优先级

优先级 0-139 140个优先级数字越小优先级越高

100-139 用户控制的

0-99 内核调整的

一般来说是要满足优先级高的进程来运行

一个进程的优先级高了意味着

1. 更多的CPU运行时间
2. 更优先获得运行机会

100-139

nice值： 优雅的，友好的

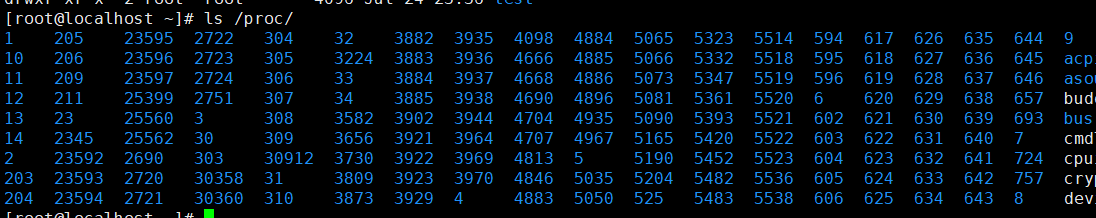
-20 到 19 对应

100 到139

普通用户仅能够调大自己进程的nice值

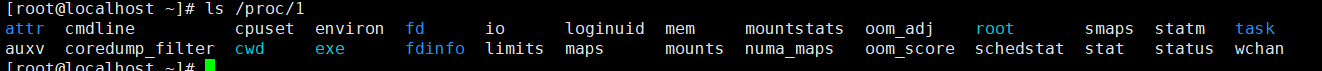
管理员就可以随便调

每一个数字都有一个进程号



init的进程为1 ，这里面都是内核被映射过来的文件，不是真正的文件，是伪文件

这里面都是跟进程相关的属性信息



## 进程的分类

和终端相关的进程

与终端无关的进程

## 进程状态



BSD

D不可中断的睡眠

R 运行或者就绪状态

S 可中断的睡眠

T 标识停止

Z 僵死状态

< 高优先级的进程

N 低优先级进程

+ 前台进程组中的进程

1 多线程进程

s 会话进程首进程

sysv

elF

