Linux计划任务以及进程检测与控制

一、Linux计划任务

1、计划任务权限

☆ 黑名单

crontab是任何用户都可以创建的计划任务，但是超级管理员可以通过配置来设置某些用户不允许设置计划任务 。

提出问题：如果我们想限定某个用户（如itheima）使用计划任务，如何处理呢?

答：可以使用计划任务的黑名单，黑名单文件路径 => ==/etc/cron.deny==文件

案例：把普通账号itheima加入到cron.deny黑名单中，禁止其创建计划任务

第一步：切换到超级管理员root

```
# su - root
```

第二步：使用vim打开/etc/cron.deny文件

```
# vim /etc/cron.deny
```

第三步：把你需要禁止的用户名单，加入此文件（如itheima）

```
itheima
```

切换到itheima账号，测试是否可以使用crontab命令

```
[itheima@yunwei ~]$ crontab -e
You (itheima) are not allowed to use this program (crontab)
See crontab(1) for more information
[itheima@yunwei ~]$
```

☆ 白名单

在Linux的计划任务中，除了拥有黑名单以外，还有白名单。作用：允许哪些用户使用计划任务。

白名单文件的路径 => ==/etc/cron.allow==，但是要特别注意，此文件需要手工创建。

> 注意：白名单优先级高于黑名单，如果一个用户同时存在两个名单文件中，则会被默认允许创建计划任务。

2、查看计划任务的保存文件

问题：计划任务文件具体保存在哪里呢?

答：`/var/spool/cron/用户名称`，如果使用root用户编辑计划任务，则用户文件名为root

```
# ll /var/spool/cron
total 4
-rw-------. 1 itheima itheima  0 Mar 24 09:50 itheima
-rw-------. 1 root    root    40 Mar 24 10:21 root
```

3、计划任务的日志程序

问题：在实际应用中，我们如何查看定时任务运行情况?

答：通过计划任务日志，日志文件位于`/var/log/cron`

案例：演示计划任务的日志程序

第一步：使用root账号创建一个计划任务

```
# su - root
# crontab -e
* * * * * echo 1 >> ~/readme.txt
```

第二步：使用tail -f命令监控/var/log/cron日志程序

```
# tail -f /var/log/cron
```

4、扩展内容：at命令

在Linux系统下，有两个命令可以实现计划任务：crontab与at（第三方需要额外安装）

crontab ： 每天定时执行计划任务（最小单元分钟）

at ： 一次性定时执行任务

☆ 安装at命令

CentOS7自带，其他版本可能需要手工安装

```
# yum install at -y
```

☆ 启动底层服务

```
# systemctl start atd
# systemctl enable atd
```

> atd = at + d = at命令 + daemon缩写

☆ 案例演示

案例1：三天后下午5点执行/bin/ls

```
# at 5pm+3 days
at>/bin/ls >/root/readme.txt
at>按Ctrl+D
```

> am = 上午、pm = 下午、3 days = 3天

案例2：明天17点，输出时间到指定的文件中

```
# at 17:00 tomorrow
at>date>/root/readme.txt
at>按Ctrl+D
```

