内核通过一个接口将进程信息输出给用户：就是**/proc 这个目录**

Ls一下会发现很多以数字命名的目录

这就是进程id命名的也就是进程的PID

对应进程的信息

Cd /proc/1 进程1的信息

一般都是只读的

相关文件的简单说明

例如

1)cmdline保存启动这个命令的命令行

cat看一下

Init3

2)stat

Statm

**status**

状态相关 的很多信息

前两个都是一串数值

不过status更加详细，各种信息几乎都有

**在调优的中用的到**

但是使用专用 的工具 也可以变为容易查看的信息：

例如

Pstree

进程更为详细的命令

Ps 命令process show

直接命令比较少

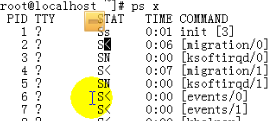
一般加选项

Bsd 直接字符

Sysv风格 加个横线

-aux与 aux不同

常用选项



a 所有的进程 所有与终端相关的进程

x 所有与终端无关的进程 (TTY显示为?)

有5个字段：PID TTY STAT TIME COMMAND

**STAT:**

S 可中断的进程

s 进程领导者session leader

< 高优先级进程

N 低优先级进程

R 运行或者就绪的状态

l 多线程的进程

+ 这属于一个前台进程组(由某个进程领导的，例如bash就是一个前台进程组)

D不可中断的睡眠

T 停止态的进程

Z 僵尸态进程

ax 就是所有的进程 中断相关和无关的进程

u 以用户为中心显示

**aux 每个进程 详细信息 最常用的组合**

字段：

USER

%cpu %CPU cpu utilization of the process in "##.#" format. Currently, it is the CPU time used divided by the time the process has

been running (cputime/realtime ratio), expressed as a percentage. It will not add up to 100% unless you are lucky.

(alias pcpu).

%mem %MEM ratio of the process's resident set size to the physical memory on the machine, expressed as a percentage. (alias pmem).

rss RSS (此进程占据的物理内存空间，不包括swap分区的) resident set size, the non-swapped physical memory that a task has used (in kiloBytes). (alias rssize, rsz).

VSZ 虚拟内存占用大小

rtprio RTPRIO realtime priority.

start STARTED time the command started. If the process was started less than 24 hours ago, the output format is "HH:MM:SS", else it is

" mmm dd" (where mmm is a three-letter month name).

ps –ef -e类似ax f详细信息 F额外信息差不多两个

ps –ejH 缩进显示进程状态类似树的状态

ps axjf 更加直观的子进程树的样子

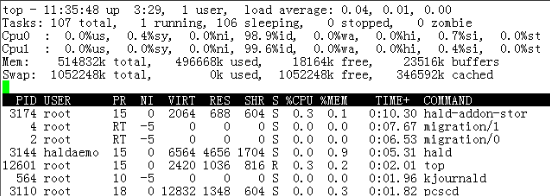
ps -eo -o指定要显示的字段 例子：ps –eo pid,%cpu,comm

pgrep 进程过滤

-U USERNAME 指定与哪个用户相关的进程

-G GROUPNAME 指定与哪个组相关的进程

**pidof 查看特定进程的进程号 多进程的服务器上用的多，例如使用了多少个httpd进程**



**top**  动态显示 5s刷新

**load**  cpu队列中等待执行的进程个数，队列长度的平均值，三个值分别为1min 5min 15min的平均值。当平均长度超过3的时候差不度就是性能瓶颈。

Tasks 任务总数

CPU(s): us用户空间占用率 sy内核空间占用率 ni nice值调整的占用率 id空闲比例 wa wait状态的cpu占用率 hi si 回来介绍st(stolen 虚拟机占用的cpu) 按1 切换cpu分开或者汇总显示

**PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND**

PR 优先级

NI nice值

VIRT 虚拟内存集

RES 真正占用的内存空间

SHR 共享内存空间

S 状态

CPU 这个刷新时间周期内的CPU

TIME+占用的CPU累积时长

%CPU -- CPU usage

The task's share of the elapsed CPU time since the last screen update, expressed as a percentage of total CPU time. In a true SMP

environment, if 'Irix mode' is Off, top will operate in 'Solaris mode' where a task's cpu usage will be divided by the total number

of CPUs. You toggle 'Irix/Solaris' modes with the 'I' interactive command.