## ps

ps 默认只显示前台进程

ps : process state

Unix两大阵营

sysv 风格

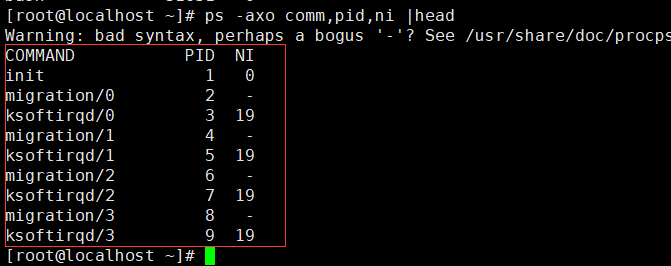
BSD风格

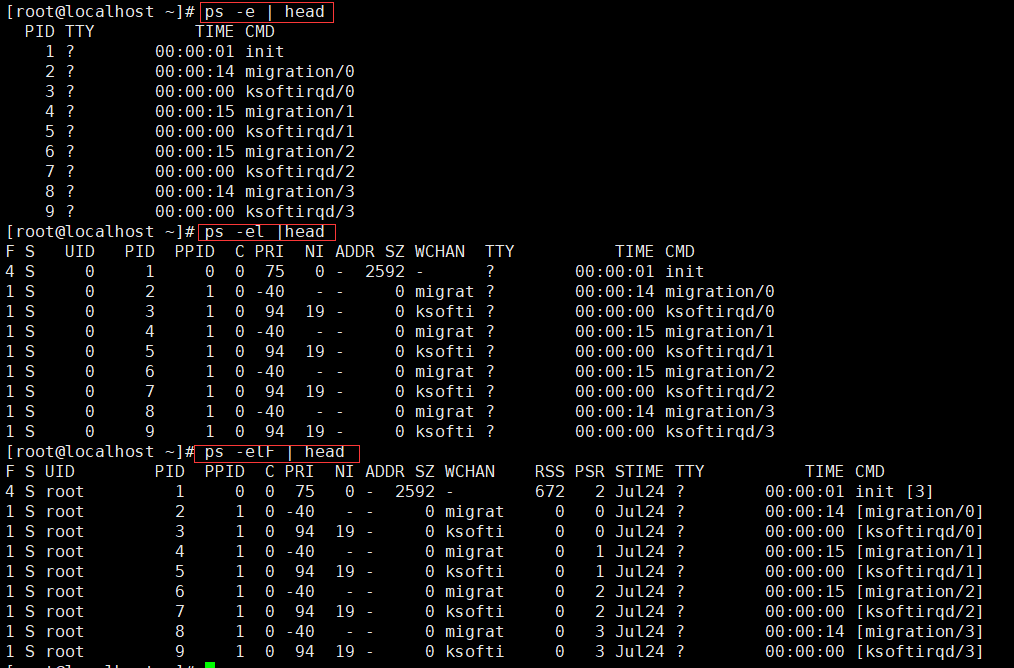
a: 所有与终端相关的进程

u: 用户相关的进程

x: 所有与终端无关的进程

o: 指定显示哪些字段





UID : 用户

PID : 进程ID

PPID：父进程ID

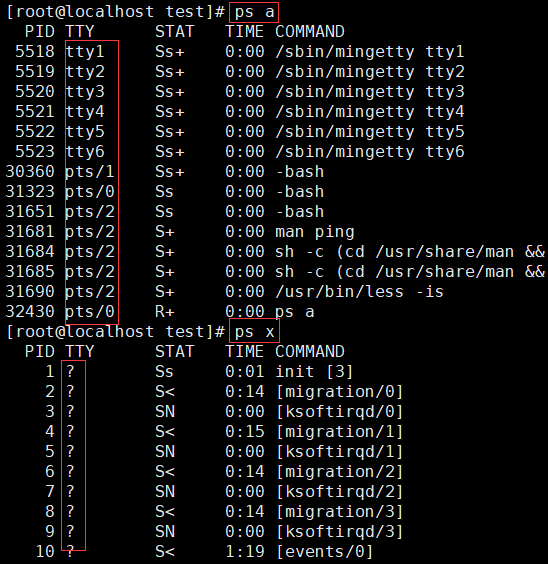
PRI: 优先级

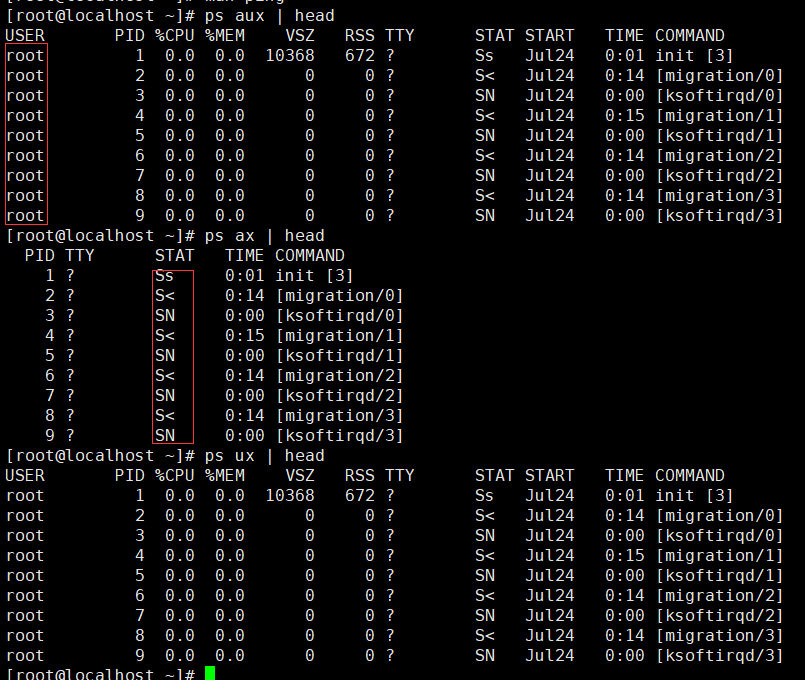
NI：nice值

sz : size值

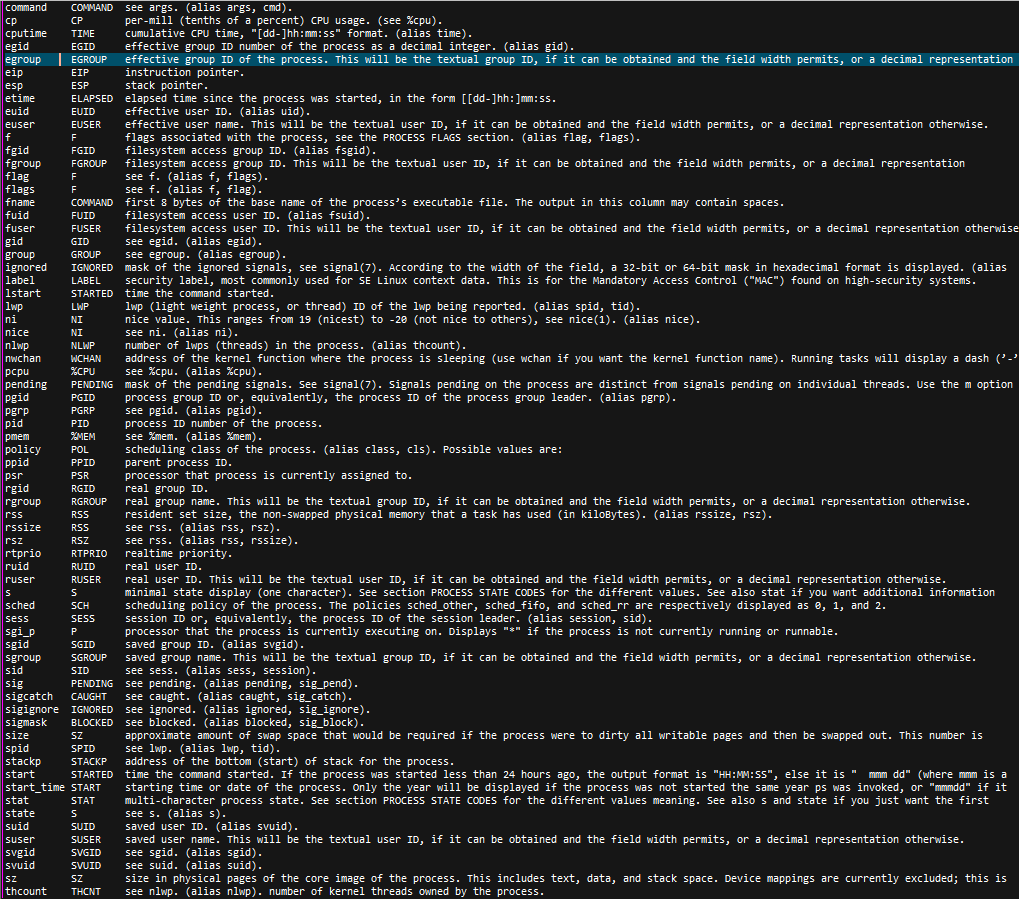
RSS：常驻内存集

PSR：哪个核上运行



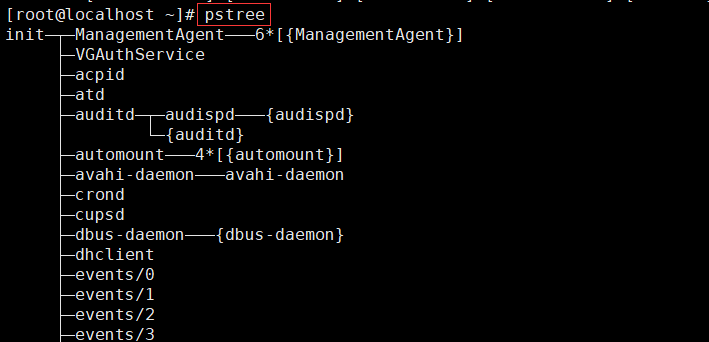


显示的字段



## pstree

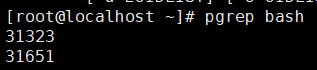
显示进程树

