**第十章 Centos7-系统进程管理**

**本节所讲内容：**

**10.1 进程概述和ps查看进程工具**

**10.2 uptime查看系统负载-top动态管理进程**

**10.3 前后台进程切换- nice进程优先级-实战screen后台执行命令**

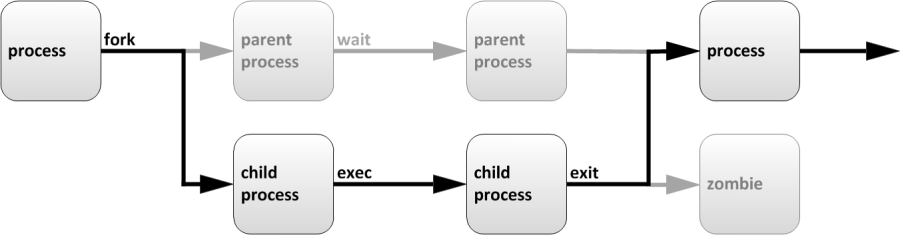
# 10.1 进程概述和ps管理进程

## 10.1.1 什么是进程？

**进程是已启动的可执行程序的运行实例，进程有以下组成部分：  
 • 已分配内存的地址空间；  
 • 安全属性，包括所有权凭据和特权；  
 • 程序代码的一个或多个执行线程；  
 • 进程状态**

**程序： 二进制文件，静态 /bin/date,/usr/sbin/sshd   
进程： 是程序运行的过程， 动态，有生命周期及运行状态。**

**下图所示的是进程的生命周期：**

****

**描述如下：**

**父进程复制自己的地址空间（fork**  **[fɔ:k] 分叉）创建一个新的（子）进程结构。每个新进程分配一个唯一的进程 ID （PID），满足跟踪安全性之需。PID 和 父进程 ID （PPID）是子进程环境的元素，任何进程都可以创建子进程，所有进程都是第一个系统进程的后代。**

