**实验三：进程管理（一）**

返回[目录](#_目录)

**实验目的：**

1. 学习进程的基本概念，理解进程的基本属性
2. 掌握进程操作的主要命令
3. 掌握通过/proc文件系统获取进程信息
4. 理解Linux系统的进程树

**实验时间**

6学时

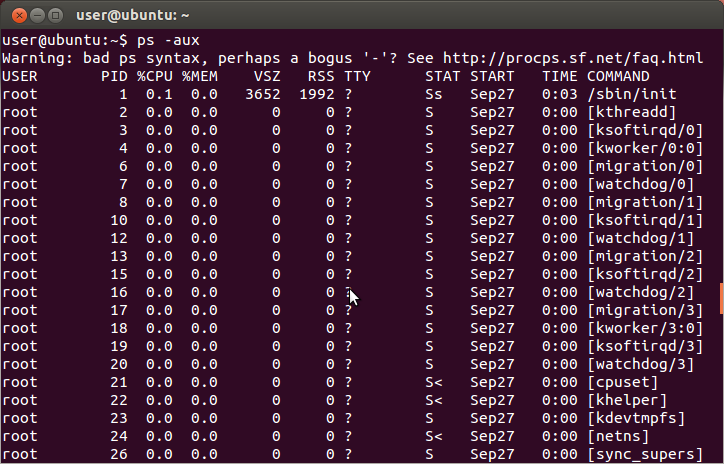
**预备知识：**

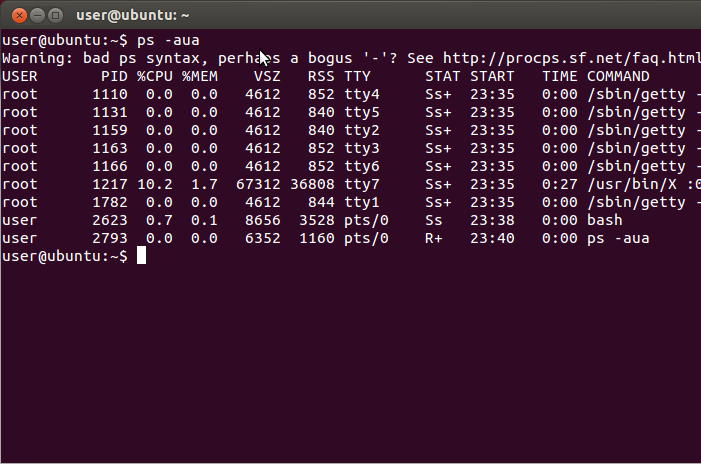
1. 有关进程的主要命令：
   1. ps：查看系统运行进程

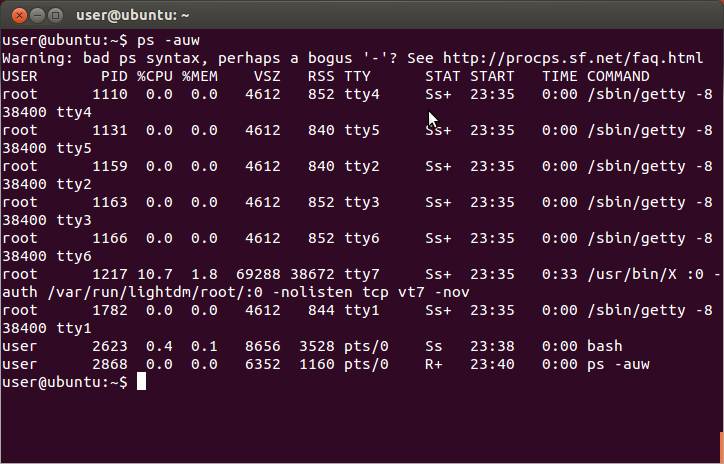
ps常用选项（最常用ps -aux）：

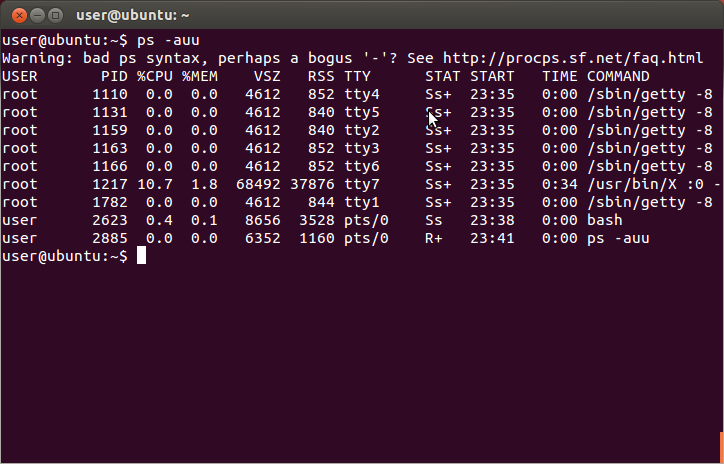
|  |  |
| --- | --- |
| 选项 | 说明 |
| x | 显示机器上的所有进程，不以终端来区分 |
| a | 显示终端上的所有进程，不以用户来区分 |
| w | 提供详细的宽范围输出 |
| u | 显示面向用户的格式 |
| f | 以进程树格式列出进程 |

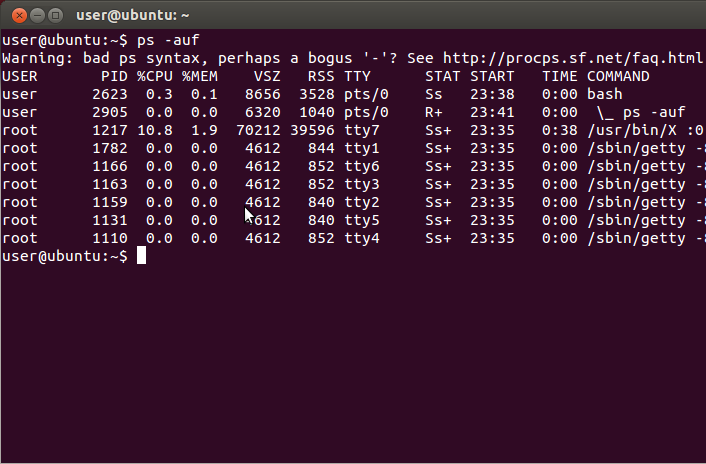
没有列全











ps运行结果的各字段含义：

|  |  |
| --- | --- |
| 字段 | 说明 |
| UID | 用户ID |
| PID | 进程ID |
| PPID | 父进程ID |
| TTY | 控制终端ID |
| PRI | 优先级编号（编号越低，分配给此进程的计算机时间越多） |
| NI(nice) | 影响动态优先级调整 |
| STAT | 当前的进程状态 |
| TIME | 使用的CPU时间 |
| COMMAND | 命令的名称 |

进程状态(STAT)可为以下状态之一：

|  |  |
| --- | --- |
| 状态代码 | 说明 |
| R（可运行） | 可运行的进程 |
| S（休眠） | 正在等待外部事件的进程（如数据到达）。 |
| D（不可中断的休眠） | 类似“S”（此时不能终止此进程） |
| T（已跟踪或已停止） | 进程已被暂停 |
| X （不用） | 进程已死 |
| Z（僵进程） | 进程已自行终止，但尚未释放 |





















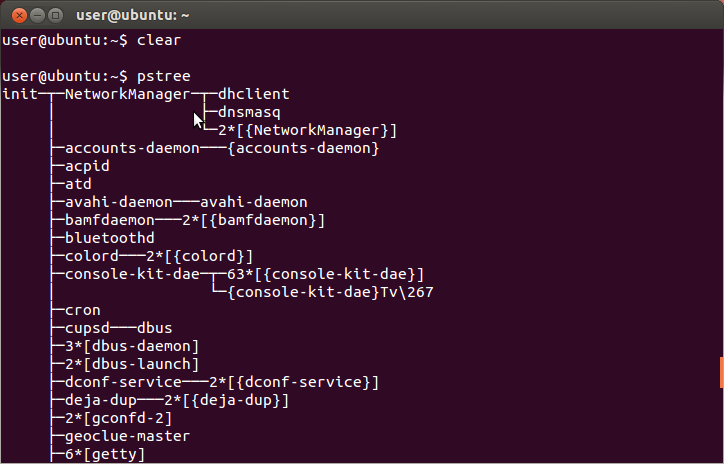
* 1. pstree：以树结构显示进程列表

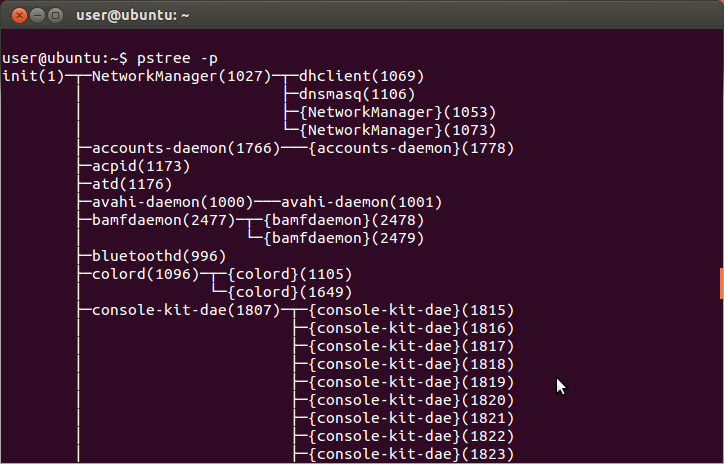
pstree常用选项：

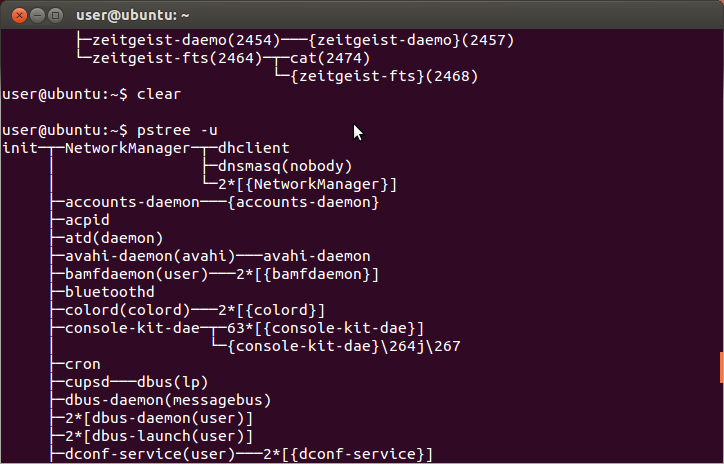
|  |  |
| --- | --- |
| 选项 | 说明 |
| -p | 显示进程PID |
| -u | 显示用户ID |

2号进程及子孙：

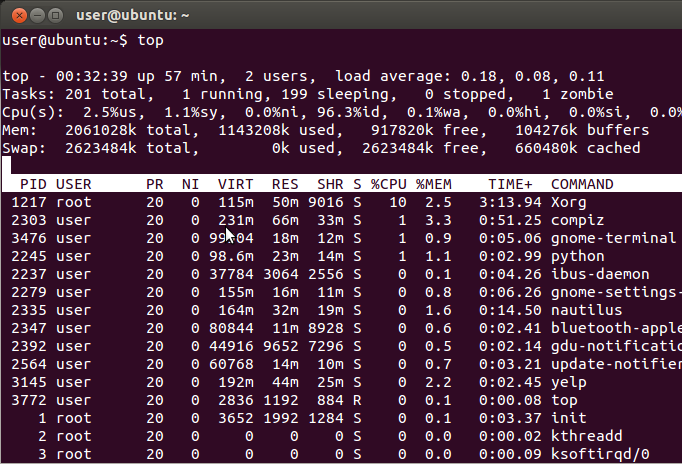
pstree 2 -p







1.3 top: 实时显示正在运行的进程



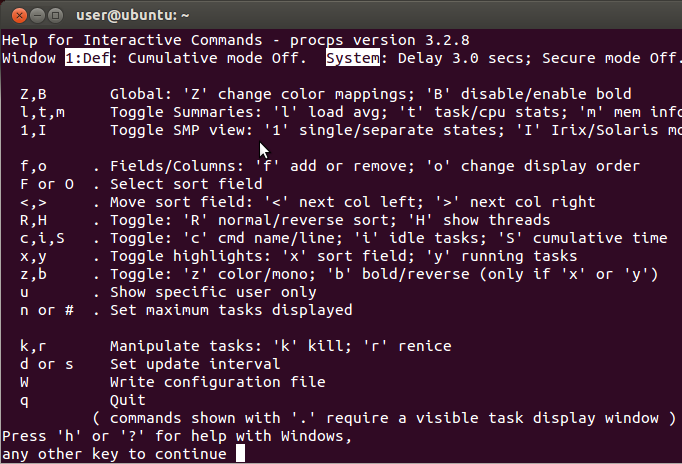
top运行结果的各字段含义：

|  |  |
| --- | --- |
| 字段 | 说明 |
| PID | 进程ID |
| USER | 用户名 |
| PR | 优先级 |
| NI | Nice值 |
| VIRT | 进程使用的虚拟内存总量（单位为KB） |
| RES | 进程持有的物理内存总量（单位为KB） |
| SHR | 共享内存大小（单位为KB） |
| S | 进程状态 |
| %CPU | CPU使用率 |
| %MEM | 内存使用率(RES) |
| TIME | CPU时间 |
| COMMMAND | 命令名称/行 |

top常用命令：

|  |  |
| --- | --- |
| 字段 | 说明 |
| ？ | 帮助 |
| h | 帮助 |
| r | 将一个新的 nice 值分配给运行中的进程 |
| k | 向某个运行中的进程发送终止信号（与 kill 或 killall相同） |
| N | 按进程 ID 排序 |
| P | 按 CPU 负载排序 |