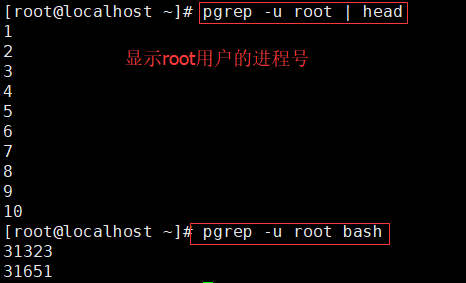


## pgrep

-u

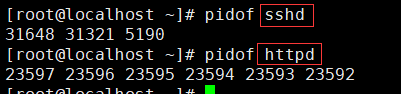
显示属于root用户的进程号





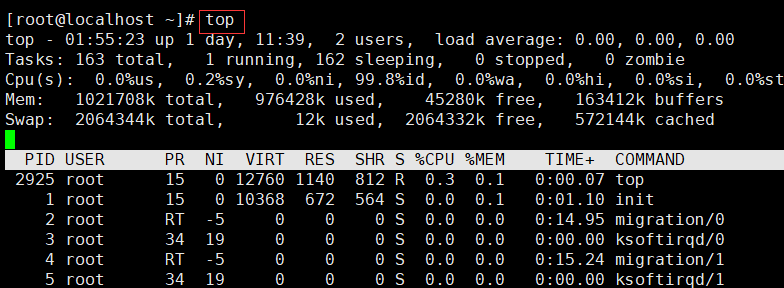
## pidof

根据程序名称，查找其相关进程的ID号



## top

监控系统最好用的命令之一



load average : 平均负载，队列长度过去5分钟 10分钟 15分钟的情况，越小CPU负载越低

Cpu(s) : 多个cpu的平均值，**按 1 键显示多个CPU**

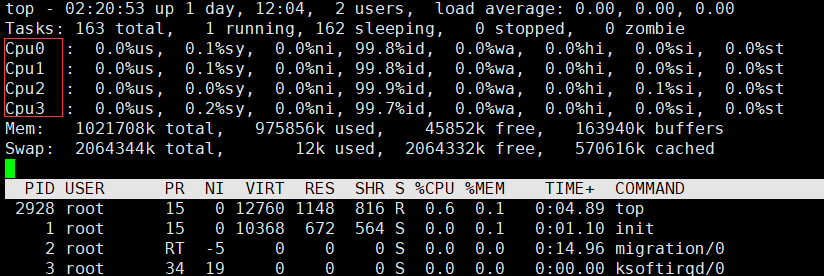
us 用户空间的比例 sy 系统空间的比例

ni 调整nice值以后的比例 id 空闲百分比 wa 等待io完成所占的比例

hi 硬件中断占据的比例 si 软中断占据的比例 st cpu被偷走的时间

Mem 内存情况

Swap 交换空间情况



PR 优先级 RT 实时优先级

NI nice值

VIRT 虚拟内存集

RES 常驻内存集

SHR 共享内存大小