**centos5或6PID为1的进程是： init**

**centos7 PID为1的进程是： systemd**

**僵尸进程：一个进程使用fork创建子进程，如果子进程退出，而父进程并没有调用wait或waitpid获取子进程的状态信息，那么子进程的进程描述符仍然保存在系统中。这种进程称之为僵尸进程。**

**用自己的话表达:父进程退出了， 子进程没有退出， 那么这些子进程就没有父进程来管理了，就变成僵尸进程。**

## 10.1. 2 进程的属性

**进程ID（PID)：是唯一的数值，用来区分进程**

**父进程的ID（PPID)**

**启动进程的用户ID（UID）和所归属的组（GID）**

**进程状态：状态分为运行R、休眠S、僵尸Z**

**进程执行的优先级**

**进程所连接的终端名**

**进程资源占用：比如占用资源大小（内存、CPU占用量）**

## 10.1.3 使用ps查看进程工具

**1、ps查看进程工具**

**例1：常用的参数：**

**a: 显示跟当前终端关联的所有进程**

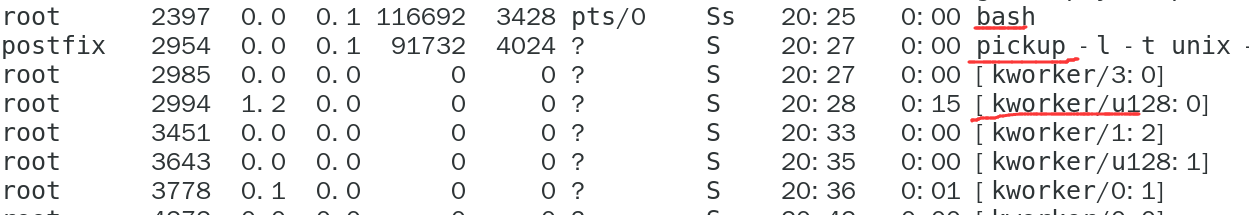
**u: 基于用户的格式显示（U: 显示某用户ID所有的进程）**

**x: 显示所有进程，不以终端机来区分**

**例2：常用的选项组合是 ps -aux**

**[root@xuegod63 ~]# ps -axu | more**





**注： 最后一列[xxxx] 使用方括号括起来的进程是内核态的进程。 没有括起来的是用户态进程。**

**上面的参数输出每列含意：**

**USER: 启动这些进程的用户**

**PID: 进程的ID**

**%CPU 进程占用的CPU百分比；   
%MEM 占用内存的百分比；**

**VSZ：进程占用的虚拟内存大小（单位：KB）   
RSS：进程占用的物理内存大小（单位：KB）**

**STAT：该程序目前的状态，Linux进程有5种基本状态：**

**R ：该程序目前正在运作，或者是可被运作；**

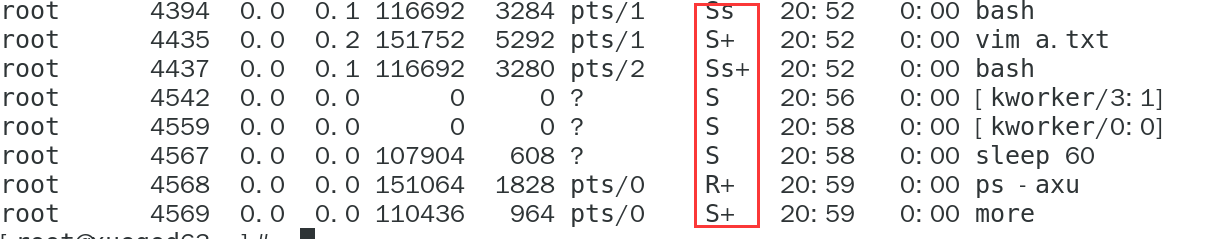
**S ：该程序目前正在睡眠当中 (可说是 idle 状态啦！)，但可被某些讯号(signal) 唤醒。**

**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**

**D 不可中断状态.**

**5个基本状态后，还可以加一些字母，比如：Ss、R+，如下图：**



**它们含意如下:：**

**<: 表示进程运行在高优先级上**

**N: 表示进程运行在低优先级上**

**L: 表示进程有页面锁定在内存中**

**s: 表示进程是控制进程**

**l: 表示进程是多线程的**

**+: 表示当前进程运行在前台**

**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**

**例1： 查看进程状态**

**[root@xuegod63 ~]# vim a.txt**

**在另一个终端执行：**

**[root@xuegod63 ~]# ps -aux | grep a.txt #查看状态 S表示睡眠状态， + 表示前台**

**root 4435 0.0 0.2 151752 5292 pts/1 S+ 20:52 0:00 vim a.txt**

**root 4661 0.0 0.0 112676 996 pts/0 S+ 21:05 0:00 grep --color=auto a.txt**

**在vim a.txt 这个终端上 按下： ctrl+z**

**[1]+ 已停止 vim a.txt**

**在另一个终端执行：**

**[root@xuegod63 ~]# ps -aux | grep a.txt #查看状态 T表示停止状态**

**root 4435 0.0 0.2 151752 5292 pts/1 T 20:52 0:00 vim a.txt**

**root 4675 0.0 0.0 112676 996 pts/0 S+ 21:05 0:00 grep --color=auto a.txt**

**注：**

**ctrl-c 是发送 SIGINT 信号，终止一个进程**

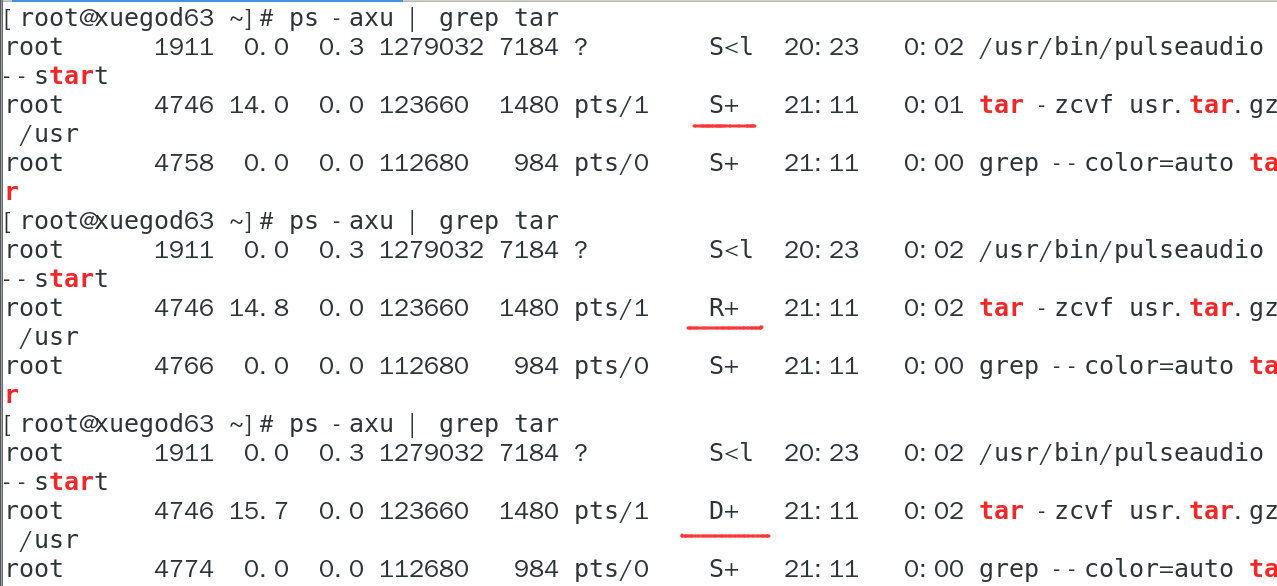
**ctrl-z 是发送 SIGSTOP信号，挂起一个进程。将作业放置到后台(暂停)**