> tomorrow = 明天

案例3：使用atq查看没有执行的计划任务

```
# atq
```

> atq = at + q = at命令 + query查询

案例4：删除指定的计划任务

```
# atq
# atrm 任务号
```

> atrm = at + rm = at命令 + remove移除

二、Linux进程与程序

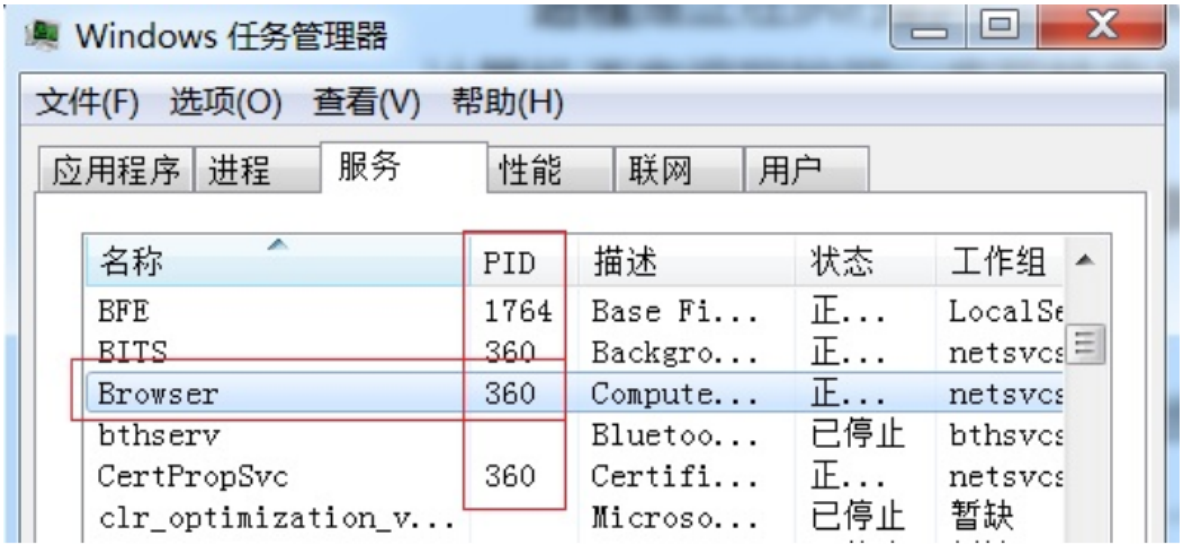
1、了解一下进程与程序的关系

进程是正在执行的一个程序或命令，每个进程都是一个运行的实体，并占用一定的系统资源。**程序**是人使用计算机语言编写的可以实现特定目标或解决特定问题的代码集合。

简单来说，程序是人使用计算机语言编写的，可以实现一定功能，并且可以执行的代码集合。进程是正在执行中的程序。

举例：谷歌浏览器是一个程序，当我们打开谷歌浏览器，就会在系统中看到一个浏览器的进程，当程序被执行时，程序的代码都会被加载入内存，操作系统给这个进程分配一个 ID，称为 **PID**（进程 ID）。我们打开多个谷歌浏览器，就有多个浏览器子进程，但是这些进程使用的程序，都是chrome

