进程和作业管理

毛迪林
dlmao@fudan.edu.cn

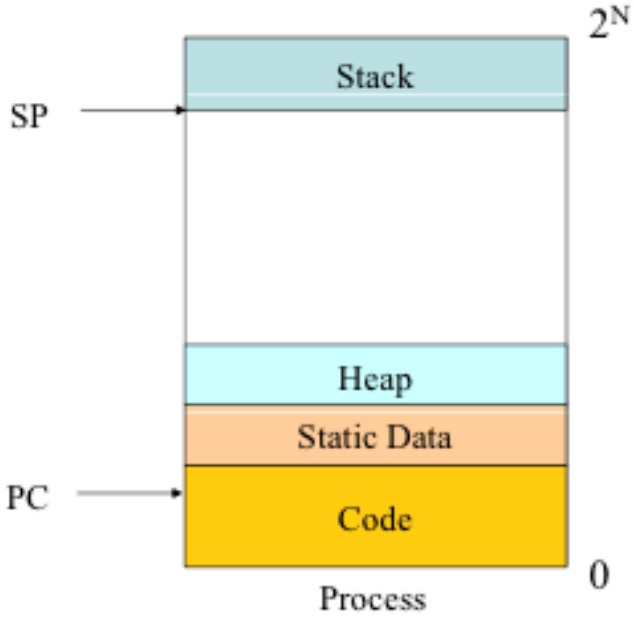
进程(Process)

查看当前shell PID： echo $$

- 进程：指的是一个加载到内存中执行的程序
  - 每个进程有自己相应的上下文（数据和执行）
- 进程不是程序或者应用
  - 我们可以多次执行某个程序
  - 程序的每次执行对应一个进程
  - 每个进程有自己的地址空间和上下文
  - 每个进程从自己的角度看似乎完全拥有整个系统，通过操作系统的调度来保证这一点
- 内核负责管理进程
  - 每个进程分配一个唯一的ID，称为PID（进程ID）
  - 内核维护一个进程表，纪录了进程的状态信息: 实际用户ID/有效用户ID/进程ID/父进程ID/进程组ID/会话ID/控制终端/工作目录/环境等
  - 一个Linux系统同时有多个进程执行，有的是用户启动的，有的是系统启动的
  - 调度器选择正在执行的多个进程中的其中一个或者多个（多处理器系统），让其运行一段短的时刻(10毫秒CPU时间)

进程上下文

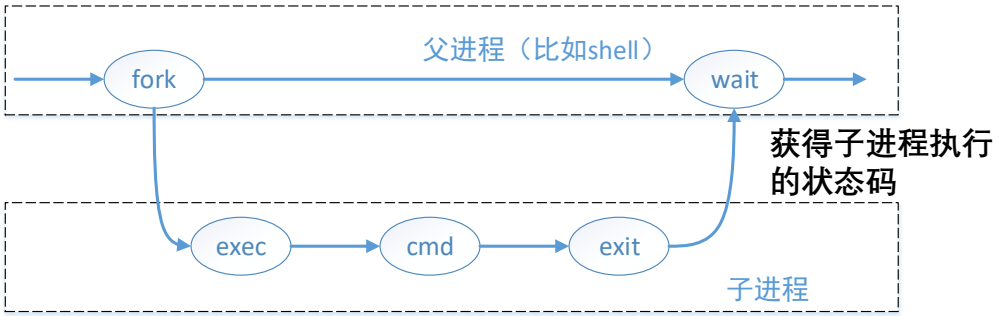
- PC： Program Counter
  - 要执行的下一个指令的地址
- Stack: 临时数据
  - 维护函数调用的状态，包括函数参数、返回地址、本地变量
- Heap:动态分配的内存空间
  - 对象，字符串等
- Static Data: 全局变量
- Code： 执行的程序代码