**ctrl-d 不是发送信号，而是表示一个特殊的二进制值，表示 EOF。代表输入完成或者注销**

**例2： D 不可中断状态**

**[root@xuegod63 ~]# tar -zcvf usr-tar.gz /usr/**

**#然后在另一个终端不断查看状态，由S+，R+变为D+**

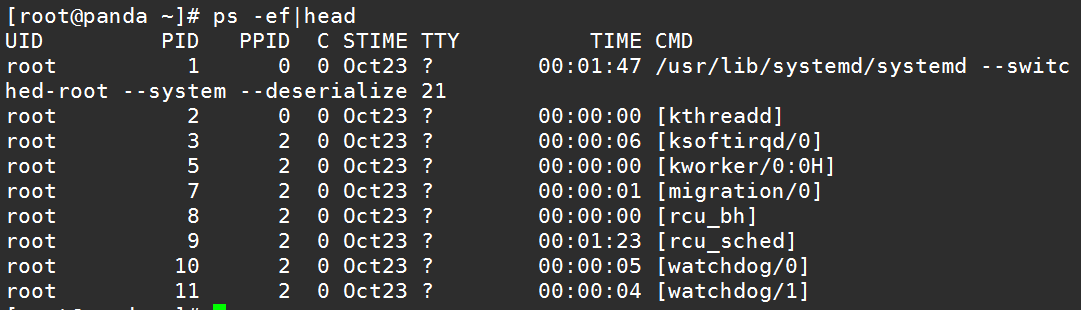


**2、ps常用的参数： ps -ef**

**-e 显示所有进程**

**-f 显示完整格式输出**

**我们常用的组合： ps -ef**

****

**包含的信息如下**

**UID: 启动这些进程的用户**

**PID: 进程的ID**

**PPID: 父进程的进程号**

**C: 进程生命周期中的CPU利用率**

**STIME: 进程启动时的系统时间**

**TTY: 表明进程在哪个终端设备上运行。如果显示 ?表示与终端无关，这种进程一般是内核态进程。另外， tty1-tty6 是本机上面的登入者程序，若为 pts/0 等，则表示运行在虚拟终端上的进程。**

**TIME: 运行进程一共累计占用的CPU时间**

**CMD: 启动的程序名称**

**例1：测试CPU使用时间。**

**dd if=/dev/zero of=/a.txt count=10 bs=100M**

**[root@localhost ~]# ps -axu | grep dd**

**注：**

**ps aux 是用BSD的格式来显示进程。**

**ps -ef 是用标准的格式显示进程**

# 10.2 uptime查看系统负载-top动态管理进程

**10.2.1 uptime查看CPU负载工具**

**[root@localhost ~]# uptime**

**13:22:30 up 20days,  2 users,  load average: 0.06, 0.60, 0.48**

**弹出消息含意如下：**

|  |  |
| --- | --- |
| **13:22:30** | **当前时间** |
| **up 20days** | **系统运行时间 ，说明此服务器连续运行20天了** |
| **2 user** | **当前登录用户数** |
| **load average: 0.06, 0.60, 0.48** | **系统负载，即任务队列的平均长度。 三个数值分别为  1分钟、5分钟、15分钟前到现在的平均值。** |