> PID = Process ID = 进程编号



2、Linux下的进程管理工作

进程查看，通过查看，判断健康状态

进程终止

进程优先级控制

三、Linux下进程管理命令

1、任务背景

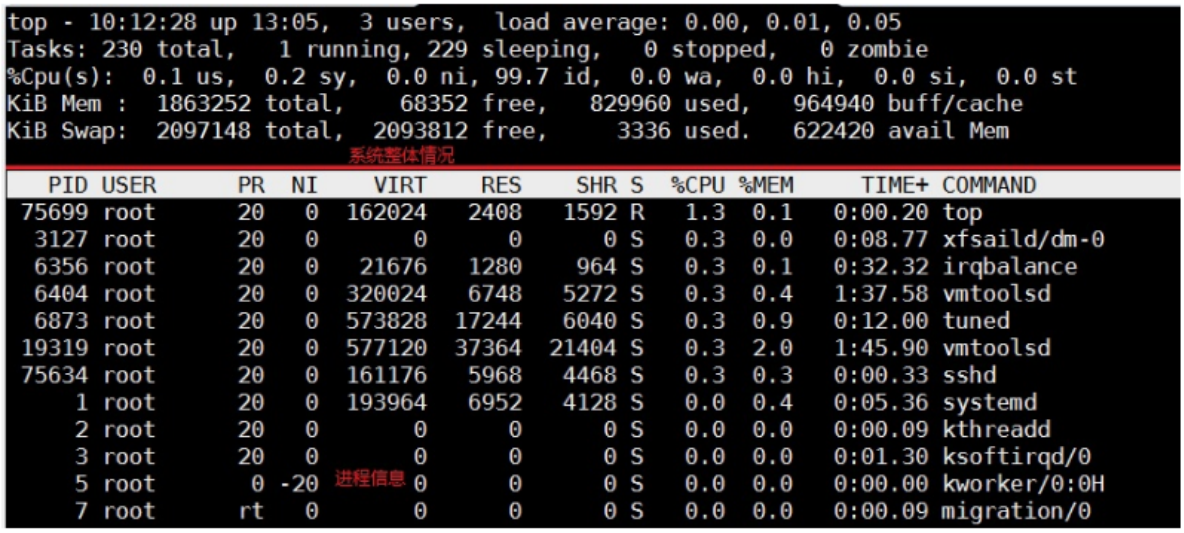
工作场景：

小黑入职到一家公司，接到的第一项任务，就是监控生产服务器的性能，提到服务器性能，我们首先想到的就是CPU，内存和磁盘。

2、使用top命令动态监测CPU信息

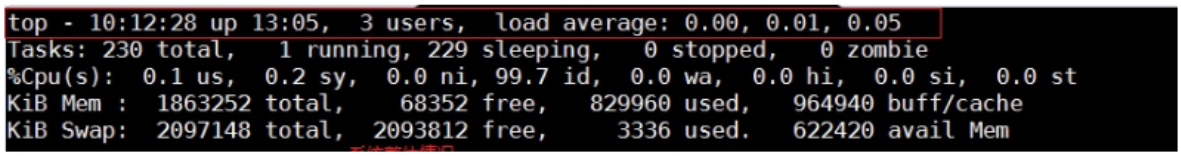
基本语法：

```
# top
```



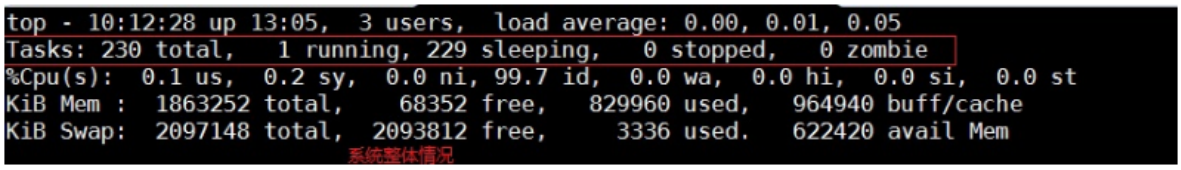
3、系统的整体情况

☆ 第一行



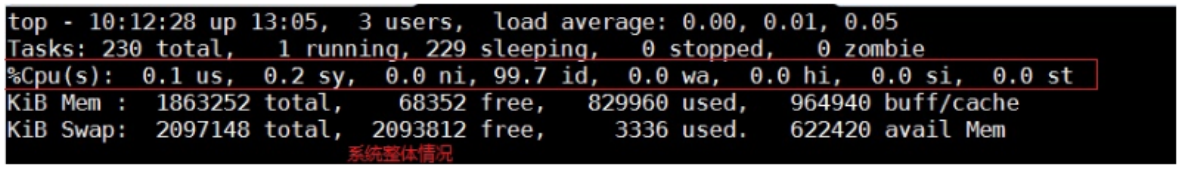
内 容	说 明
10:12:28	系统当前时间
up 13:05	系统的运行时间.本机已经运行 13 小时 05 分钟
3 users	当前登录了三个用户
load average: 0.00,0.01, 0.05	系统在之前 1 分钟、5 分钟、15 分钟的平均负载。如果 CPU 是单核的，则这个数值超过 1 就是高负载：如果 CPU 是四核的，则这个数值超过 4 就是高负载