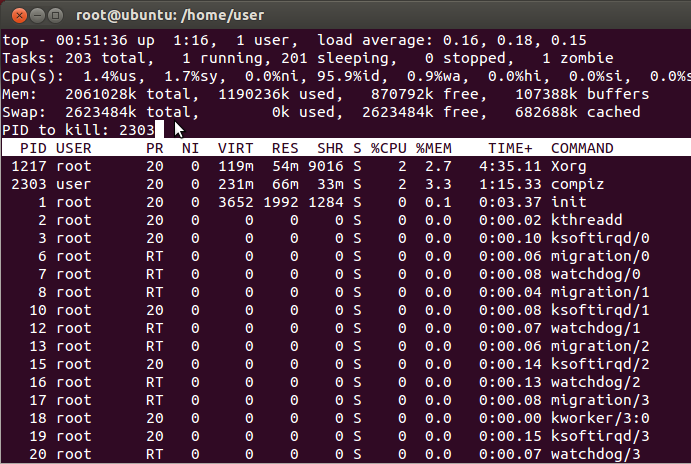
1.？/h

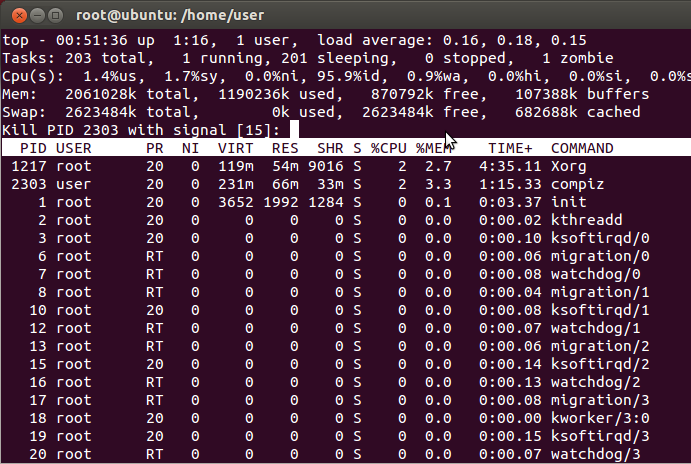


2.r

在后面~

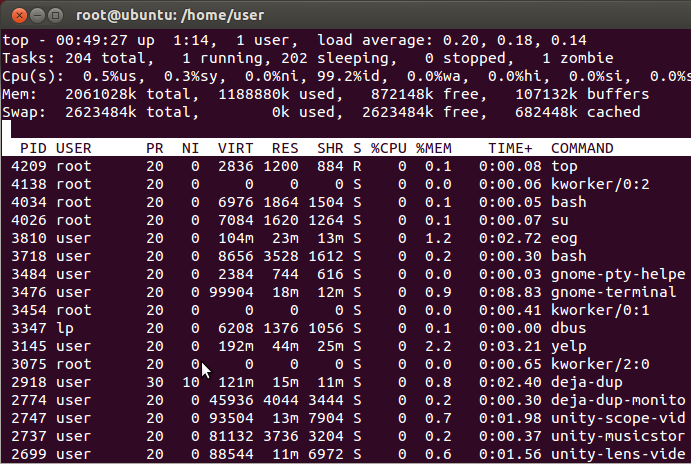
3.K



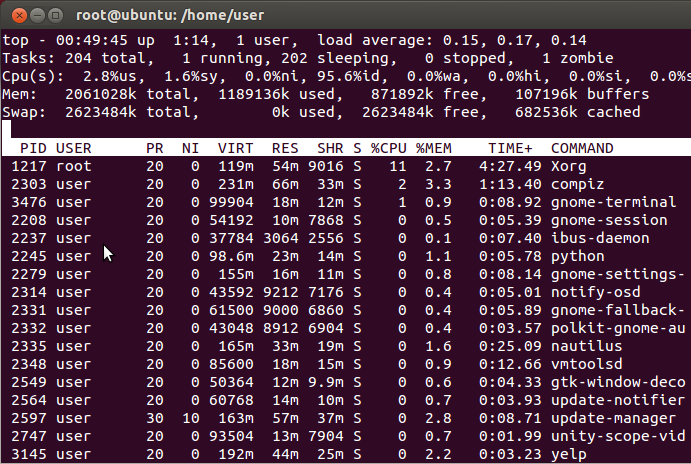


好像然后就死机了，应该是关了一个系统必要的进程。。。

4.N



5.P



1.4 nice和renice

nice:为进程指派静态优先级

E.g. $nice –n +5 sleep 99

renice:更改运行进程的nice值

E.g. $renice 5 1012

注意： 1.调度程序使用nice级别来确定为运行的进程提供服务的频率。

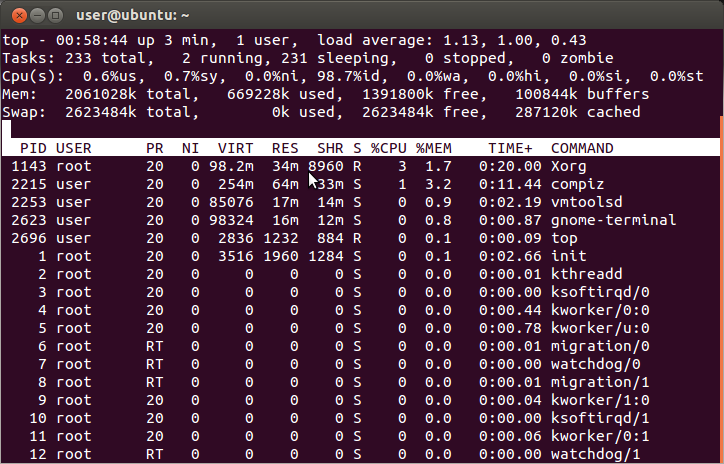
2.nice的范围：-20~20；nice值越低，进城优先级越高。

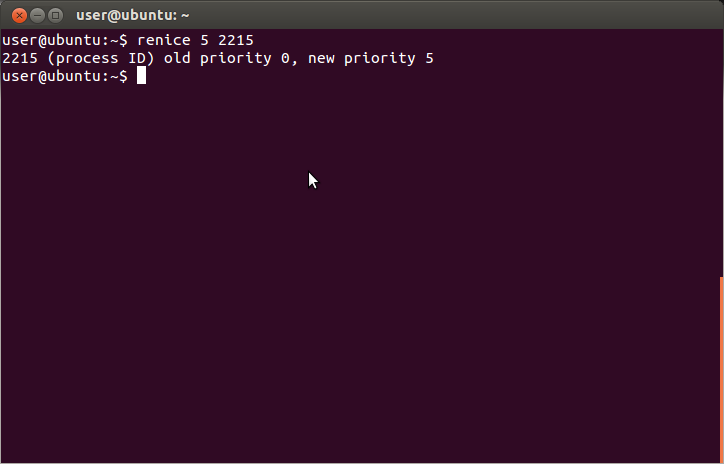
3.nice默认为10。

4.只有root用户可以负的nice值启动进程；只有root用户可以降低运行进程的nice值。

1.nice

2.renice







1.5 前台与后台进程

$command & 以后台方式启动进程

fg 将进程切换到前台

bg 将进程切换到后台运行

jobs 列出作业控制的内容

ctrl+Z 挂起进程

1.6 kill和killall 向进程发送信号

kill 用PID向进程发送信号

E.g. $kill –SIGTERM 1703

killall 用进程的命令名称向进程发送信号

常用信号：

|  |  |  |
| --- | --- | --- |
| 编号 | 名称 | 说明 |
| 1 | SIGHUP | 重装载配置文件 |
| 2 | SIGINT | 通过键盘（Ctrl+C）中断 |
| 9 | SIGKILL | 停止进程。 |
| 15 | SIGTERM | 立即结束进程（以受控的方法终止进程，因此可以进行清理） |
| 18 | SIGCONT | 继续由STOP停止的进程 |
| 19 | STOP | 停止进程 |