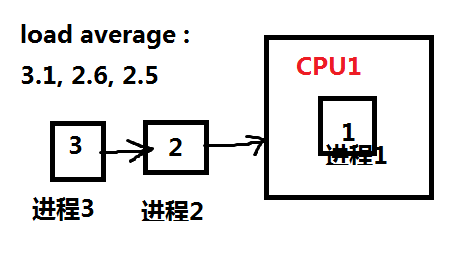
**任务队列的平均长度是什么？**

**大厅排除买票：**

****

**这时队列是4:**

**cpu队列数为3时，如图：**

****

**任务队列的平均长度 懂：1 不懂：2**

**互动：例1：找出当前系统中，CPU负载过高的服务器？**

**服务器1： load average: 0.15, 0.08, 0.01 1核**

**服务器2： load average: 4.15, 6.08, 6.01 1核**

**服务器3： load average: 10.15, 10.08, 10.01 4核**

**答案：服务器2**

**如果服务器的CPU为1核心，则load average中的数字 >=3 负载过高，如果服务器的CPU为4核心，则load average中的数字 >=12 负载过高。**

**经验：单核心，1分钟的系统平均负载不要超过3，就可以，这是个经验值。**

**如下图： 1人只能买1张票，排第四的人可能会急。 所以我们认为超过3就升级CPU**

****

**10.2.2 top命令**

**[root@xuegod63 ~]# top #top弹出的每行信息含意如下：**

**第一行内容和uptime弹出的信息一样**

**进程和CPU的信息( 第二、三行)**

****

**当有多个CPU时，这些内容可能会超过两行。内容如下：**

|  |  |
| --- | --- |
| **Tasks: 481 total** | **进程总数** |
| **1 running** | **正在运行的进程数** |
| **480 sleeping** | **睡眠的进程数** |
| **0 stopped** | **停止的进程数** |
| **0 zombie** | **僵尸进程数** |
| **Cpu(s): 0.0% us** | **系统用户进程使用CPU百分比。** |
| **0.0% sy** | **内核中的进程占用CPU百分比** |
| **0.0% ni** | **用户进程空间内改变过优先级的进程占用CPU百分比** |
| **98.7% id** | **空闲CPU百分比** |
| **0.0% wa** | **cpu等待I/0完成的时间总量。**  **测试：**  **终端1：执行：top**  **终端2：dd if=/dev/zero of=/a.txt count=10 bs=100M**  **终端3：dd if=/dev/zero of=/a.txt count=10 bs=100M**  **如下：** |
| **0.0% hi（了解）** 硬中断消耗时间 | **硬中断，占的CPU百分比。1. 硬中断是由硬件产生的，比如，像磁盘，网卡，键盘，时钟等。每个设备或设备集都有它自己的IRQ（中断请求）。基于IRQ，CPU可以将相应的请求分发到对应的硬件驱动上（注：硬件驱动通常是内核中的一个子程序，而不是一个独立的进程）。# hi -> Hardware IRQ: The amount of time the CPU has been servicing hardware interrupts.** |
| **0.0% si（了解）**  **软中断消耗时间** | **软中断，占的CPU百分比。1. 通常，软中断是一些对I/O的请求。这些请求会调用内核中可以调度I/O发生的程序。对于某些设备，I/O请求需要被立即处理，而磁盘I/O请求通常可以排队并且可以稍后处理。根据I/O模型的不同，进程或许会被挂起直到I/O完成，此时内核调度器就会选择另一个进程去运行。I/O可以在进程之间产生并且调度过程通常和磁盘I/O的方式是相同。# si -> Software Interrupts.: The amount of time the CPU has been servicingsoftware interrupts.** |
| **0.0 st （steal 偷）** | **st：虚拟机偷取物理的时间。比如：物理机已经运行了KVM，XEN虚拟机。KVM虚拟机占用物理机的cpu时间** |

**内存信息(第四五行)**

****

**内容如下：**

|  |  |
| --- | --- |
| **Mem: 2033552k total** | **物理内存总量** |

|  |  |
| --- | --- |
| **340392k used** | **使用的物理内存总量** |
| **1376636k free** | **空闲内存总量** |
| **316524k buff/cache** | **用作内核缓存的内存量。**  **和free -k 一个意思** |
| **Swap: 2017948k total** | **交换区总量** |
| **0k used** | **使用的交换区总量** |
| **192772k free** | **空闲交换区总量** |
| **1518148 avail Mem** | **总的可利用内存是多少** |