☆ 第二行



Tasks: 230 total	系统中的进程总数
1 running	正在运行的进程数
229 sleeping	睡眠的进程数
0 stopped	正在停止的进程数
0 zombie	僵尸进程数。如果不是 0，则需要手工检查僵尸进程

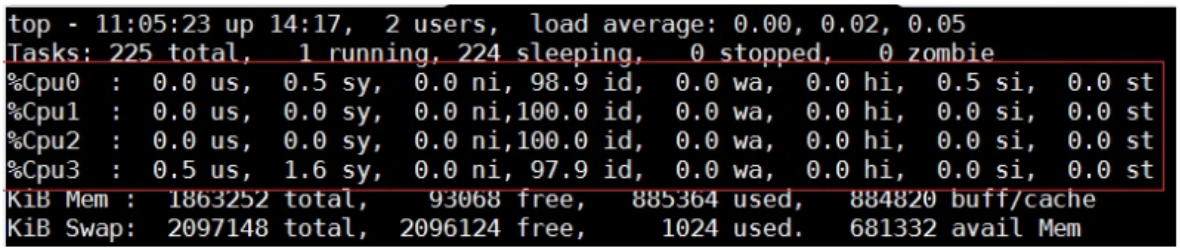
☆ 第三行



内 容	说 明
Cpu(s): 0.1 %us	用户模式占用的 CPU 百分比
0.1%sy	系统模式占用的 CPU 百分比
0.0%ni	改变过优先级的用户进程占用的 CPU 百分比
99.7%id	idle缩写，空闲 CPU 占用的 CPU 百分比
0.1%wa	等待输入/输出的进程占用的 CPU 百分比
0.0%hi	硬中断请求服务占用的 CPU 百分比
0.1%si	软中断请求服务占用的 CPU 百分比
0.0%st	st（steal time）意为虚拟时间百分比，就是当有虚拟机时，虚拟 CPU 等待实际 CPU 的时间百分比

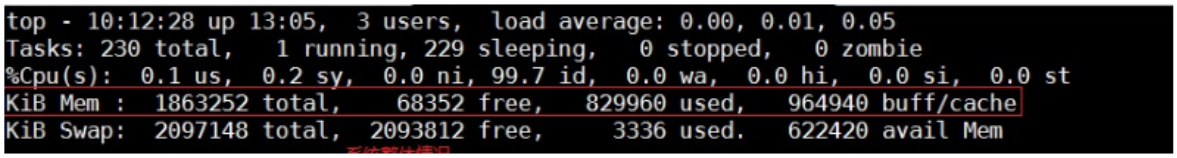
问题：如果我的机器有4核CPU，我想查看每一核心分别的负载情况怎能办？

答：交换快捷键 “1”



> CPU负载测试 => cat /dev/urandom |md5sum

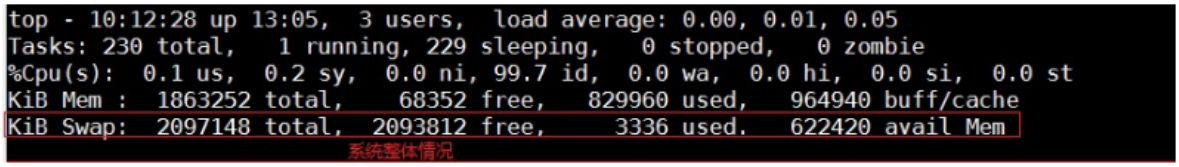
☆ 第四行



内 容	说 明
Mem: 1863252 total	物理内存的总量，单位为KB
829960 used	已经使用的物理内存数量
68352 free	空闲的物理内存数量。我们使用的是虚拟机，共分配了 628MB内存，所以只有53MB的空闲内存
96490 buff/cache	作为缓冲的内存数量