注意：kill –l 或man 7 signal可获取信号的完整列表。

2. 利用/proc文件系统查看进程信息

系统中运行的每个进程在/proc目录下有个目录，目录名称为进程PID。

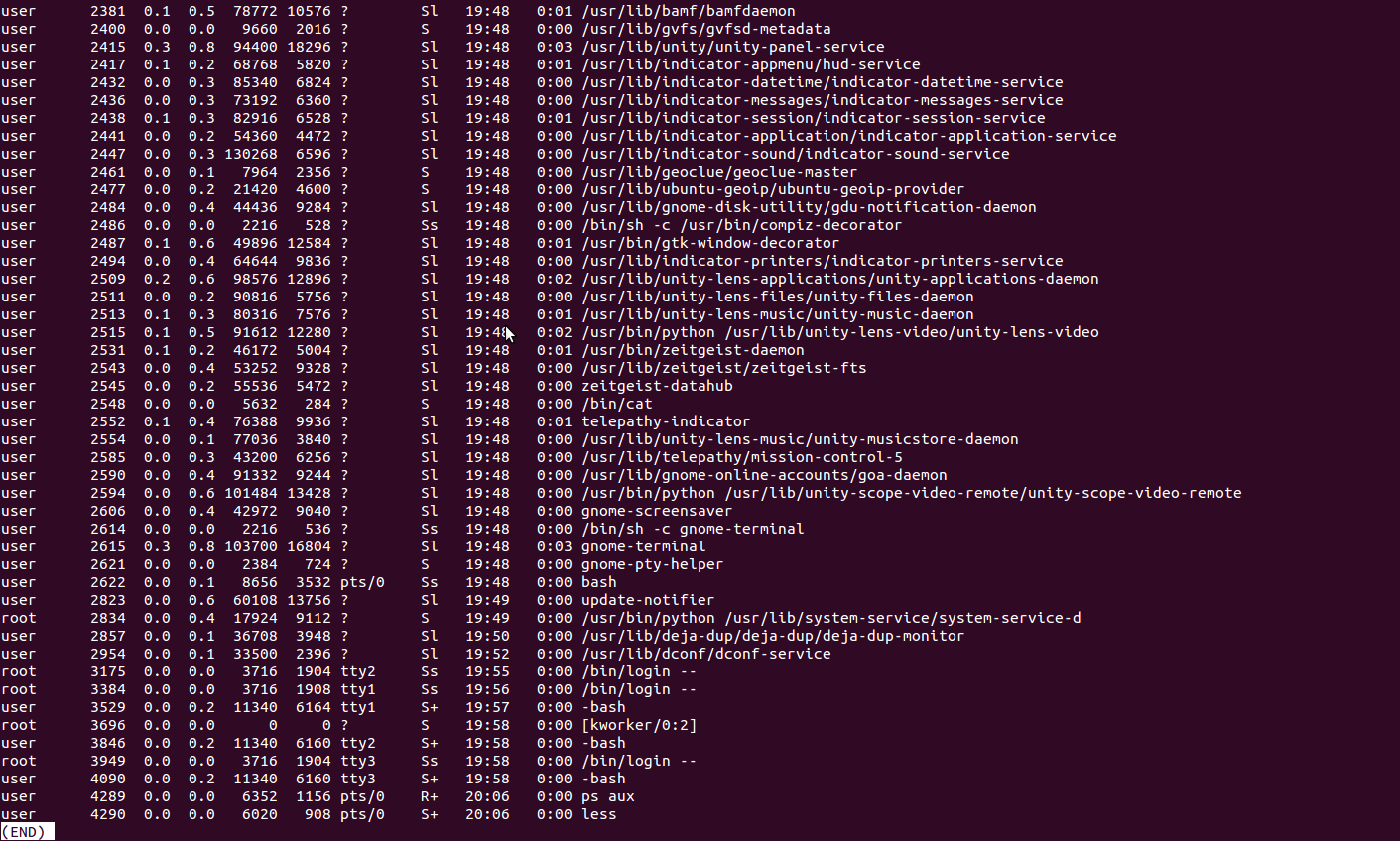
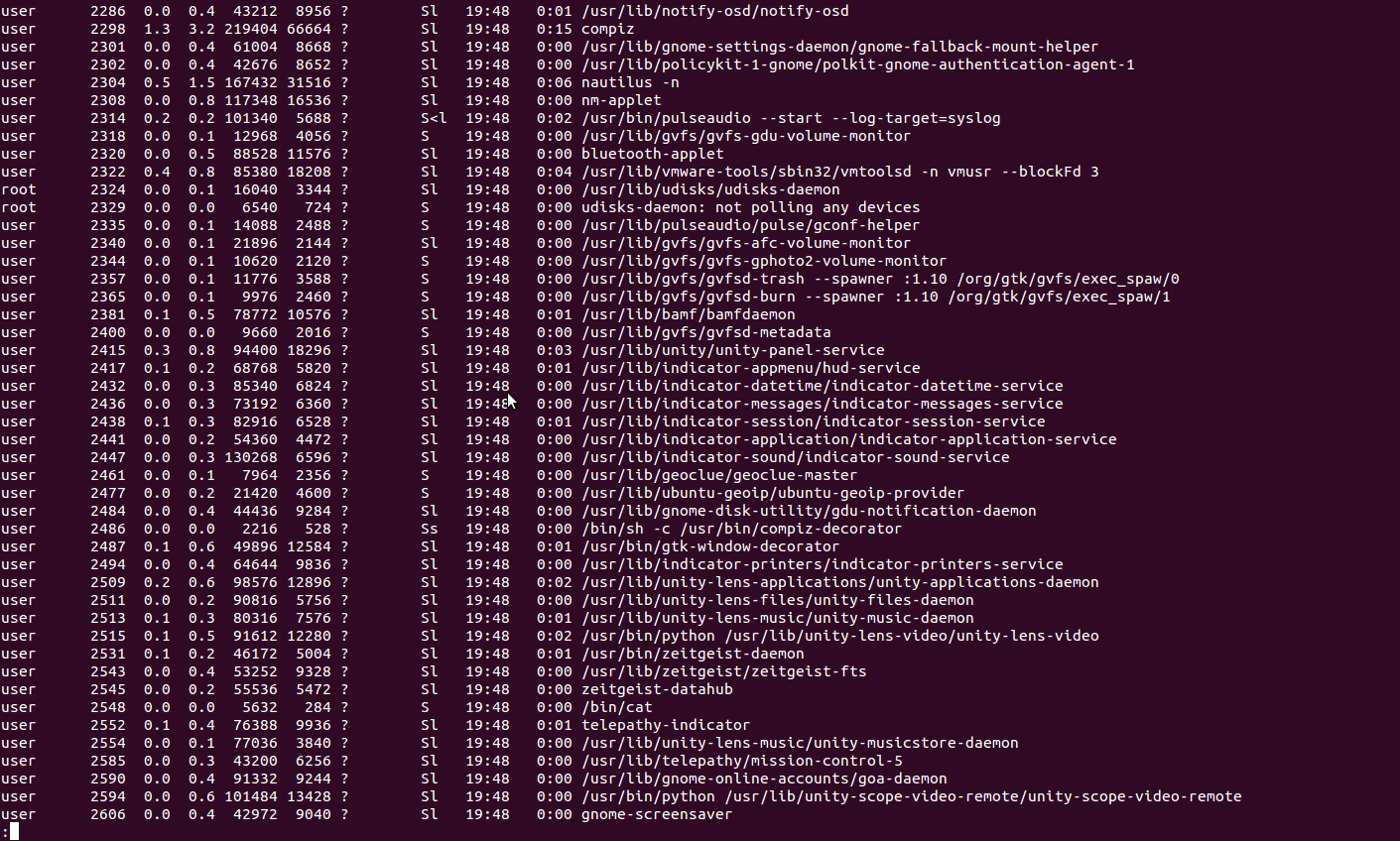
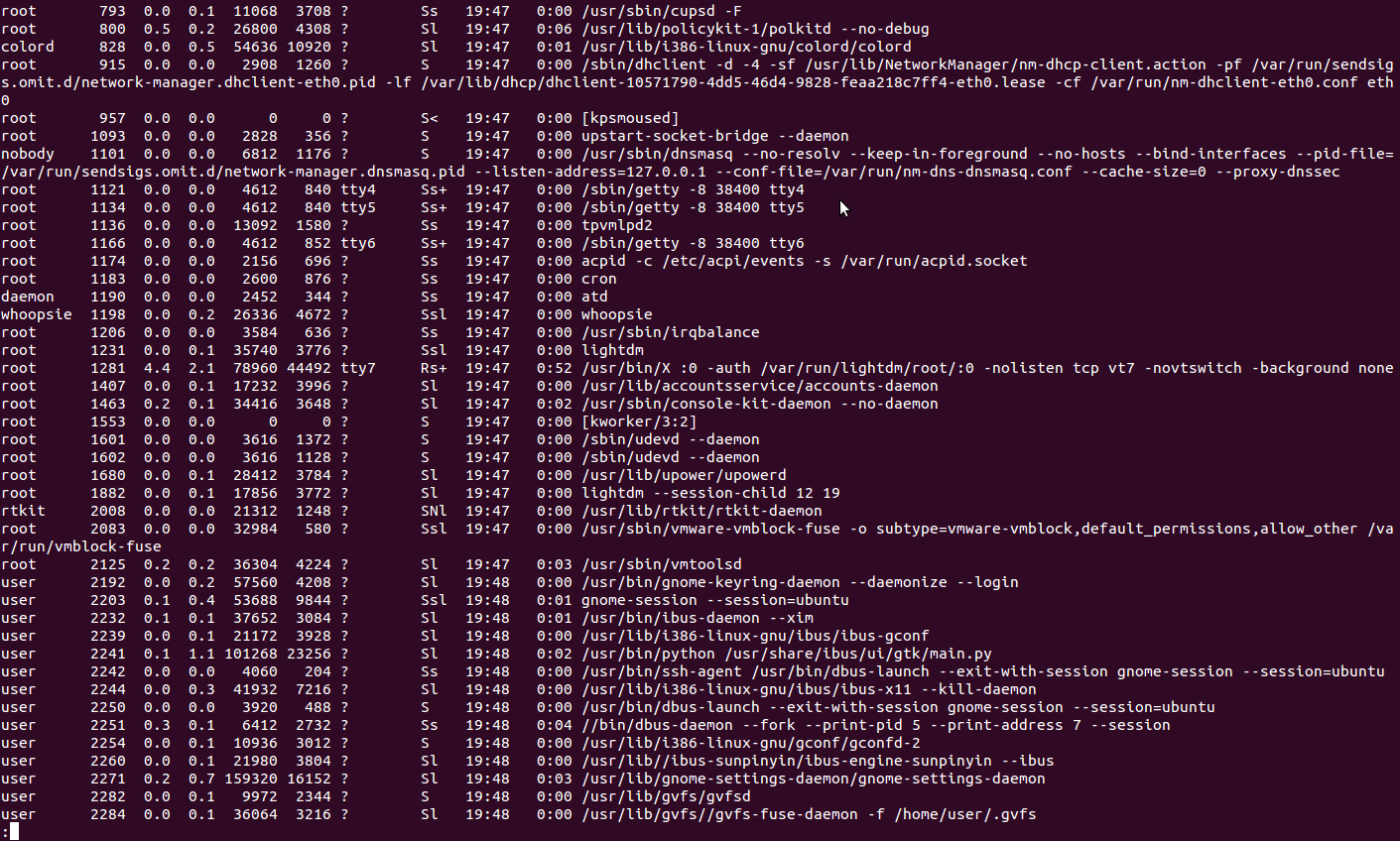
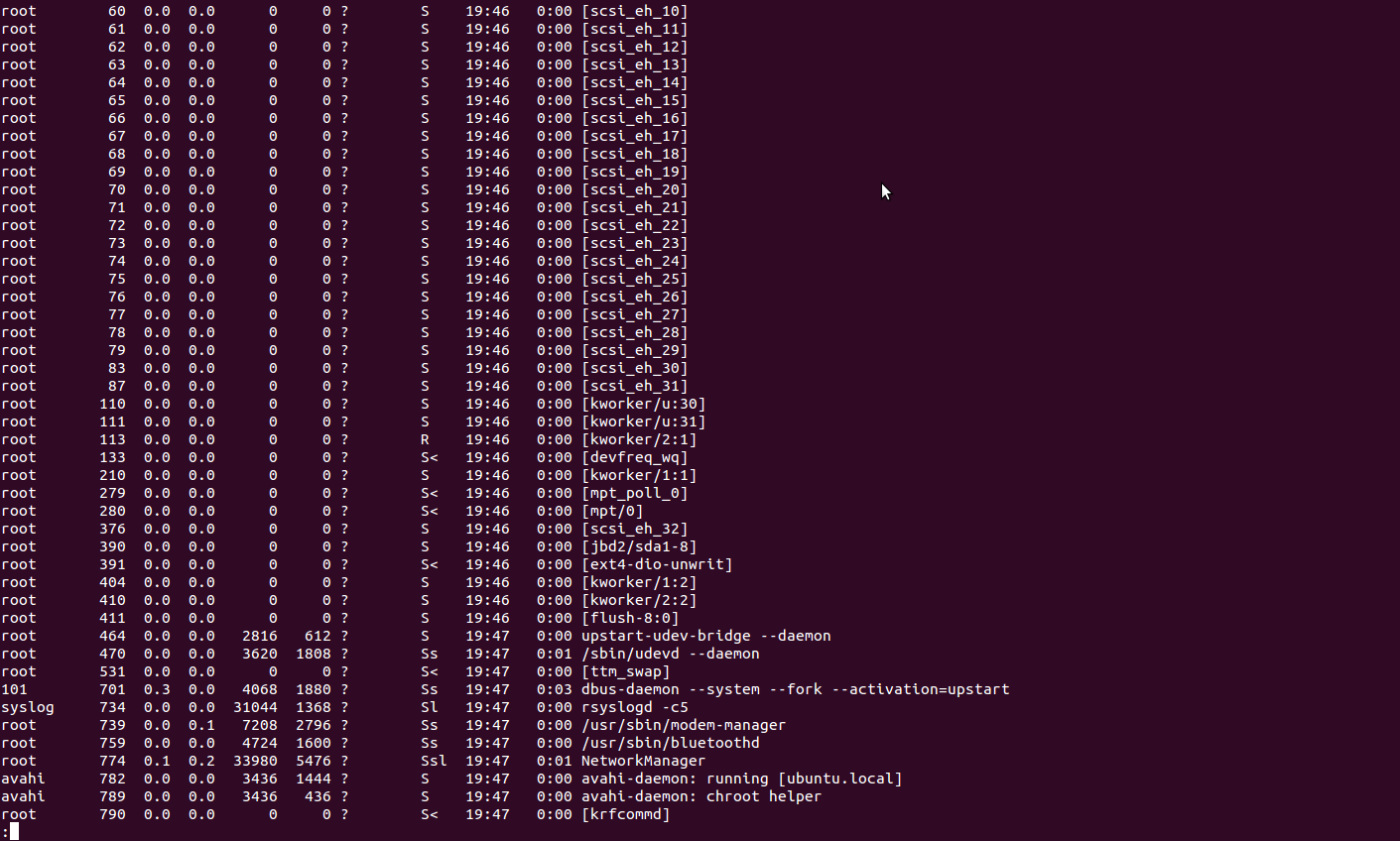
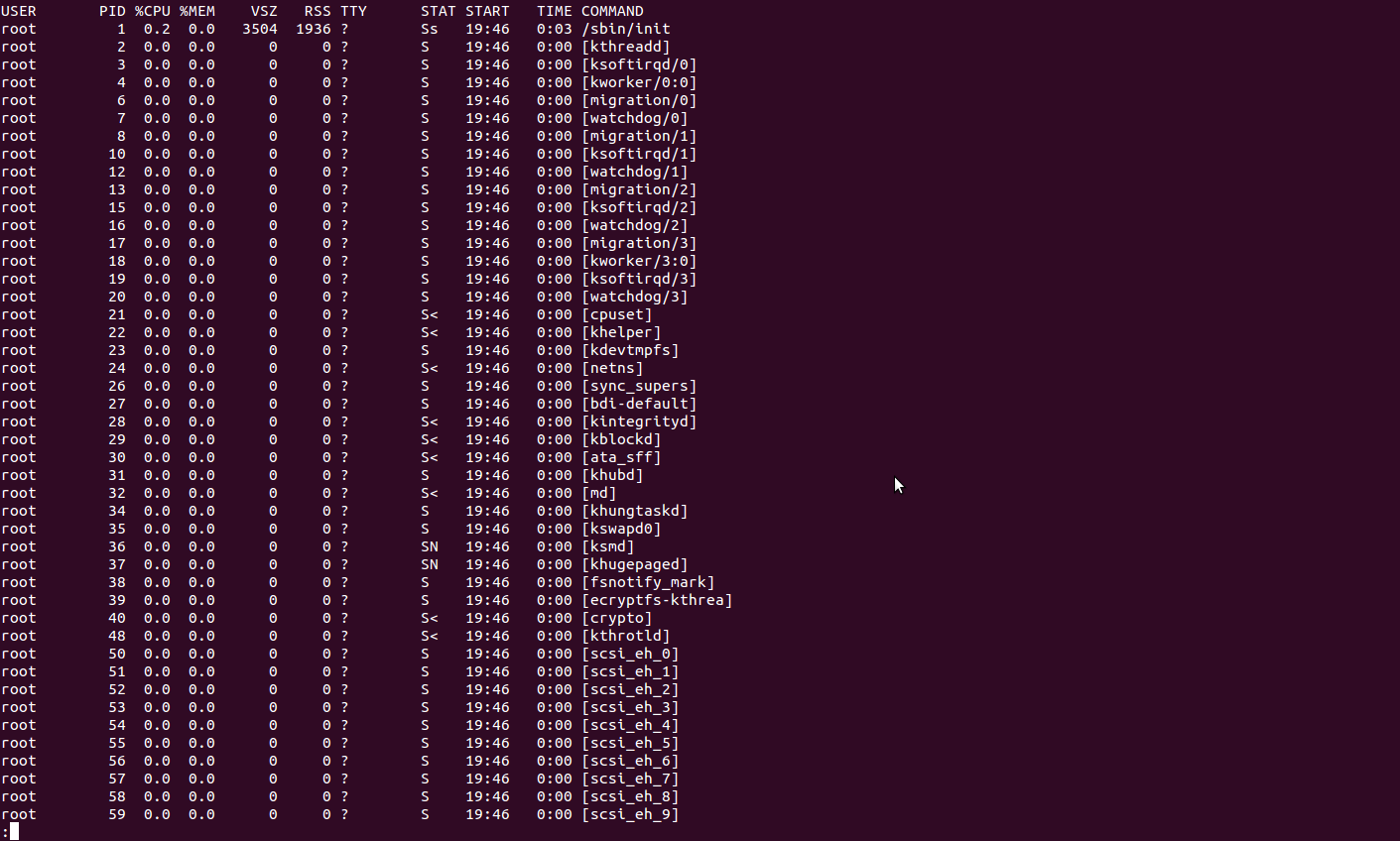
/proc/$PID/下包含进程的详细信息：