**注：如果swap分区，被使用，那么你的内存不够用了。**

**第7行进程信息**

****

|  |  |
| --- | --- |
| **列名** | **含义** |
| **PID** | **进程id** |
| **USER** | **进程所有者的用户名** |
| **PR** | **优先级（由内核动态调整），用户不能** |
| **NI** | **进程优先级。 nice值。负值表示高优先级，正值表示低优先级，用户可以自己调整** |
| **VIRT（virtual memory usage）** | **虚拟内存，是进程正在使用的所有内存（ps中标为VSZ）**  **VIRT：virtual memory usage 虚拟内存**  **1、进程“需要的”虚拟内存大小，包括进程使用的库、代码、数据等**  **2、假如进程申请100m的内存，但实际只使用了10m，那么它会增长100m，而不是实际的使用量** |
| **RES（resident memory usage）** | **是进程所使用的物理内存。实际实用内存（ps中标为RSS）**  **RES：resident memory usage 常驻内存**  **1、进程当前使用的内存大小，但不包括swap out**  **2、包含其他进程的共享**  **3、如果申请100m的内存，实际使用10m，它只增长10m，与VIRT相反**  **4、关于库占用内存的情况，它只统计加载的库文件所占内存大小** |
| **SHR** | **共享内存大小，单位kb**  **SHR：shared memory 共享内存**  **1、除M了自身进程的共享内存，也包括其他进程的共享内存**  **2、虽然进程只使用了几个共享库的函数，但它包含了整个共享库的大小**  **3、计算某个进程所占的物理内存大小公式：RES – SHR**  **4、swap out后，它将会降下来** |
| **S** | **进程状态。             D=不可中断的睡眠状态             R=运行中或可运行             S=睡眠中             T=已跟踪/已停止             Z=僵停** |
| **%CPU** | **上次更新到现在的CPU时间占用百分比** |
| **%MEM** | **进程使用的物理内存百分比** |
| **TIME+** | **进程使用的CPU时间总计，单位1/100秒** |
| **COMMAND** | **命令名/命令行** |

**top快捷键：**

**默认3s刷新一次，按s修改刷新时间**

**按空格 ：立即刷新。**

**q退出**

**P：按CPU排序**

**M：按内存排序**

**T按时间排序**

**p: 进程IP，查看某个进程状态**

**数字键1：显示每个内核的CPU使用率**

**u/U：指定显示的用户**

**h:帮助**

**例1：运行top，依次演示一下top的快捷键，让大家看一下效果**

**例2：使用TOP动态只查看某个或某些进程的信息**

**找到进程ID**

**[root@localhost ~]# ps -axu | grep vim**

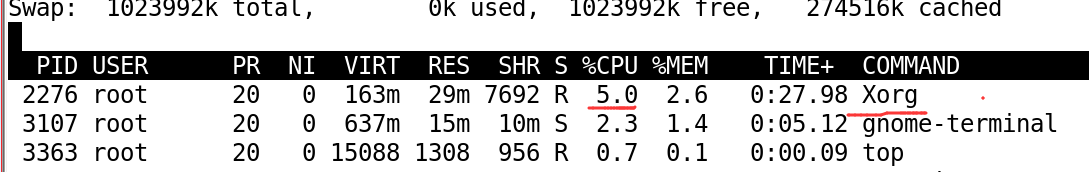
**Warning: bad syntax, perhaps a bogus '-'? See /usr/share/doc/procps-3.2.8/FAQ**