> 扩展：真正剩余内存 = free + buff/cache，真正使用内存 = used - buff/cache

☆ 第五行



内 容	说 明
Swap: 2097148 total	交换分区（虚拟内存）的总大小
3336 used	已经使用的交换分区的大小
2093812 free	空闲交换分区的大小
622420 avail Mem	可用内存

在Linux操作系统分区时，最少需要3个分区：

① /boot分区： 系统分区

② swap交换分区： 一般情况下为内存的1~2倍，但是尽量不要超过2G

③ /分区： 根分区，所有文件都存放于此

> swap分区：就是当计算机的内存不足时，系统会自动从硬盘中划出一块区域充当内存使用。

我们通过 top 命令的整体信息部分，就可以判断服务器的健康状态。如果 1 分钟、5 分钟、15 分钟的平均负载高于CPU核数，说明系统压力较大。如果物理内存的空闲内存过小，则也证明系统压力较大。

问题：根据以上信息，目前我们的系统压力如何?

答：看CPU负载及内存的使用情况

问题：如果我们发现CPU负载过大，接下来怎么办?

答：如果1分钟、5分钟以及15分钟全部超过CPU的总核心数（必须引起警觉），这个时候就要查看底部的进程信息了。

> 经验之谈：如果一个总核数=8核心的CPU，理论上平均负载达到16，也还可以坚持很长一段时间。

4、系统的进程信息

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
75699	root	20	0	162024	2408	1592	R	1.3	0.1	0:00.20	top
3127	root	20	0	0	0	0	S	0.3	0.0	0:08.77	xfsaild/dm-0
6356	root	20	0	21676	1280	964	S	0.3	0.1	0:32.32	irqbalance
6404	root	20	0	320024	6748	5272	S	0.3	0.4	1:37.58	vmtoolsd
6873	root	20	0	573828	17244	6040	S	0.3	0.9	0:12.00	tuned
19319	root	20	0	577120	37364	21404	S	0.3	2.0	1:45.90	vmtoolsd
75634	root	20	0	161176	5968	4468	S	0.3	0.3	0:00.33	sshd
1	root	20	0	193964	6952	4128	S	0.0	0.4	0:05.36	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.09	kthreadd
3	root	20	0	0	0	0	S	0.0	0.0	0:01.30	ksoftirqd/0
5	root	0	-20	进程信息	0	0	S	0.0	0.0	0:00.00	kworker/0:0H
7	root	rt	0	0	0	0	S	0.0	0.0	0:00.09	migration/0

PID	进程的 ID。
USER	该进程所属的用户。
PR	优先级，数值越小优先级越高。
NI	NICE优先级，数值越小优先级越高，取值范围-20到19，默认都是0
VIRT	该进程使用的虚拟内存的大小，单位为 KB。
RES	该进程使用的物理内存的大小，单位为 KB。
SHR	共享内存大小，单位为 KB。计算一个进程实际使用的内存 = 常驻内存（RES）- 共享内存（SHR）
S	进程状态。其中S 表示睡眠，R 表示运行
%CPU	该进程占用 CPU 的百分比。
%MEM	该进程占用内存的百分比。
TIME+	该进程共占用的 CPU 时间。
COMMAND	进程名

问题：如果我们发现CPU负载过大，接下来怎么办?