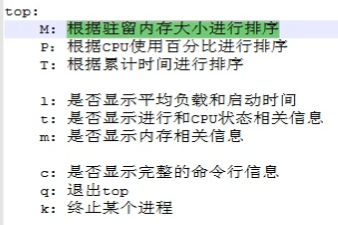
S 状态 R 运行 S 停止

%CPU

%MEM

TIME+ 真正占据CPU时间

交互式子命令



M : 根据驻留内存大小排序

P: 根据CPU使用百分比进行排序

T: 根据累计时间进行排序

l: 是否显示平均负责和启动时间

t: 是否显示进行和cpu状态相关信息

m: 是否显示内存相关信息

c: 是否显示完整的命令信息

q: 退出top

k: 终止某个进程

-d #: 刷新时长，秒

-b : 批处理模式，翻页

-n # : 在批模式下只显示#页

进程间通信（IPC： Inter Process Communication）

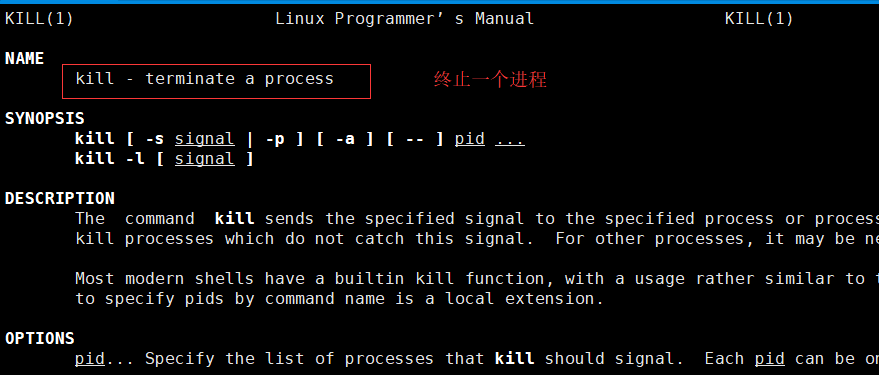
共享内存

信号 ： singnal

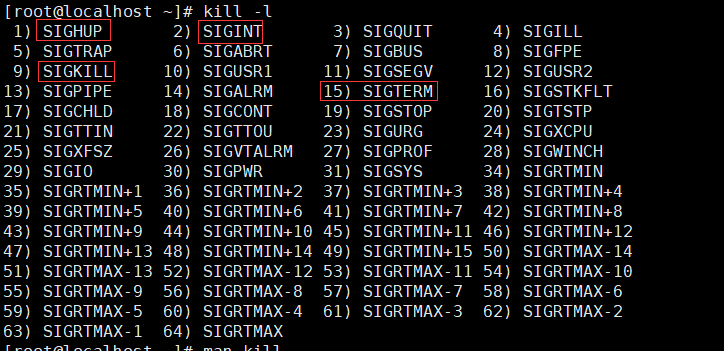
semaphore ： 旗语

## kill

主要是用来终止一个进程，向另外一个进程发信号



显示信号量



1：SIGHUP 一个进程不用重启，就可以读其配置文件,并让新的配置文件生效;