**root 9667 0.0 0.2 143620 3344 pts/1 S<+ 19:15 0:00 vim a.txt**

**[root@localhost ~]# top -p 9667**

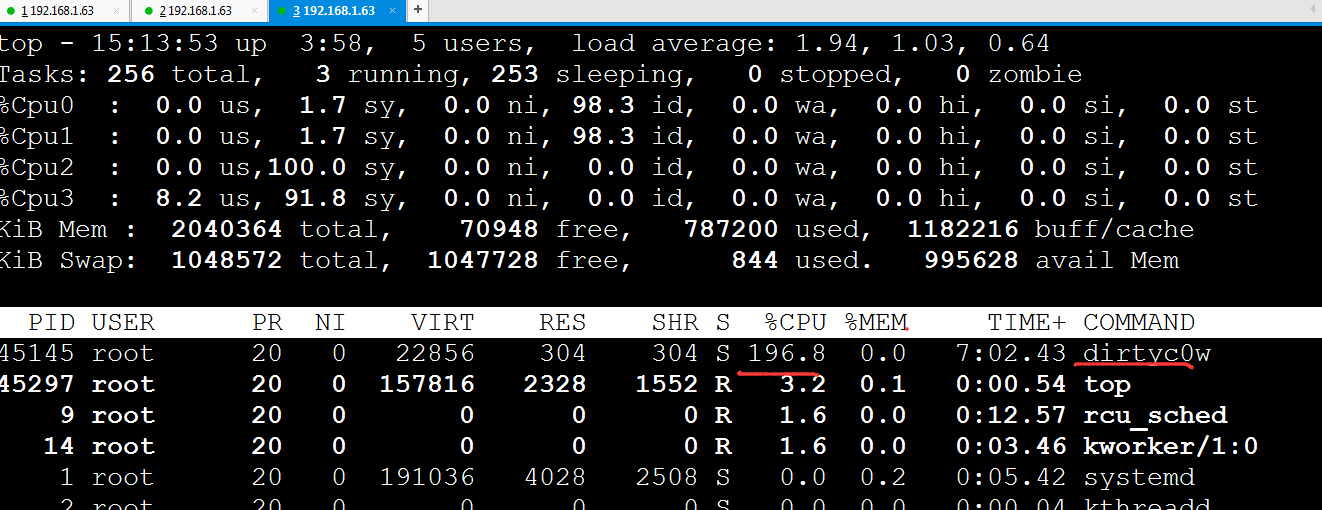
**10.2.3 实战1：找出系统中使用CPU最多的进程**

**运行top ， 找出使用CPU最多的进程 ，按大写的P，可以按CPU使用率来排序显示**

****

**互动：在linux系统中一个进程，最多可以使用100%cpu对吗？**

**如下图，可以看到dirtycow（脏牛漏洞，用于提权） 进程使用196.8%**

****

**这是你第一次看见： 1**

**如果你的4核心的cpu，你可以运行400%**

**10.2.4 lsof命令**

**lsof命令用于查看你进程打开的文件，打开文件的进程，进程打开的端口(TCP、UDP)**

**-i<条件>：列出符合条件的进程。（4、6、协议、:端口、 @ip ）**

**-p<进程号>：列出指定进程号所打开的文件；**

**例：**

**[root@xuegod63 ~]# vim a.txt**

**[root@xuegod63 ~]# ps -axu | grep a.txt**

**root 43641 0.8 0.2 151744 5280 pts/3 S+ 18:19 0:00 vim a.txt**

**root 43652 0.0 0.0 112676 996 pts/1 S+ 18:19 0:00 grep --color=auto a.txt**

**[root@xuegod63 ~]# lsof -p 43641 #一般用于查看木马进程，在读哪些文件**

**[root@xuegod63 ~]# lsof -i :22 #用于查看端口，或查看黑客开启的后门端口是哪个进程在使用**

## 10.2.4 pstree工具使用

**pstree：（display a tree of processes）以树状图显示进程，只显示进程的名字，且相同进程合并显示。**

**格式：pstree 或 pstree -p**

**以树状图显示进程，还显示进程PID。**

**[root@xuegod63 ~]# pstree -p**

# 10.3 前后台进程切换- nice进程优先级-实战screen后台执行命令

## 10.3.1 Linux后台进程与前台进程的区别

**前台进程:是在终端中运行的命令，那么该终端就为进程的控制终端，一旦这个终端关闭，这个进程也随着消失**

**后台进程: 也叫守护进程（Daemon），是运行在后台的一种特殊进程，不受终端控制，它不需要与终端交互；Linux的大多数服务器就是用守护进程实现的。比如，Web服务器httpd等。**

## 10.3.2 进程的前台与后台运行

**跟系统任务相关的几个命令（了解）：**

|  |  |
| --- | --- |
| **&** | **用在一个命令的最后，可以把这个命令放到后台执行.** |
| **ctrl + z** | **将一个正在前台执行的命令放到后台，并且暂停.** |
| **jobs** | **查看当前有多少在后台运行的进程.它是一个作业控制命令** |
| **fg（foreground process）** | **将后台中的命令调至前台继续运行, 如果后台中有多个命令，可以用 fg %jobnumber将选中的命令调出，%jobnumber是通过jobs命令查到的后台正在执行的命令的序号(不是pid)** |
| **bg(background process)** | **将一个在后台暂停的命令，变成继续执行; 如果后台中有多个命令，可以用bg %jobnumber将选中的命令调出，%jobnumber是通过jobs命令查到的后台正在执行的命令的序号(不是pid)** |