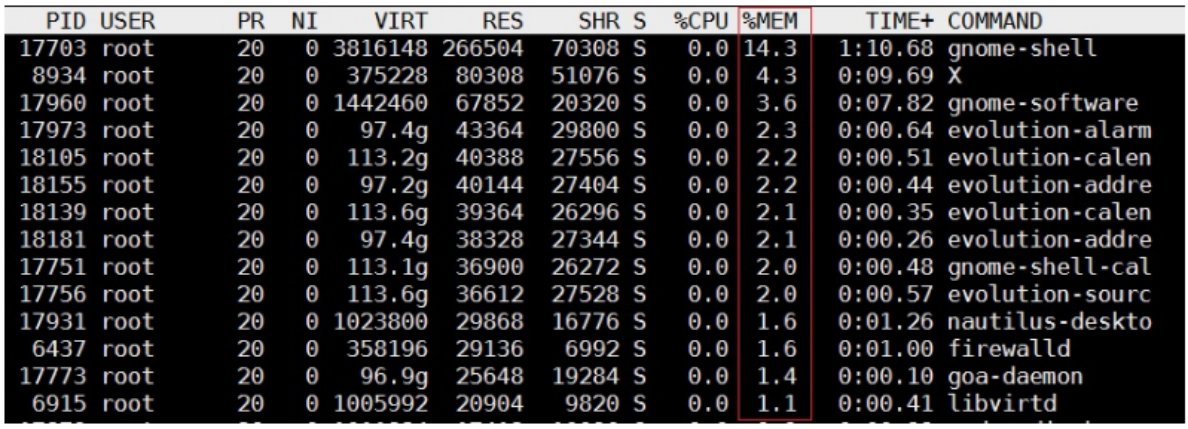
答：查看占用CPU最多的进程

问题：如何查看占用CPU最多的进程?

答：交互操作快捷键P，P（大写）：，表示将结果按照CPU 使用率从高到低进行降序排列

问题：如果我们发现内存可用量很小，接下来怎么办?

答：查看占用内存最多的进程，使用交互快捷键M（大写）：表示将结果按照内存（MEM）从高到低进行降序排列



```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
17703 root      20   0 3816148 266504  70308 S   0.0 14.3   1:10.68 gnome-shell
 8934 root      20   0  375228  80308  51076 S   0.0  4.3   0:09.69 X
17960 root      20   0 1442460  67852  20320 S   0.0  3.6   0:07.82 gnome-software
17973 root      20   0   97.4g  43364  29800 S   0.0  2.3   0:00.64 evolution-alarm
18105 root      20   0  113.2g  40388  27556 S   0.0  2.2   0:00.51 evolution-calen
18155 root      20   0   97.2g  40144  27404 S   0.0  2.2   0:00.44 evolution-addre
18139 root      20   0  113.6g  39364  26296 S   0.0  2.1   0:00.35 evolution-calen
18181 root      20   0   97.4g  38328  27344 S   0.0  2.1   0:00.26 evolution-addre
17751 root      20   0  113.1g  36900  26272 S   0.0  2.0   0:00.48 gnome-shell-cal
17756 root      20   0  113.6g  36612  27528 S   0.0  2.0   0:00.57 evolution-sourc
17931 root      20   0 1023800  29868  16776 S   0.0  1.6   0:01.26 nautilus-deskto
 6437 root      20   0  358196  29136   6992 S   0.0  1.6   0:01.00 firewalld
17773 root      20   0   96.9g  25648  19284 S   0.0  1.4   0:00.10 goa-daemon
 6915 root      20   0 1005992  20904   9820 S   0.0  1.1   0:00.41 libvirtd
```

问题：当我们查看完系统状态，需要做什么？

答：退出，使用q，按键盘上的q，就会回到#提示符的状态。

5、free查看内存使用情况

基本语法：

```
# free [选项]  1GB = 1024MB  1MB = 1024KB
选项说明：
-m ：以MB的形式显示内存大小
```

案例：显示计算机的内存使用情况

```
# free -m
```

和Centos6相比，buffer和cached被合成一组，加入了一个available。

关于此available，即系统可用内存，用户不需要去计算buffer/cache，即可以看到还有多少内存可用，更加简单直观

```
[root@localhost ~]# free -m
              total        used        free      shared  buff/cache   available
Mem:           1819         774         152          77         892         746
Swap:          2047           0        2047
[root@localhost ~]#
```

6、df查看磁盘剩余空间

基本语法：

```
# df [选项]
-h ：以较高的可读性显示磁盘剩余空间大小
```

> df = disk free = 磁盘 剩余

这几列依次是：

Filesystem	磁盘名称
Size	总大小
Used	被使用的大小
Avail	剩余大小
Use%	使用百分比
Mounted on	挂载路径（相当于Windows 的磁盘符）