|  |  |
| --- | --- |
| 文件 | 说明 |
| cmdline | 进程的命令行 |
| environ | 进程的环境信息 |
| fd目录 | 为每个打开的文件描述符提供一个入口 |
| mem | 可通过其访问进程的内存映像 |
| stat | 包含进程的大多数信息 |
| status | 包含用户可读的进程信息 |
| cwd | 指向进程的当前工作目录 |
| exe | 指向正在被执行的文件 |
| maps | 内存映射区信息 |
| root | 进程的根目录 |
| statm | 进程对内存的使用情况 |

**实验内容:**

1. 列出你的系统中当前运行的所有进程，并解释bash进程的各项信息的含义（不包括RSS和SHR）。

I.所有运行进程：



II. bash进程的各项信息的含义

PID：进程ID

CPU：进程占用CPU的百分比

MEM：进程占用内存的百分比

VSZ：进程占用虚拟内存大小

RSS：进程占用物理内存大小

TTY：控制终端ID

STAT：当前的进程状态

START：进程初始运行时间

TIME：进程使用的CPU时间

COMMAND:发起进程的命令的名称

1. 分别从至少三个虚拟终端登录，以树状形式列出你的系统中当前运行的所有进程及其PID。找出你当前运行进程的所有祖先进程，并说明其各自的作用。分析Linux系统中的进程的家族关系。

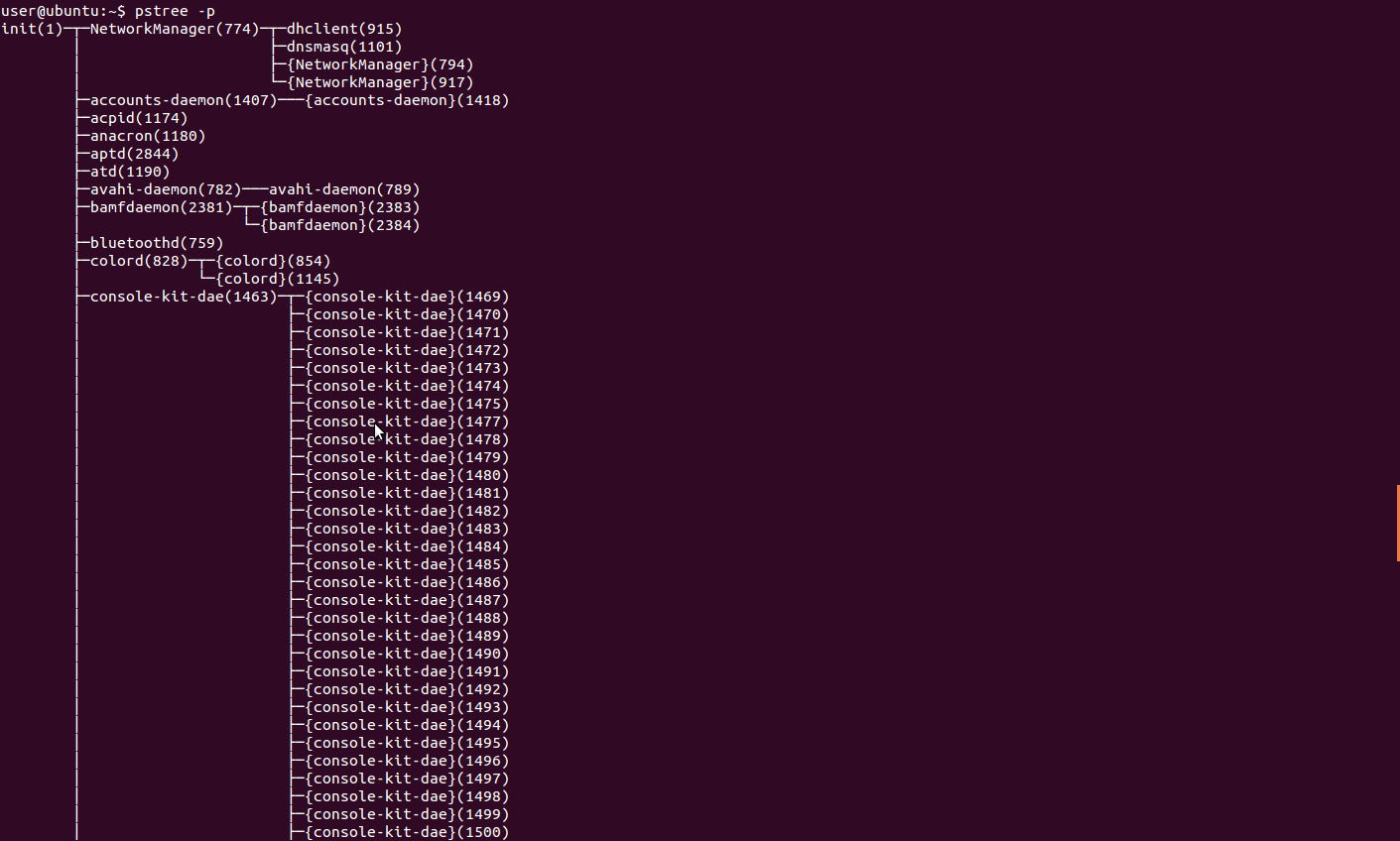
I．虚拟终端：Ctrl+Alt+Fn(n=1~6)