**实战恢复被挂起的进程（了解）**

**例： vim a.txt 按下： ctrl+z**

**[root@xuegod63 ~]# vim a.txt #打开后，然后执行 ctrl+z**

**[1]+ 已停止 vim a.txt**

**[root@xuegod63 ~]# ps -axu | grep vim**

**root 43710 0.8 0.2 151744 5304 pts/3 T 18:26 0:00 vim a.txt**

**root 43720 0.0 0.0 112676 984 pts/3 S+ 18:26 0:00 grep --color=auto vim**

**[root@xuegod63 ~]# jobs #查看当前有多少在后台运行的进程**

**[1]+ 已停止 vim a.txt**

**[root@xuegod63 ~]# fg 1 #将后台挂起的进程恢复到前台运行**

## 10.3.3 kill关闭进程

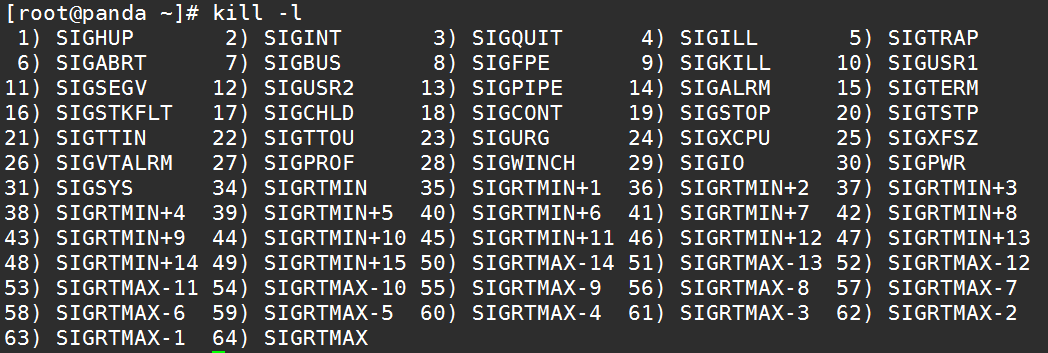
**关闭进程3个命令：kill killall pkill**

**kill关闭进程：kill 进程号 关闭单个进程**

**killall和pkill 命令用于杀死指定名字的进程**

**通过信号的方式来控制进程的**

**kill -l =====> 列出所有支持的信号（了解） 用最多的是： 9 信号**

****

**信号编号 信号名**

**1） SIGHUP 重新加载配置**

**2） SIGINT 键盘中断 crtl+c**

**3) SIGQUIT 退出**

**9） SIGKILL 强制终止**

**15） SIGTERM 终止（正常结束），缺省信号**

**18） SIGCONT 继续**

**19） SIGSTOP 停止**

**20） SIGTSTP 暂停 crtl+z**

**例1： kill和killall终止进程**

**[root@xuegod63 ~]# kill -9 2342**

**[root@xuegod63 ~]# killall sshd**

**[root@xuegod63 ~]# pkill sshd**

## 10.3.4 进程的优先级管理

**优先级取值范围为（-20,19），越小优先级越高, 默认优先级是0**

**命令1：nice 指定程序的运行优先级**

**格式：nice n command**

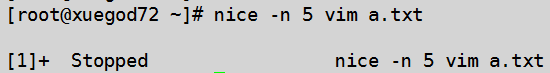
**命令2：renice 改变程序的运行优先级**

**格式：renice -n pid**

**例1：指定运行vim的优先级为5**

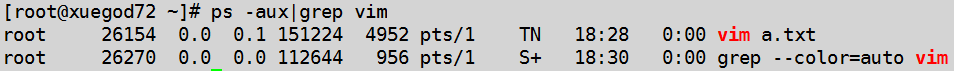
**[root@xuegod63 ~]# nice -n 5 vim a.txt**