7、ps查看系统进程信息

top： 动态查看系统进程的信息（每隔3s切换一次）

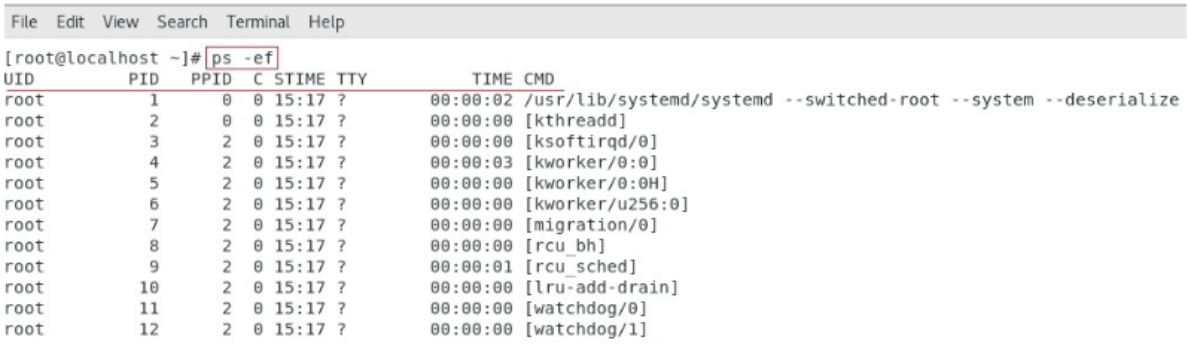
ps ： 静态查看系统进程的信息（只能查询运行ps命令瞬间，系统的进程信息）

基本语法：

```
# ps [选项]
选项说明：
-e ： 等价于“-A”，表示列出全部（all）的进程
-f ： 表示full，显示全部的列（显示全字段）
```

案例：显示当前系统中所有进程的信息

```
# ps -ef
```



UID	该进程执行的用户ID
PID	进程ID
PPID	该进程的父级进程ID，如果找不到，则该进程就被称之为僵尸进程（Parent Process ID）
C	Cpu的占用率，其形式是百分数
STIME	进程的启动时间
TTY	终端设备，发起该进程的设备识别符号，如果显示“?”则表示该进程并不是由终端设备发起
TIME	进程实际使用CPU的时间
CMD	该进程的名称或者对应的路径

> 经验之谈：我们在实际工作中使用ps命令其实主要用于查询某个进程的PID或PPID

工作场景

小黑用学到的命令，发现某个进程占用CPU很高，希望进一步查看这个简称的信息。

ps -ef 会列出全部进程，但是我们发现进程非常多，我们很难找到自己想要看的进程。这里需要使用过滤命令grep，来过滤掉我们不需要的信息。

基本语法：

```
用法：ps -ef |grep 想要看到的进程名
示例代码：
# ps -ef |grep crond
含义：查看crond进程的详细情况
注意：查询结果中，如果只有一条则表示没查到对应的进程（这1 条表示刚才ps 指令的自身）。只有查到的结果多余1 条，才表示有对应的进程。
```

案例：查询crond的进程信息

```
# ps -ef |grep crond
root       7102      1  0 Mar23 ?        00:00:04 /usr/sbin/crond -n
root      24752  12881  0 16:34 pts/2    00:00:00 grep --color=auto crond
```

问题：以上信息只有第一行是crond的进程，第二行，实际是管道命令发起时，grep所启动的进程，如何去掉?

```
# ps -ef |grep crond |grep -v "grep"
root       7102      1  0 Mar23 ?        00:00:04 /usr/sbin/crond -n
```