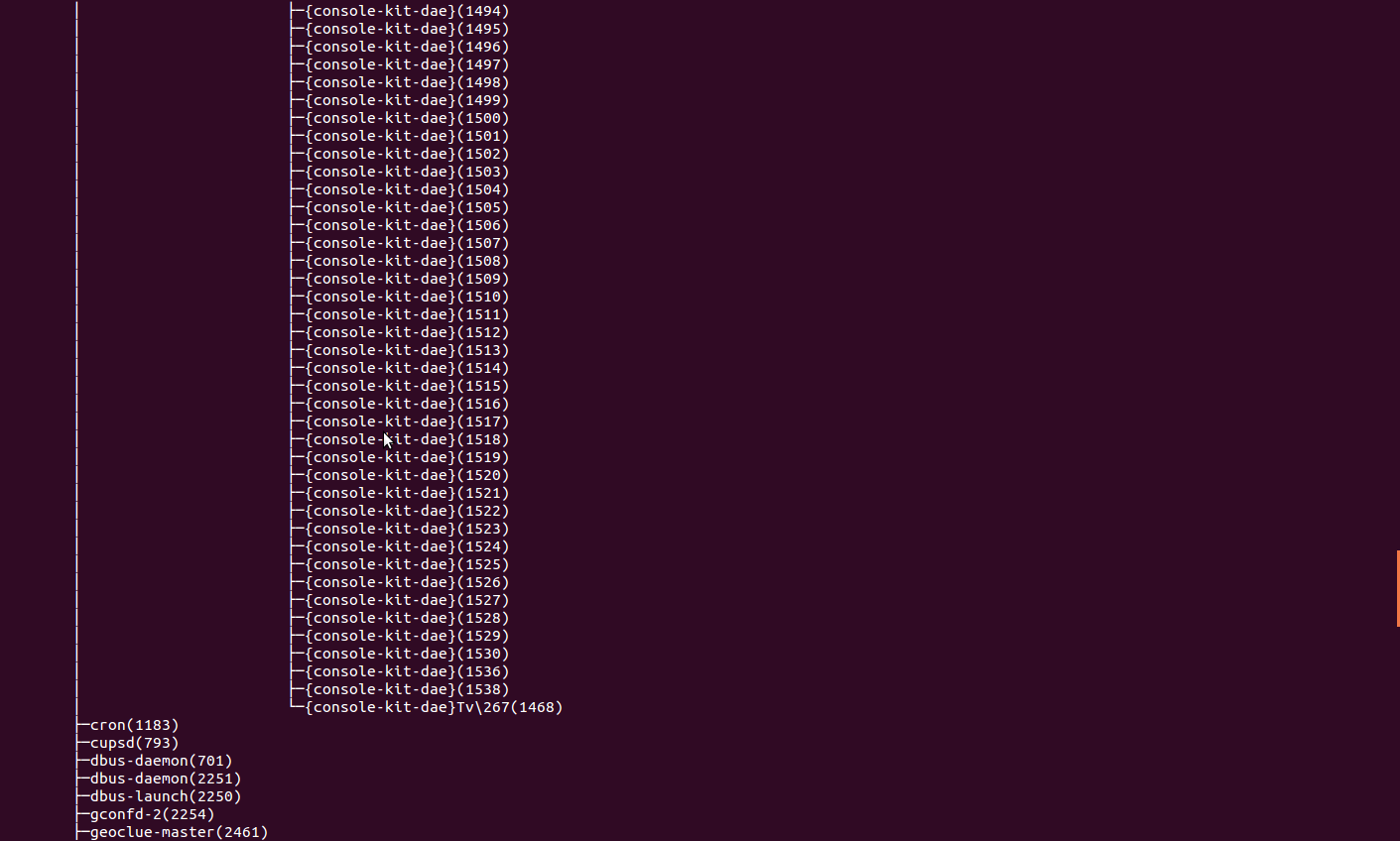
Ctrl+Alt+F7进入linux图形操作界面

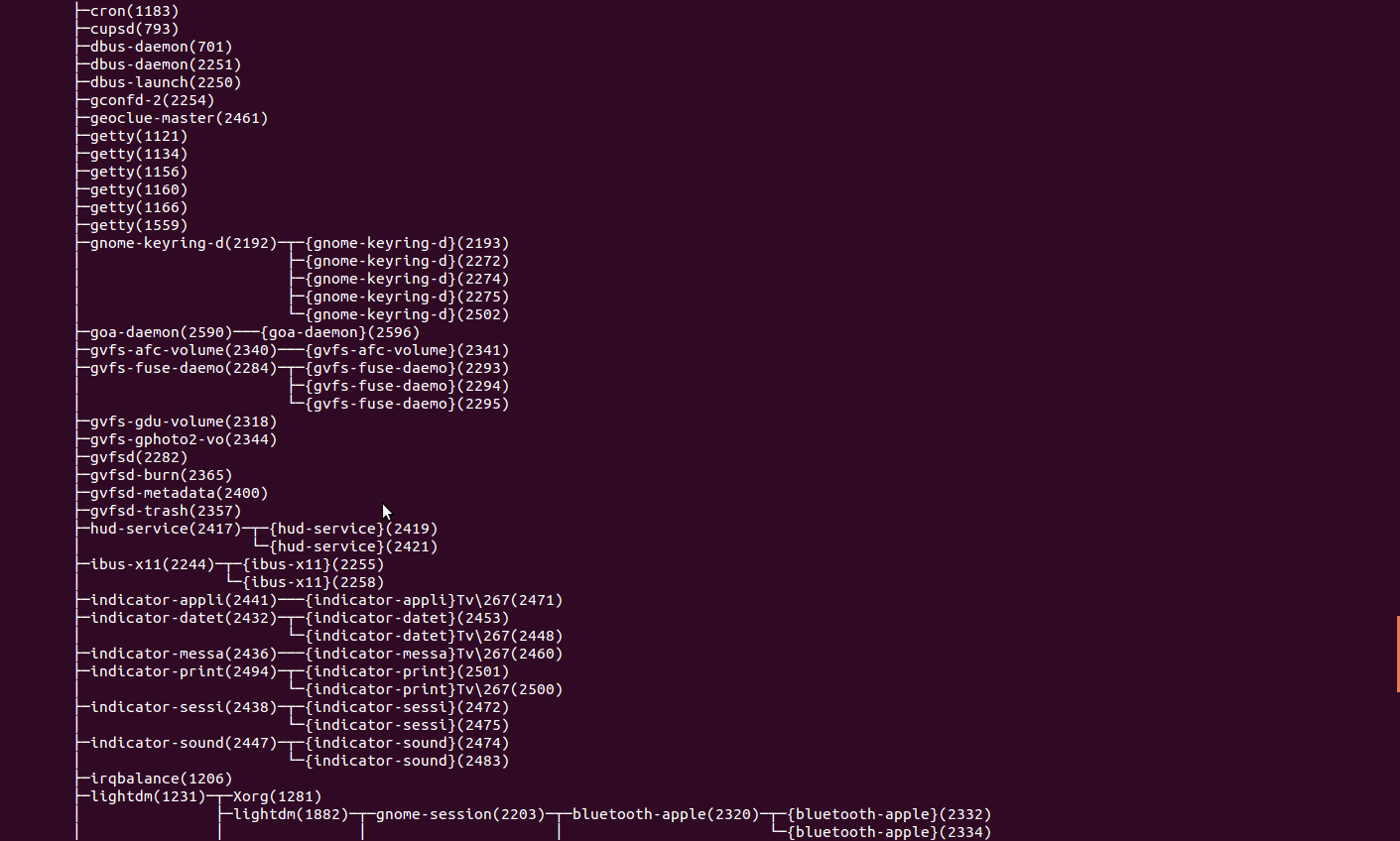
Ctrl+Alt+F1~F6进入控制台字符操作界面，显示的是黑屏，需要进行登录(login:user/root password:1)

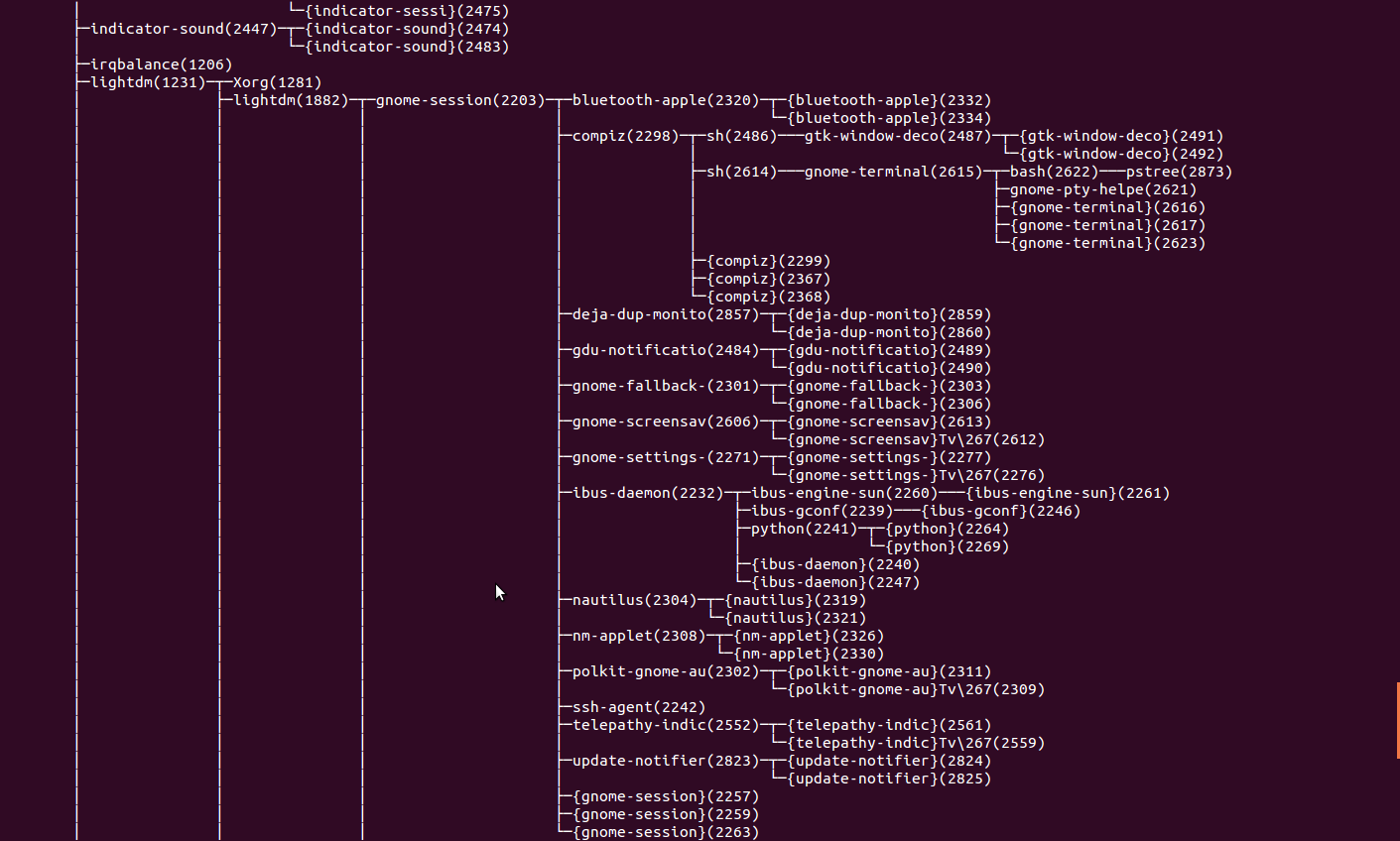
进行pstree –p操作，不同界面结果都相同

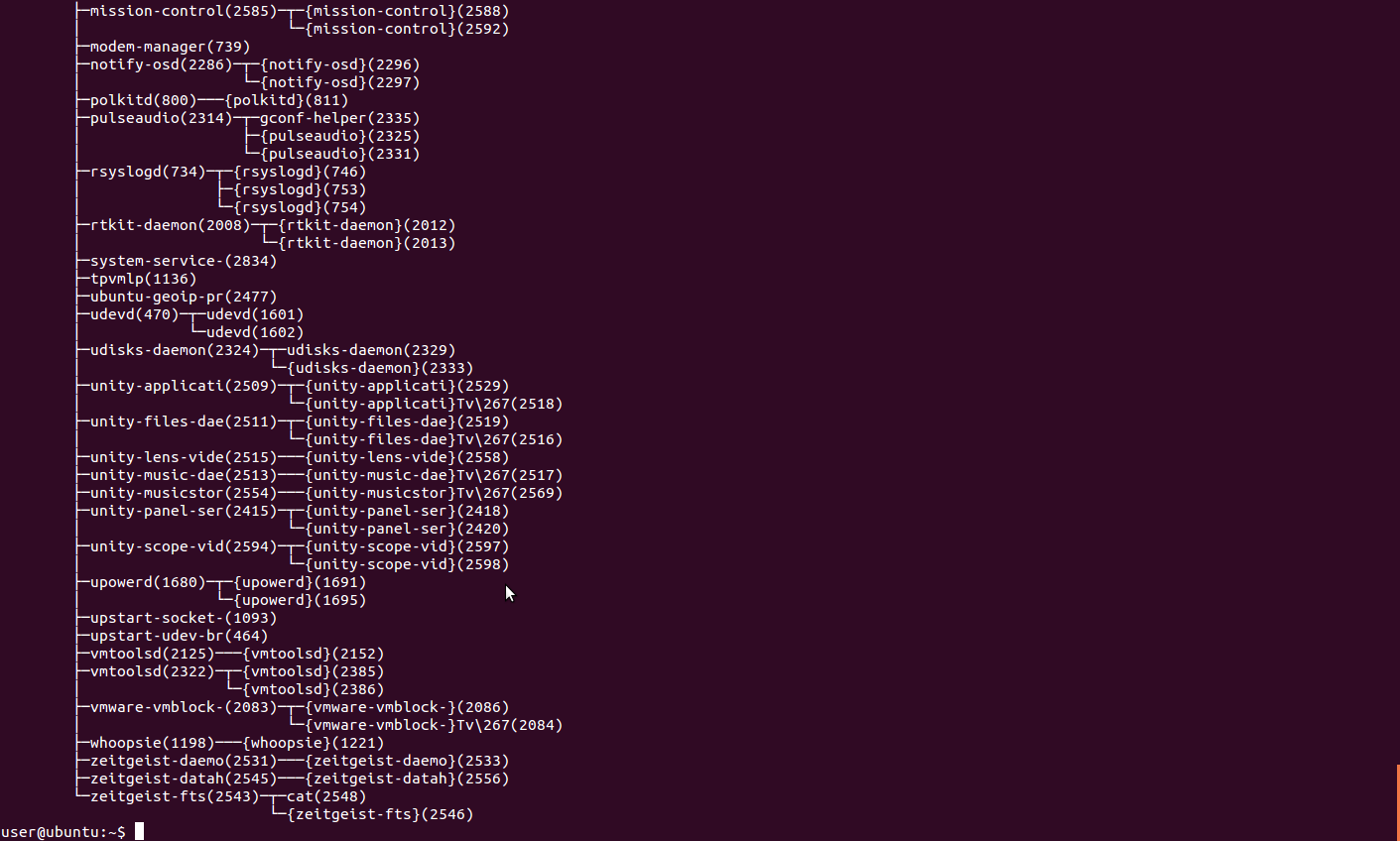
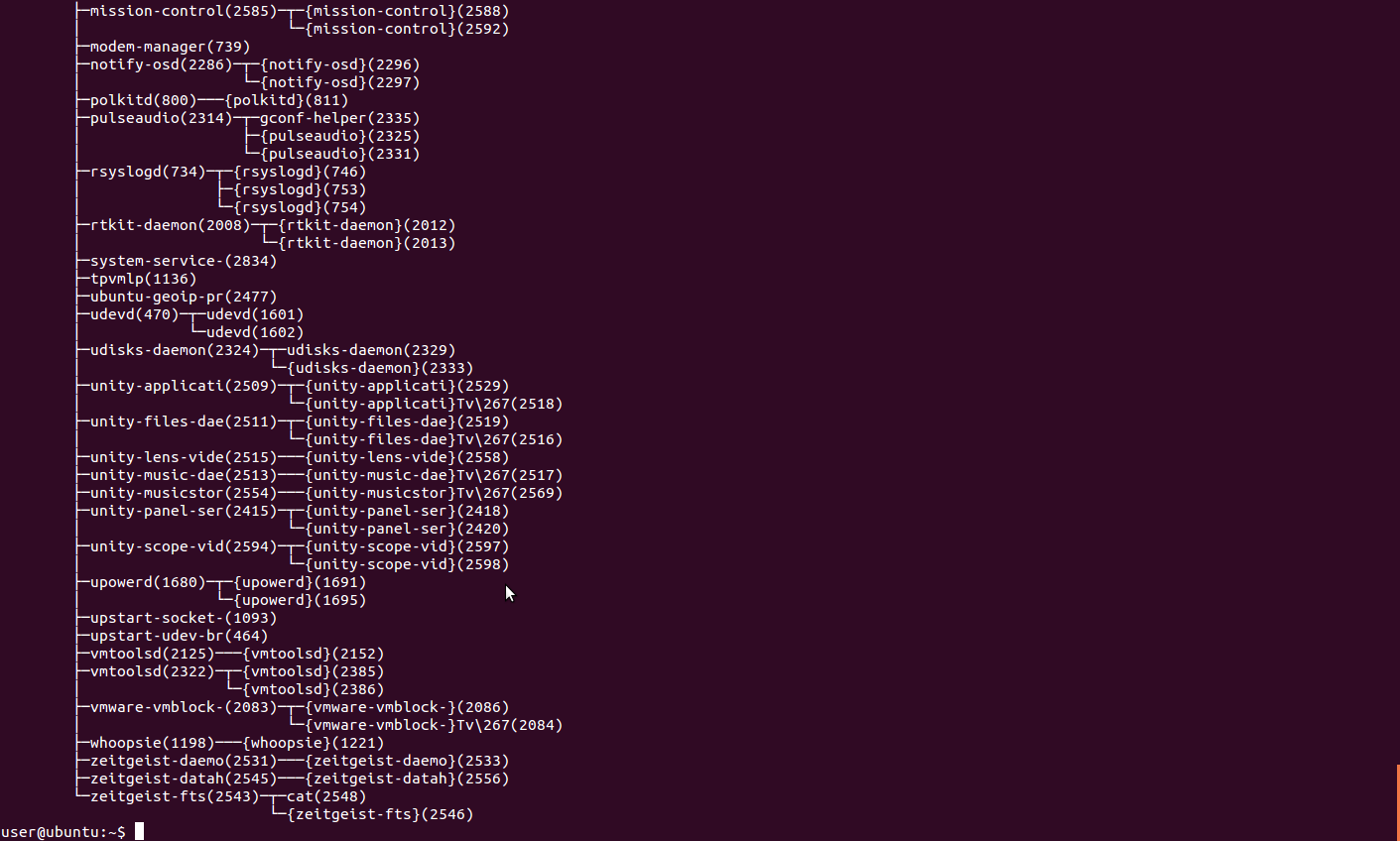
II. 以树状形式列出你的系统中当前运行的所有进程及其PID：