2 : SIGINT CTR+C 中断一个进程

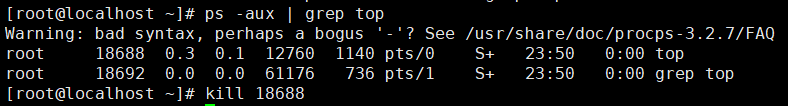
9 : SIGKILL 杀死一个进程 暴力杀死

15 : SIGTERM 终止一个进程（默认使用） 很人道的杀死

指定信号的方式

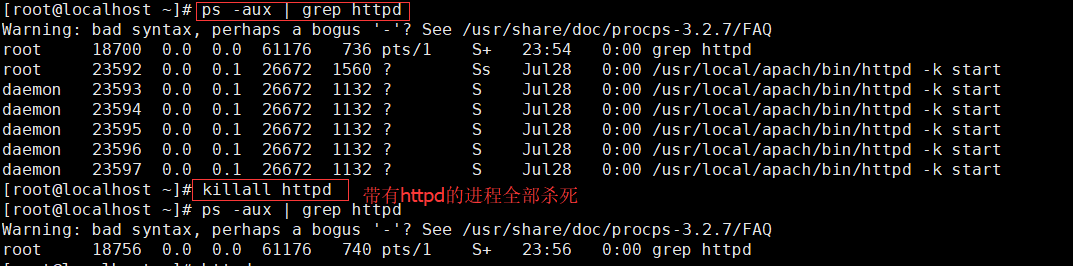
1. 信号号码 kill -1
2. 信号名称 kill -SIGHUP
3. 信号名称简写 Kill –HUP

kill pid



## killall

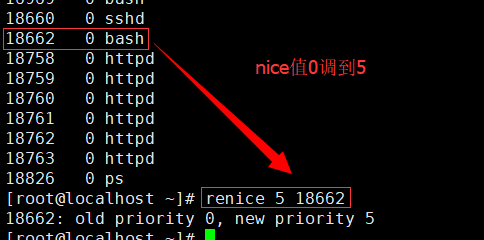
killall 服务command

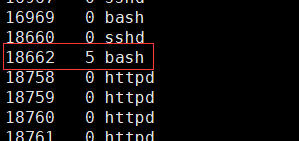


## renice

调整进程的优先级nice值

如果已启动使用 : renice NI PID





在启动时指定nice值

nice –n NI command

## &

前台：占据了命令提示符，

一个命令后面加上 & 直接释放前台

后台：启动了之后,释放了命令提示符，后续的操作在后台完成。

前台-> 后台 :