TIME -- CPU Time

Total CPU time the task has used since it started. When 'Cumulative mode' is On, each process is listed with the cpu time that it

and its dead children has used. You toggle 'Cumulative mode' with 'S', which is a command-line option and an interactive command.

See the 'S' interactive command for additional information regarding this mode.

TIME+ -- CPU Time, hundredths

The same as 'TIME', but reflecting more granularity through hundredths of a second.

%MEM -- Memory usage (RES)

A task's currently used share of available physical memory.

VIRT -- Virtual Image (kb)

The total amount of virtual memory used by the task. It includes all code, data and shared libraries plus pages that have been

swapped out. (Note: you can define the STATSIZE=1 environment variable and the VIRT will be calculated from the /proc/#/state

VmSize field.)

VIRT = SWAP + RES.

RES -- Resident size (kb)

The non-swapped physical memory a task has used.

SHR -- Shared Mem size (kb)

The amount of shared memory used by a task. It simply reflects memory that could be potentially shared with other processes.

**排序：**

**SORTING of task window：**

M %MEM Yes

N PID Yes

P %CPU Yes

T TIME+ Yes

**是否显示最上面的信息**

**SUMMARY Area Commands：**

l :Toggle\_Load\_Average/Uptime -- On/Off

This is also the line containing the program name (possibly an alias) when operating in full-screen mode or the ?current' win-

dow name when operating in alternate-display mode.

m :Toggle\_Memory/Swap\_Usage -- On/Off

This command affects two summary area lines.

t :Toggle\_Task/Cpu\_States -- On/Off

This command affects from 2 to many summary area lines, depending on the state of the '1' toggle and whether or not top is run-

ning under true SMP.

1 :Toggle\_Single/Separate\_Cpu\_States -- On/Off

This command affects how the 't' command's Cpu States portion is shown. Although this toggle exists primarily to serve mas-

sively-parallel SMP machines, it is not restricted to solely SMP environments.

top -d delay延迟刷新时间

-b bat模式

-n # 批次的个数 -b -n 3 分3批显示

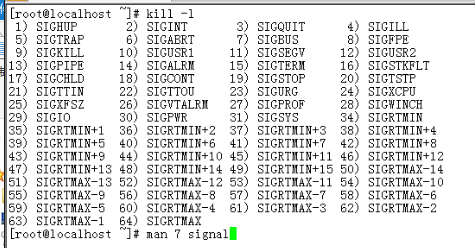
**进程间通信：IPC 的方式：**

**管道，共享内存，信号，消息队列**

这里主要介绍信号：

信号：简短信息，

kill –l 列出所有可用的信号 或者使用man 7 signal也可以显示



标准信号

**1: SIGHUP** 例如运行httpd读取配置文件中的定义，当进程运行时改变进程时需要重新载入服务才可以重新生效，发送sighup信号重新读取配置文件

**2: SIGINT** int interrput终止 ctrl c实际就是发送了一个sigint信号

**9: SIGKILL** kill 杀死

**15: SIGTERM** term 终止 杀死和终止：终止可以被保存（默认的）

17 19 23 SIGSTOP 停止态

19 18 25 sigcont 停止态转就绪态

发送信号的例子：

kill -9 PID

kill -SIGKILL PID

kill -KILL PID

killall -9 comm 同名的全部杀死

pkill 类似于pgrep 杀死某一用户的进程

通过调整**NICE值**改变进程的优先级：

nice值范围，越小优先级越高，但并不是很影响动态优先级和静态优先级

-20 - 19

查看nice值：

ps –ef 没有

top命令可以

ps –eo pid

定义nice值：进程没有启动，直接以这个niice值启动

nice -n # command

启动的进程改进nice值

renice # PID

**作业控制：**

job:

前台作业：

在某终端启动，并占据着终端直到进程运行结束

后台作业：

作业启动后释放终端，并在后台执行完成

logout 终端一关闭后台进程也关闭了 这样后台进程会关闭，解决方法：

**nohup command &**

终端无关

前台作业-->后台作业：

**Ctrl+Z** (**将作业挂起，但处于停止状态**)<--> **fg**(回到前台，多个时看下边)

**jobs:**

**查看后台中的作业列表，**有作业号+-

fg

默认调回带+号的那个，然后是减号

fg %JOBS\_NUM (%可省略) 例子：fg 1

bg命令：调入后台并**执行**

bg %JOBS\_NUM

终止后台作业：

kill作业 kill %作业号 %不能省略