III.终极祖先进程Init作用：

是linux系统启动时第一个启动的进程，用于启动其它进程。

IV.Linux系统中的进程的家族关系：

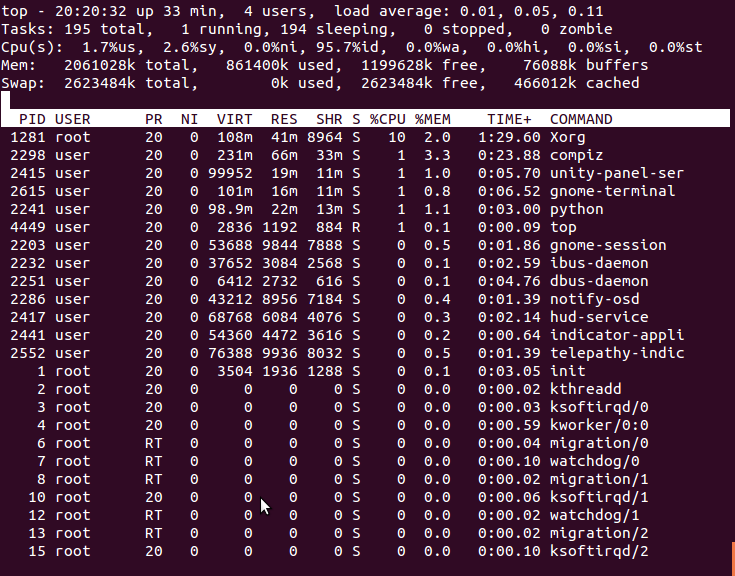
树形关系。生成进程：父进程生成子进程，从顶到底；关闭进程：先子进程关闭，再父进程关闭，从底到顶。

1. 试验有几种方法可以列出系统中的所有运行进程？

I.ps (-aux)

II.pstree (-p/-u)

III.top(运行的进程依次显示，但是并没有一次全部显示)



1. 执行下列命令和操作：

$sleep 180

^Z

$jobs

$sleep 240&

$sleep 300&

$sleep 330&

$sleep 630&

$jobs

$fg 4

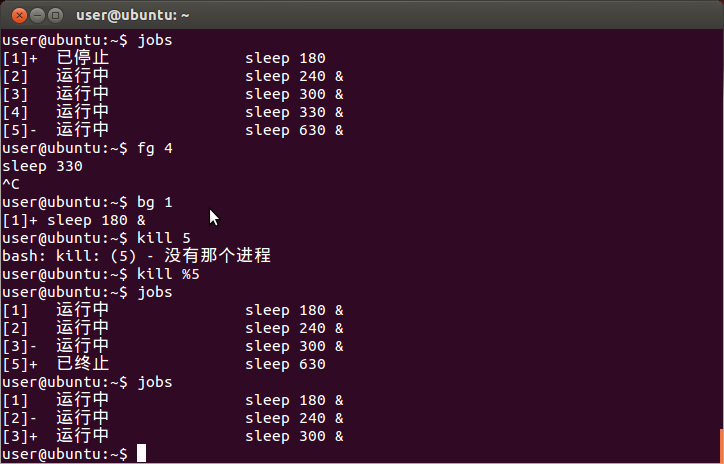
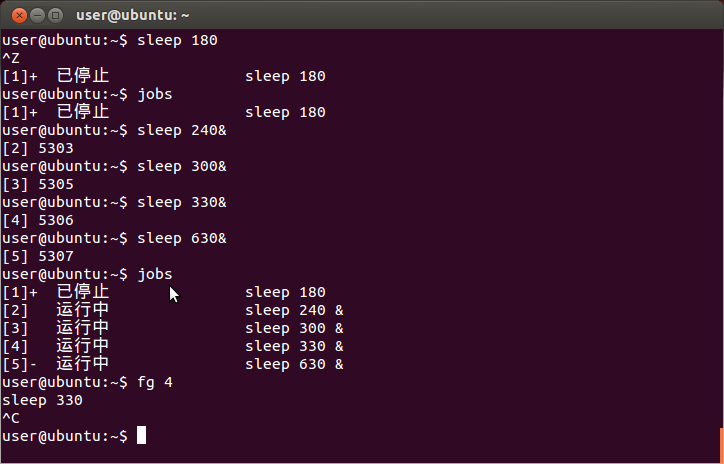
^C

$bg 1

$kill 5

解释命令的作用和执行结果。

I.



II.

$sleep 180 #创建一个新进程，休眠180秒

^Z #挂起进程，使进程停止

$jobs #列出作业控制的内容

$sleep 240& #以后台方式启动进程

$sleep 300&

$sleep 330&

$sleep 630&

$jobs

$fg 4 #将进程4切换到前台，即处于休眠状态

^C #终止进程4

$bg 1 #将进程1切换到后台运行，没有起到任何作用

$kill %5 #杀死进程5，然后再执行jobs指令，进程5终止，后执行jobs指令，进程5消失

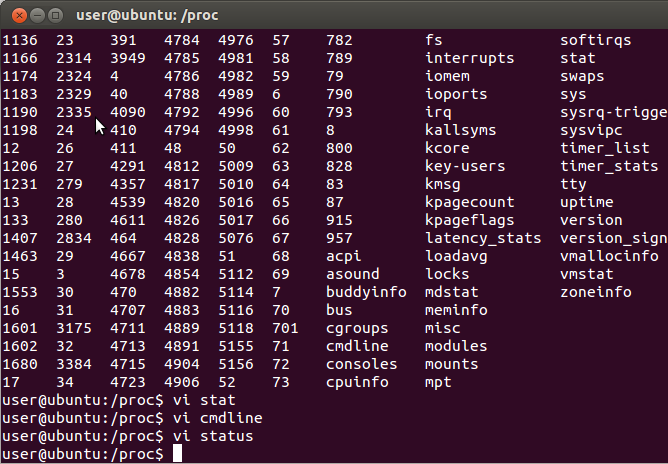
1. 利用/proc文件系统，选择一个进程，如-bash，列出并解释/proc/$PID/cmdline, /proc/$PID/stat, /proc/$PID/status文件的内容。

cmdline 进程的命令行

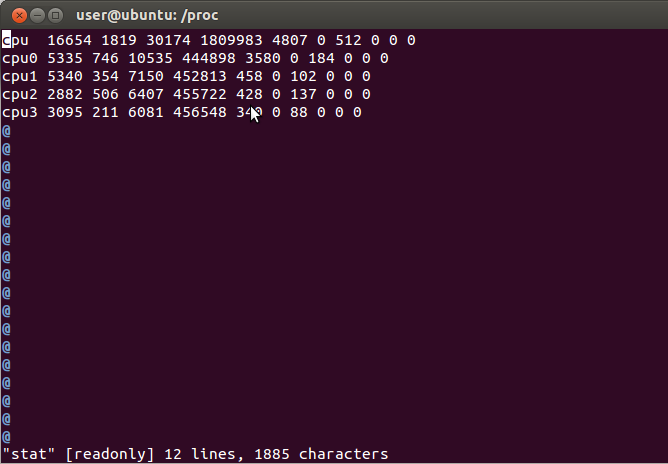
stat 包含进程的大多数信息

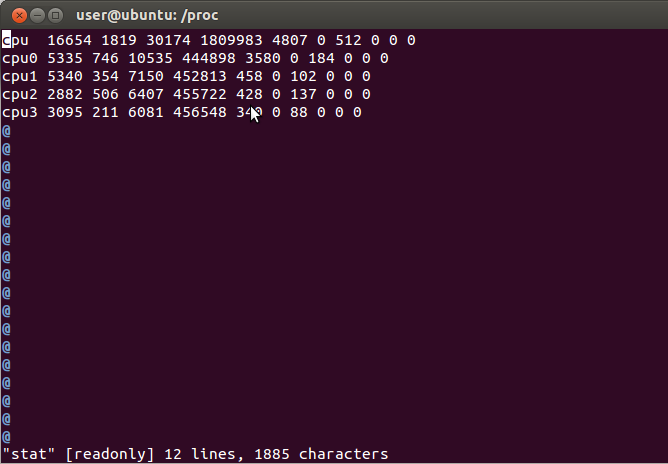
status 包含用户可读的进程信息

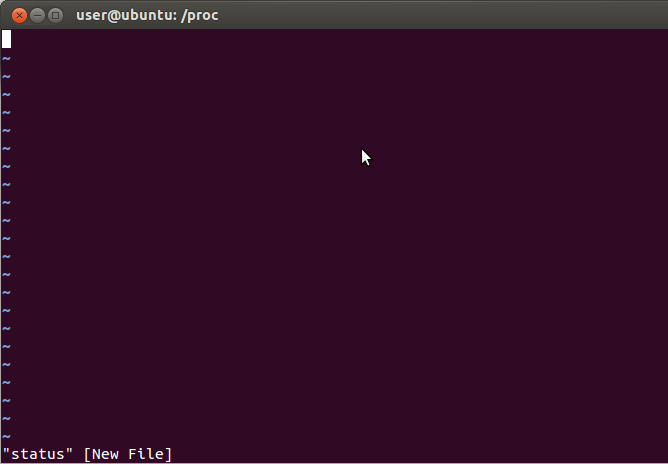




在/proc下的文件：

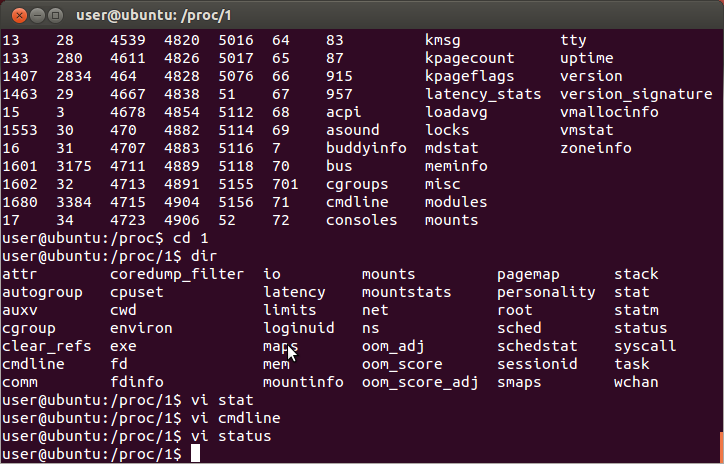


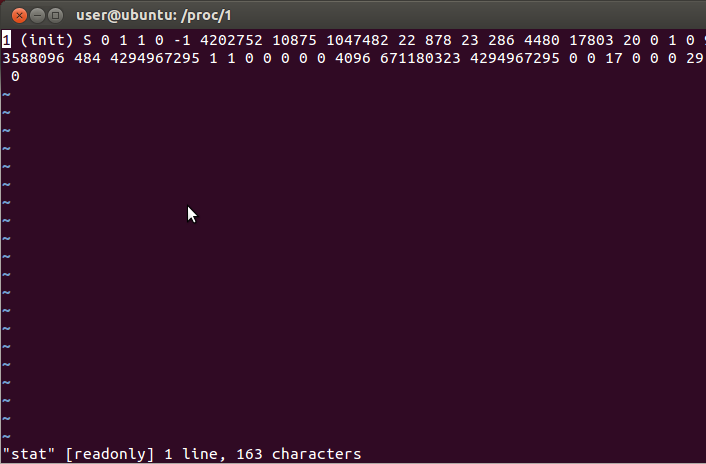




在/proc/1下的文件：

I.