> grep -v 需要去除的相关信息： 去除包含指定关键词的那一行

扩展：ps aux命令

```
# ps aux
```

> # man ps
>
> 1 UNIX options, which may be grouped and must be preceded by a dash. ps -ef
> 2 BSD options, which may be grouped and must not be used with a dash. ps aux

```
[root@bogon ~]# ps aux
USER        PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root          1  0.0  0.3 193824  6328 ?        Ss   Feb15   0:06 /usr/lib/system
root          2  0.0  0.0      0     0 ?        S    Feb15   0:00 [kthreadd]
root          3  0.0  0.0      0     0 ?        S    Feb15   0:01 [ksoftirqd/0]
root          5  0.0  0.0      0     0 ?        S<   Feb15   0:00 [kworker/0:0H]
root          7  0.0  0.0      0     0 ?        S    Feb15   0:00 [migration/0]
root          8  0.0  0.0      0     0 ?        S    Feb15   0:00 [rcu_bh]
```

USER：该 process 属于哪个使用者账号
==PID ： 该 process 的ID==
==%CPU：该 process 使用掉的 CPU 资源百分比==
==%MEM：该 process 所占用的物理内存百分比==
VSZ ： 该 process 使用掉的虚拟内存量 (Kbytes)
RSS ： 该 process 占用的固定的内存量 (Kbytes)
TTY ： 该 process 是在那个终端机上面运作，若与终端机无关，则显示 ?，另外， tty1-tty6 是本机上面的登入者程序，若为 pts/0 等等的，则表示为由网络连接进主机的程序。
==STAT：该程序目前的状态，主要的状态有==
R ： 该程序目前正在运作，或者是可被运作
S ： 该程序目前正在睡眠当中 (可说是 idle 状态)，但可被某些讯号 (signal) 唤醒。
T ： 该程序目前正在侦测或者是停止了
==Z ： 该程序应该已经终止，但是其父程序却无法正常的终止他，造成 zombie (疆尸) 程序的状态==
START：该 process 被触发启动的时间
TIME ： 该 process 实际使用 CPU 运作的时间
COMMAND：该程序的实际指令

8、netstat/ss查询网络访问信息

基本语法：

```
# netstat [选项] |grep 进程名称
选项说明：
-t：表示只列出tcp 协议的连接（tcp协议与udp协议）
-n：表示将地址从字母组合转化成ip 地址，将协议转化成端口号来显示  10.1.1.10:80
-l：表示过滤出"state（状态）"列中其值为LISTEN（监听）的连接
-p：表示显示发起连接的进程pid 和进程名称
```

案例：查询Web Server（httpd）服务的端口信息

```
# netstat -tnlp |grep httpd
```

基本语法：

```
# ss -naltp |grep 进程名称
```

案例：查询sshd服务的端口信息

```
# ss -naltp |grep sshd
```

> netstat与ss区别？① netstat信息比较简洁，ss更加丰富 ② ss执行效率比netstat略高一些

9、kill/killall杀死进程

☆ 根据pid杀掉进程

```
命令：kill
语法：kill [信号] PID
作用：kill 命令会向操作系统内核发送一个信号（多是终止信号）和目标进程的 PID，然后系统内核根据收到的信号类型，对指定进程进行相应的操作

经验：kill经常结合ps命令一起使用
```

> kill命令用于杀死某个进程，这其实只是其一个功能。kill命令的实质是向进程发送信号

信号种类：

信号编号	含义
9	杀死进程，即强制结束进程。
15	正常结束进程，是 kill 命令的默认信号。

案例：使用kill命令杀死crond进程

```
# ps -ef |grep crond
7102
# kill 7102
```

> 备注：在互联网中，经常看到kill -9 进程PID，强制杀死某个进程，kill -9 pid

☆ 根据进程名称杀掉进程

基本语法：

```
# killall [信号编号] 进程名称
```

案例：使用killall命令杀死crond进程

```
# killall crond
```

案例：使用killall命令杀死httpd进程

```
# killall httpd
```