**输入内容，然后ctrl+z 挂起**

****

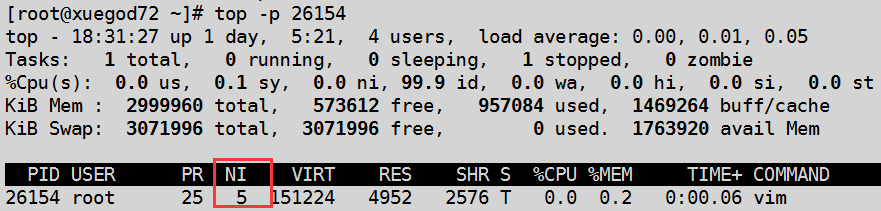
**通过ps查看这个文件的PID号**

**[root@xuegod63 ~]# ps -aux|grep vim**

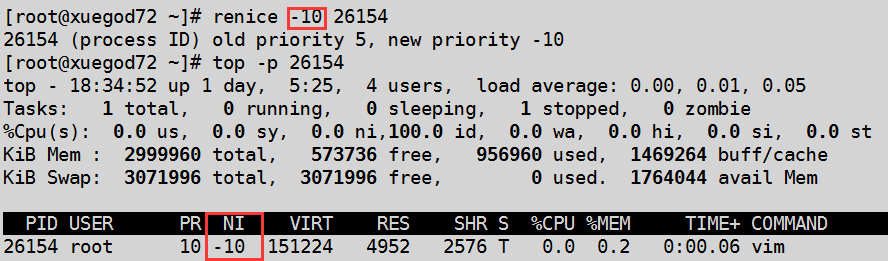
****

**通过top命令查看优先级**

**[root@xuegod63 ~]# top -p 26154**

****

**改变正在运行的进程的优先级**

****

## 10.3.5 实战：使用screen后台实时执行命令备份命令

**实战场景：公司晚上需要备份1T数据，我在xshell上直接执行备份脚本back.sh可以吗？ 或直接运行back.sh & 放到后台运行可以吗？ 当关了xshell后，back.sh & 还在后台执行吗？**

**答：xshell长时间连接，如果本地网络偶尔断开或xshell不小心关闭，都会让后台运行的备份命令停止运行的。正确做法使用： srceen**

**10.3.6 screen概述和安装**

**Screen中有会话的概念,，用户可以在一个screen会话中创建多个screen窗口，在每一个screen窗口中就像操作一个真实的telnet/SSH连接窗口那样。**

**安装screen软件包**

**# rpm -ivh /mnt/Packages/screen-4.1.0-0.23.20120314git3c2946.el7\_2.x86\_64.rpm**

**或者**

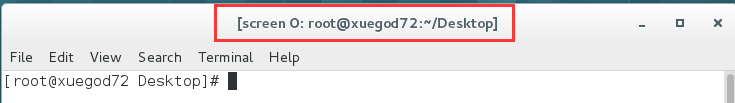
**[root@xuegod63 ~]# yum -y install screen**

**10.3.7 screen使用方法**

**直接在命令行键入screen命令回车，如下图**

**[root@xuegod63 ~]# screen**

**Screen将创建一个执行shell的全屏窗口。你可以执行任意shell程序，就像在ssh窗口中那样**

****

**例如，我们在做某个大型的操作但是突然之间断开：**

**实战：使用screen后台实时执行命令备份命令**

**[root@xuegod63 ~]# screen #进入**

**[root@xuegod63 ~]# vim a.txt #执行命令， 或执行你自己需要运行的备份命令**

**此时想离开一段时间，但还想让这个命令继续运行**

**[root@xuegod63 ~]# #在screen当前窗口键入快捷键Ctrl+a+d**

**[detached from 44074.pts-3.xuegod63] #分离出来独立的一个会话**

**detached [dɪˈtætʃt] 分离，独立**

**半个小时之后回来了，找到该screen会话：**

**[root@tivf06 ~]# screen -ls #查看已经建立的会话ID**

**There is a screen on:**

**44074.pts-1.tivf06 (Detached)**

**1 Socket in /tmp/screens/S-root.**

**重新连接会话：**

**[root@xuegod63 ~]# screen -r 44074**

**root@xuegod63 ~]# exit #不想使用screen 会话了，执行：exit退出。**

**附：常用screen参数**

**screen -S test -> 新建一个叫test的会话**

**screen -ls -> 列出当前所有的会话**

**screen -r test -> 回到test会话**

**总结：**

**10.1 进程概述和ps查看进程工具**

**10.2 uptime查看系统负载-top动态管理进程**

**10.3 前后台进程切换- nice进程优先级-实战screen后台执行命令**