来自网络：

[root@localhost ~]# cat /proc/6873/stat

6873 (a.out) R 6723 6873 6723 34819 6873 8388608 77 0 0 0 41958 31 0 0 25 0 3 0 5882654 1409024 56 4294967295 134512640 134513720 3215579040 0 2097798 0 0 0 0 0 0 0 17 0 0 0 [root@localhost ~]#

每个参数意思为：

参数 解释

pid=6873 进程(包括轻量级进程，即线程)号

comm=a.out 应用程序或命令的名字

task\_state=R 任务的状态，R:runnign, S:sleeping (TASK\_INTERRUPTIBLE), D:disk sleep (TASK\_UNINTERRUPTIBLE), T: stopped, T:tracing stop,Z:zombie, X:dead

ppid=6723 父进程ID

pgid=6873 线程组号

sid=6723 c该任务所在的会话组ID

tty\_nr=34819(pts/3) 该任务的tty终端的设备号，INT（34817/256）=主设备号，（34817-主设备号）=次设备号

tty\_pgrp=6873 终端的进程组号，当前运行在该任务所在终端的前台任务(包括shell 应用程序)的PID。

task->flags=8388608 进程标志位，查看该任务的特性www.linuxidc.com

min\_flt=77 该任务不需要从硬盘拷数据而发生的缺页（次缺页）的次数

cmin\_flt=0 累计的该任务的所有的waited-for进程曾经发生的次缺页的次数目

maj\_flt=0 该任务需要从硬盘拷数据而发生的缺页（主缺页）的次数

cmaj\_flt=0 累计的该任务的所有的waited-for进程曾经发生的主缺页的次数目

utime=1587 该任务在用户态运行的时间，单位为jiffies

stime=1 该任务在核心态运行的时间，单位为jiffies

cutime=0 累计的该任务的所有的waited-for进程曾经在用户态运行的时间，单位为jiffies

cstime=0 累计的该任务的所有的waited-for进程曾经在核心态运行的时间，单位为jiffies

priority=25 任务的动态优先级

nice=0 任务的静态优先级

num\_threads=3 该任务所在的线程组里线程的个数

it\_real\_value=0 由于计时间隔导致的下一个 SIGALRM 发送进程的时延，以 jiffy 为单位.

start\_time=5882654 该任务启动的时间，单位为jiffies

vsize=1409024（page） 该任务的虚拟地址空间大小

rss=56(page) 该任务当前驻留物理地址空间的大小

Number of pages the process has in real memory,minu 3 for administrative purpose.

这些页可能用于代码，数据和栈。

rlim=4294967295（bytes） 该任务能驻留物理地址空间的最大值

start\_code=134512640 该任务在虚拟地址空间的代码段的起始地址

end\_code=134513720 该任务在虚拟地址空间的代码段的结束地址

start\_stack=3215579040 该任务在虚拟地址空间的栈的结束地址

kstkesp=0 esp(32 位堆栈指针) 的当前值, 与在进程的内核堆栈页得到的一致.

kstkeip=2097798 指向将要执行的指令的指针, EIP(32 位指令指针)的当前值.

pendingsig=0 待处理信号的位图，记录发送给进程的普通信号

block\_sig=0 阻塞信号的位图

sigign=0 忽略的信号的位图

sigcatch=082985 被俘获的信号的位图

wchan=0 如果该进程是睡眠状态，该值给出调度的调用点

nswap 被swapped的页数，当前没用

cnswap 所有子进程被swapped的页数的和，当前没用

exit\_signal=17 该进程结束时，向父进程所发送的信号

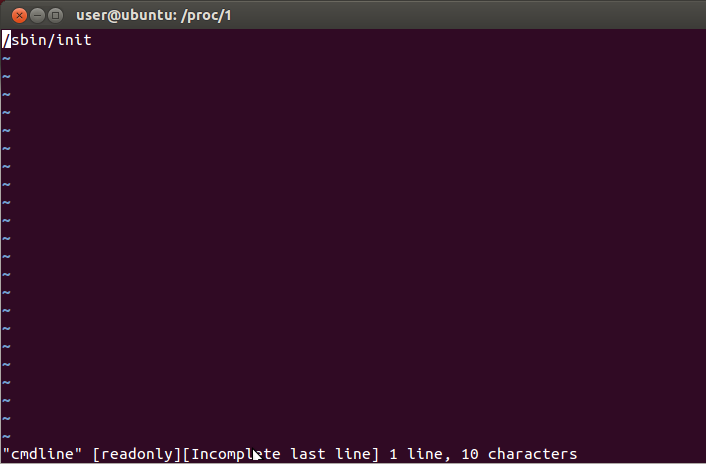
task\_cpu(task)=0 运行在哪个CPU上

task\_rt\_priority=0 实时进程的相对优先级别

task\_policy=0 进程的调度策略，0=非实时进程，1=FIFO实时进程；2=RR实时进程

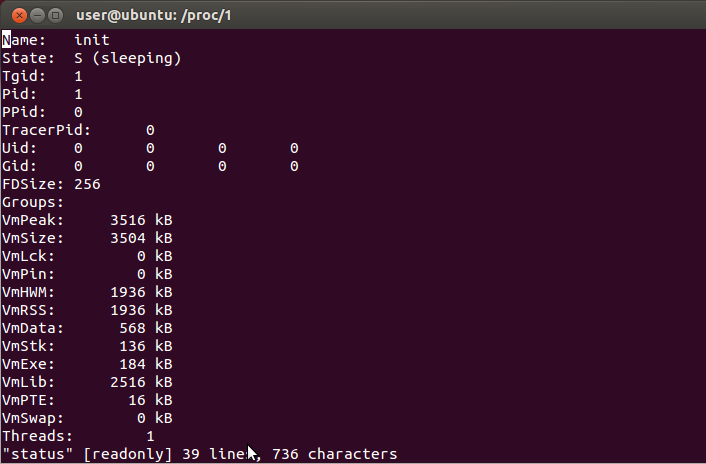
本篇文章来源于 Linux公社网站(www.linuxidc.com) 原文链接：<http://www.linuxidc.com/Linux/2010-12/30589.htm>

II.



发起进程的命令的名称

III.



Name：init 进程名称

State：S(sleeping) 休眠状态

Pid：1 进程编号

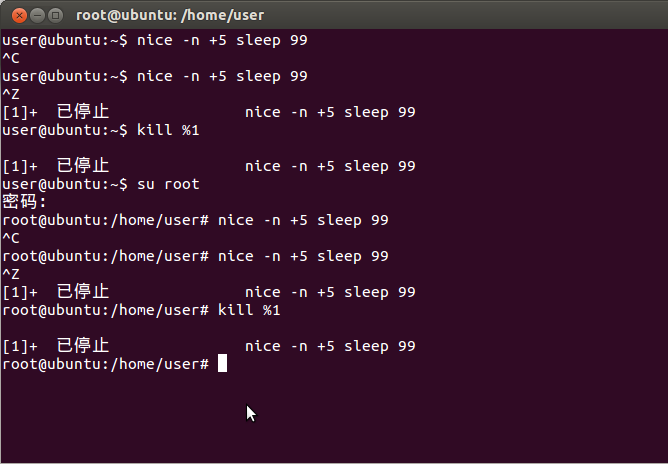
Threads：该进程有1个线程

“status”： [readonly] 该进程程序只读

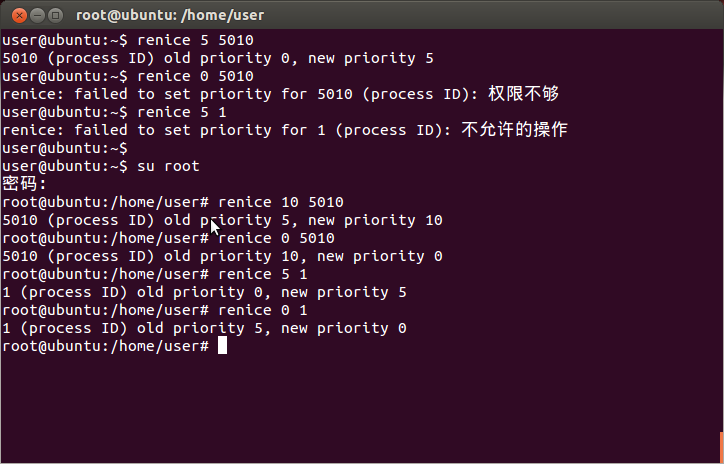
1. 分别以root和普通用户身份运行一个进程，使用nice、renice和top设置或者改变其优先级，并验证是否成功。