ctrl+z 把正在前台的作用送到后台，是一个停止的状态，bg可以让其继续执行

command & 让命令在后台执行

## bg

bg 让后台的作业继续执行。

bg 作业号

## jobs

查看后台的所有作业，每一个作业都有一个作业号，

作业号不是进程号，

+ 标识命令将默认操作的作业

- 标识命令将第二个默认操作的作业

## fg

fg将后台从作业调回到前台

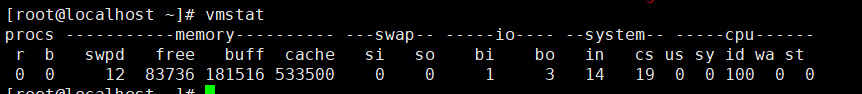
fg %作业号 == kill %作业号，%不能省

## nohup

nohup command & 即使关闭远程终端也可以继续让程序在后台运行

## vmstat

系统状态查看命令



procs 进程相关

r 运行队列长度

b 阻塞队列长度

memory 内存相关

swpd 交换内存大小

free 空闲内存大小

buffer 缓冲大小

cache 缓存大小

swap 动态的交换内存

si 有多少个内存页面从硬盘上（虚拟内存）换进物理内存

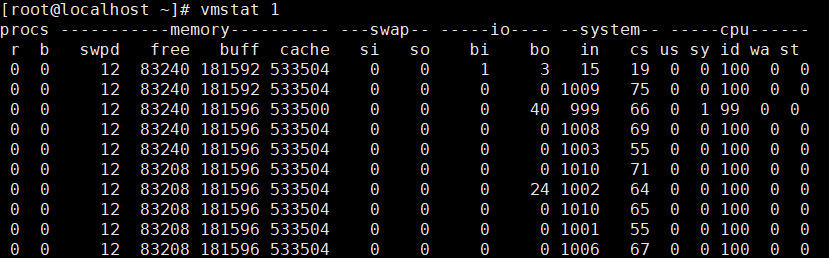
so 有多少个内存页面从物理内存换出到虚拟内存

io

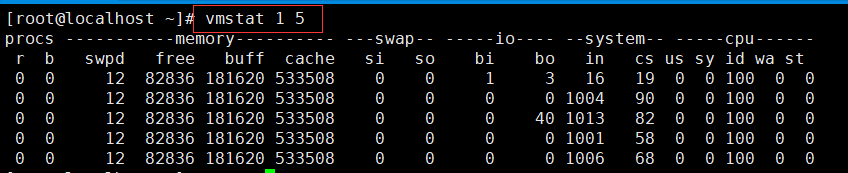
bi 有多少个块进内存

bo 有多少个块写入到磁盘

vmstat 1 每隔1秒刷新



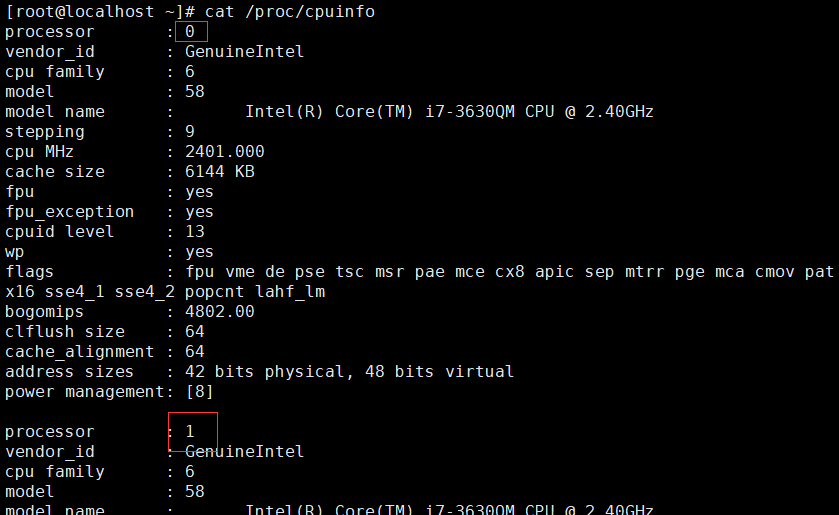
vmstat 1 5 5次



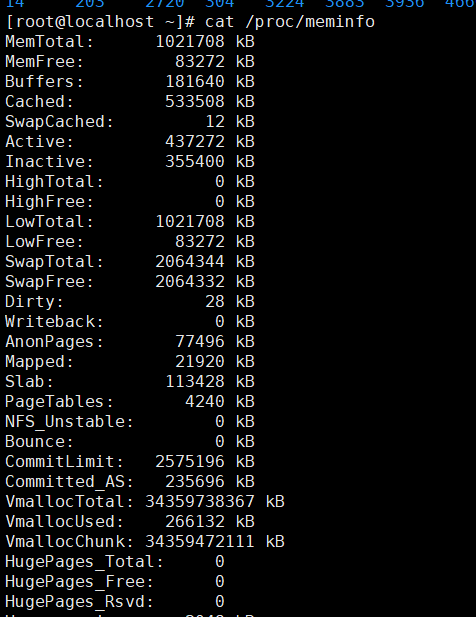
uptime



## cat /proc/cupinfo



## cat /proc/meminfo



## cat /proc/1/maps