重点见III.top的说明

I.nice

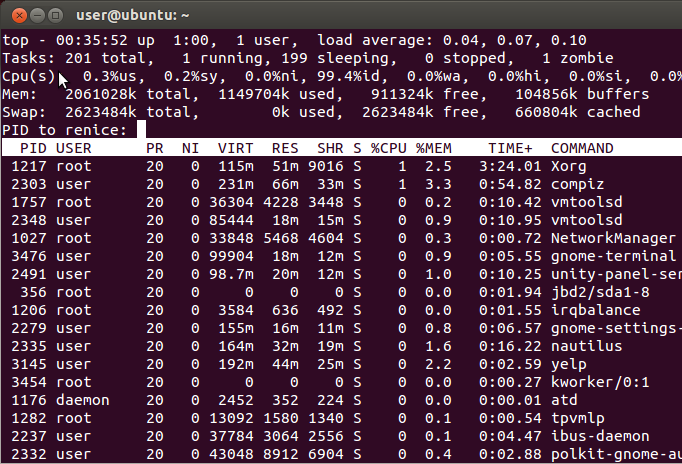


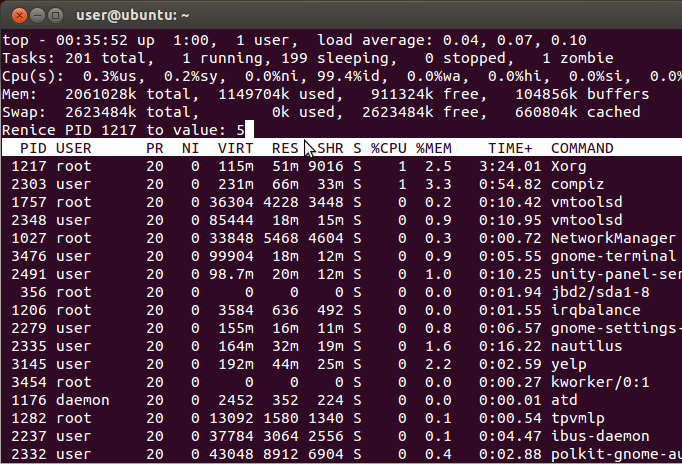
II.renice

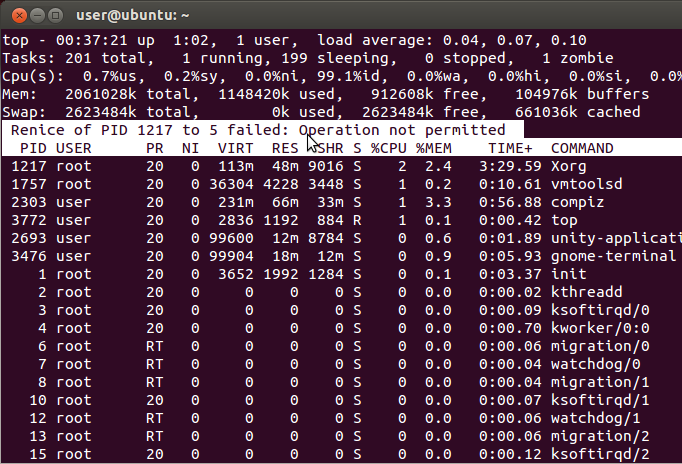


III.top

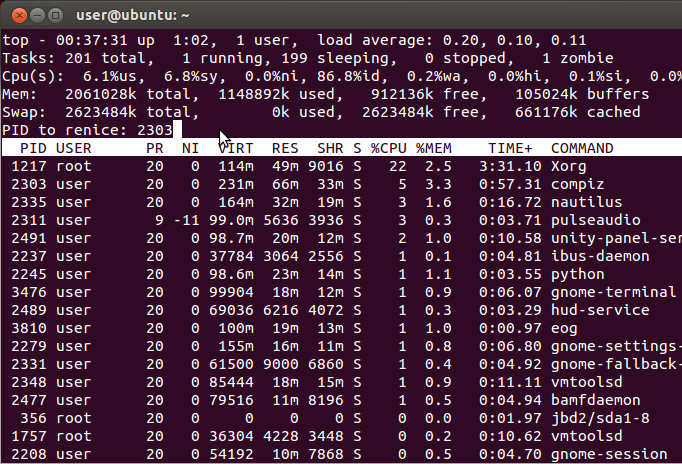
Situation1(user模式下不能改root的进程)：

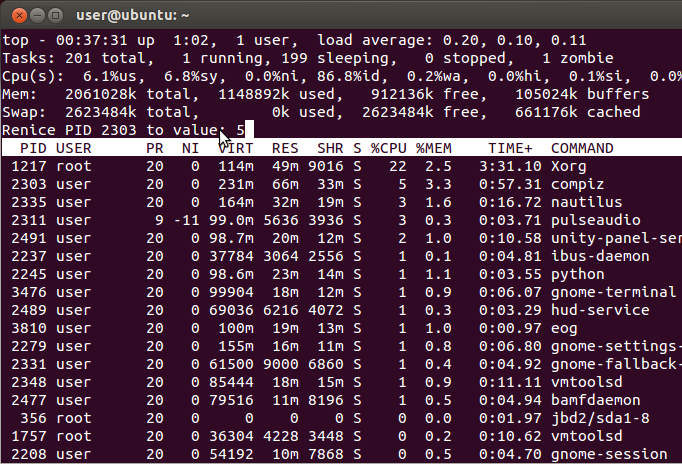


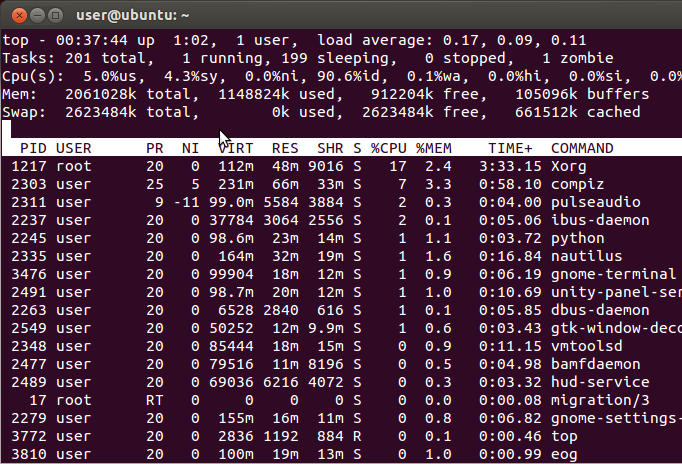




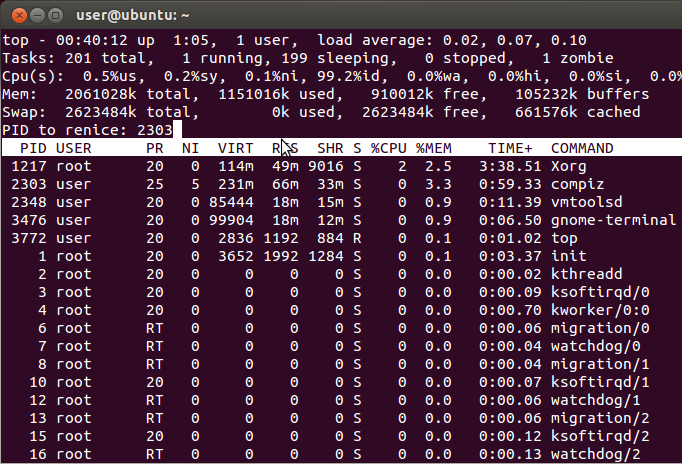
Situation2(user模式下可以改非root的进程)：：

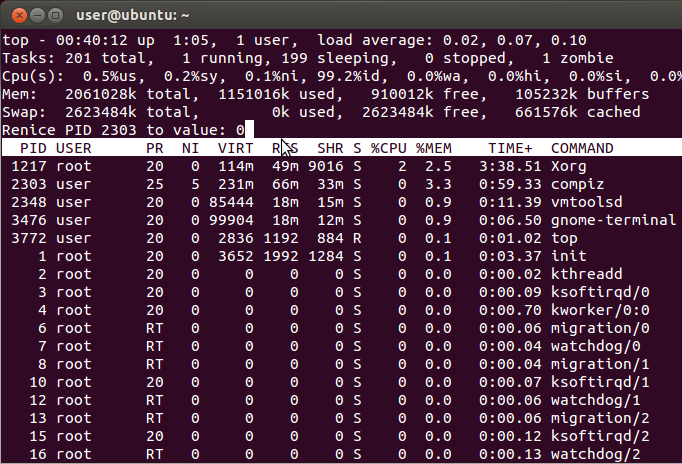


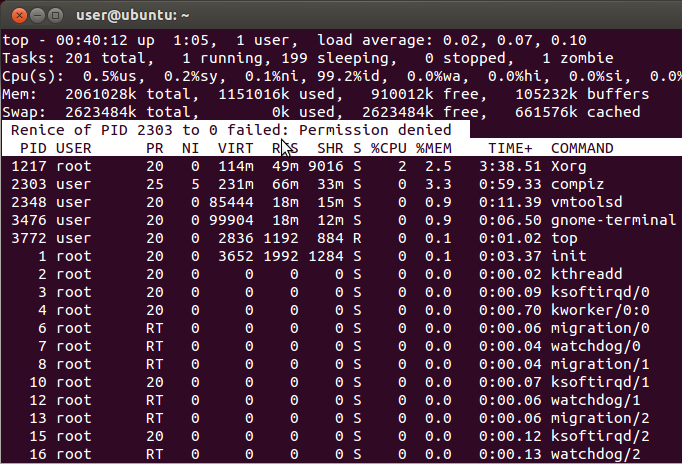




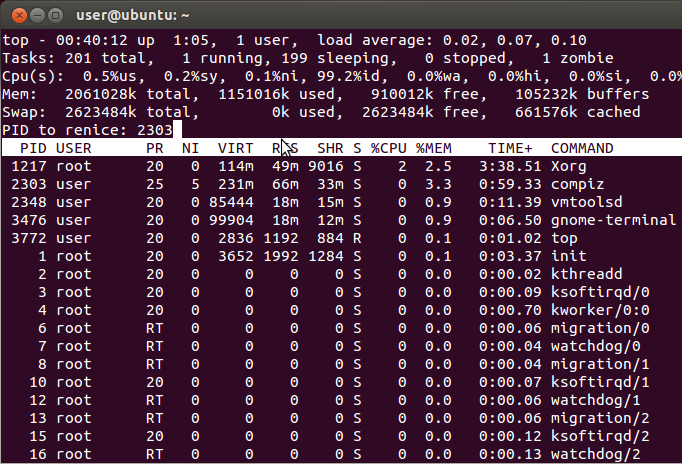
Situation3(user模式下不能把nice值改小)：

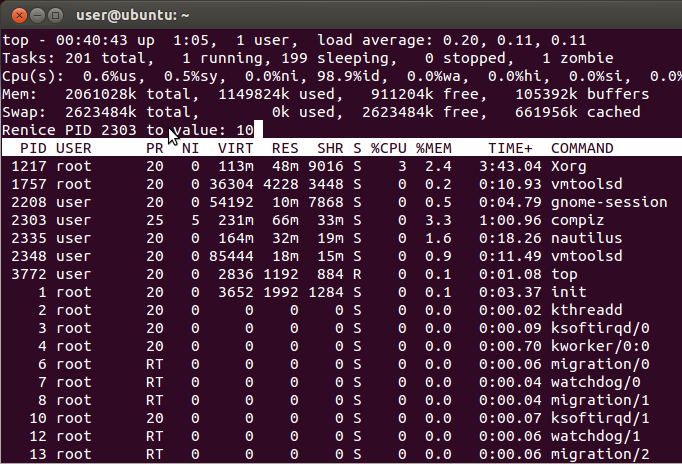


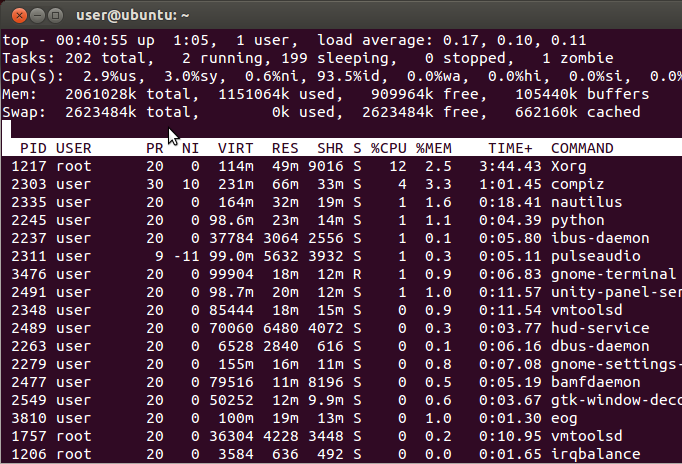




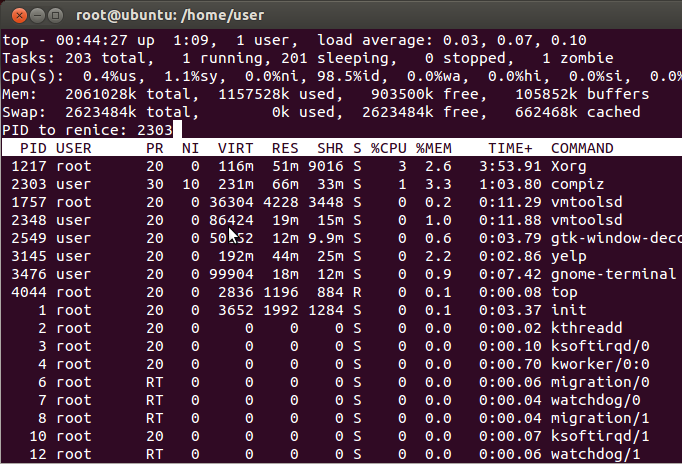
Situation4(user模式下不能把nice值改大)：

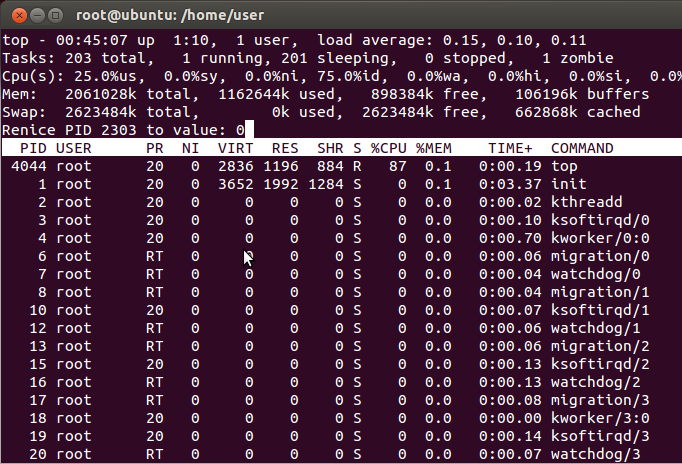


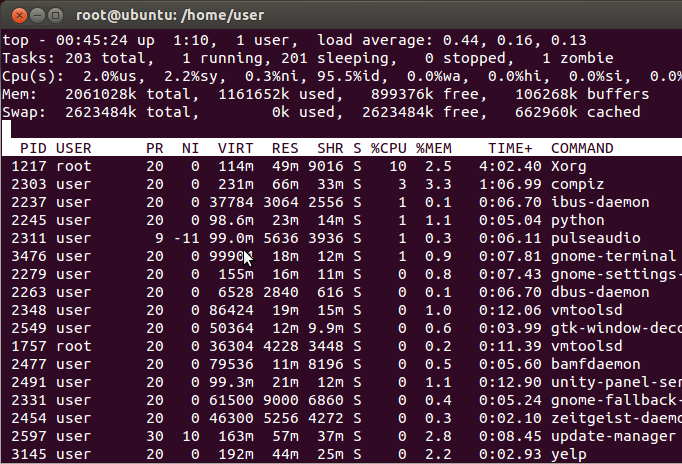




Situation6(root模式下可以把数值改小)：







Situation7(root模式下可以修改root的进程)：