进程的启动

查看当前shell PID： echo $$
使用ps命令查看各个进程的pid等信息

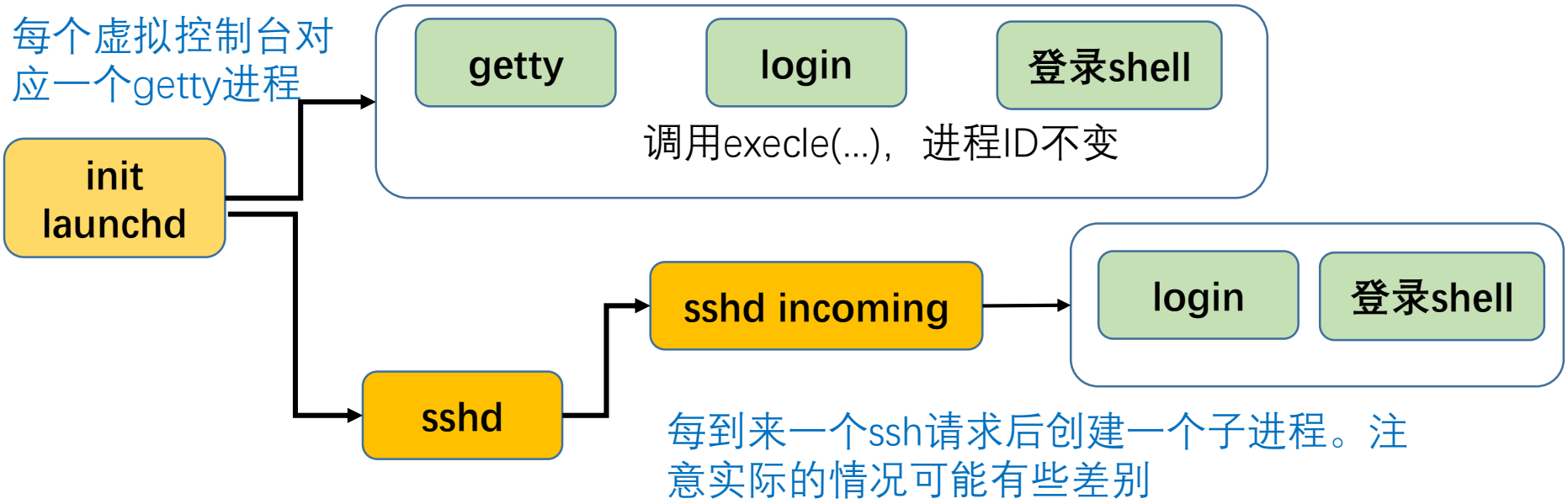
- 进程树的根为init或systemd进程(pid=1)，由操作系统引导时创建
- 父进程启动一个子进程，子进程可再启动另一个进程，形成一个进程树
- 一个进程调用fork来创建一个当前进程的克隆进程，称为子进程
- 子进程调用exec来停止执行父进程的代码，转为执行相应的程序
- 父进程应该调用wait来等待子进程结束，了解子进程执行的结果
- 子进程在执行完毕后调用exit来结束自己（子进程）
- 内核发现子进程结束时
  - 释放子进程所使用的资源
  - 但并不是马上从进程表中移走，仍然保留一些必要的信息，等待父进程获取子进程执行返回的状态码，这种进程称为ZOMBIE进程
  - 唤醒父进程(发送SIGCHILD信号)，父进程可以查看子进程返回的结果。内核此时将僵尸进程移走



孤儿进程(Orphan)

许多发行版采用systemd来替代原有的init

- 僵尸进程的父进程有可能已经异常退出，或不主动调用wait来获取子进程的状态码
- 发现某个进程退出时，内核进程init会检查看是否存在该进程的子进程，将这些子进程的父进程ID设置为init，即领养孤儿进程
- 如果父进程存在，但是没有调用wait来等待子进程结束，则子进程结束后成为遗弃（abandoned）僵尸进程
  - 即父进程还存在，但是不回收该僵尸进程
  - 用户可以通过kill杀死父进程，这样init会收养并回收该僵尸进程



进程组、会话和控制终端

man credentials

- 进程组是一个或多个进程的集合
  - 引入进程组的目的是方便给进程组的多个进程发送信号
  - 通常通过shell执行的外部程序对应着一个进程组，通过管道执行的多个命令也是属于同一个进程组
  - 每个进程属于且只属于一个进程组，进程组的ID为该进程组组长的PID
  - 每个进程可通过setpgid()加入已有的进程组或创建一个新的进程组
- 会话是一个或者多个进程组的集合
  - 通常用户登录时开启一个会话，退出时结束会话
  - 开启会话的第一个进程(session leader)称为控制进程，一个会话的ID就是控制进程的进程ID
  - 一个会话可以有一个控制终端，控制进程所打开的终端设备称为**控制终端。控制进程(一般为shell进程）**从控制终端接收输入，给进程发送信号，输出到该控制终端等
  - 一个会话中的多个进程组中，有一个进程组为前台进程组(作业)，其他为后台进程组(作业)
  - 前台作业为由控制终端控制的作业，可以接收终端的输入，也可输出到控制终端
  - 后台作业为独立于控制终端的作业，它无法接收终端的输入，但是一般可以输出到控制终端

进程组、会话和控制终端

会话

控制进程=会话首进程

登录shell

后台进程组

proc1

proc2

proc1 | proc2 &

后台进程组

proc3

proc5

proc4

proc3 | proc4 | proc5

前台进程组

终端断开时的信号(SIGHUP)

fg命令将作业放到前台，发送信号SIGCONT给该作业

bg命令将作业放到后台，发送信号SIGCONT给该作业

1. 后台作业从终端读时，终端发送SIGTTIN信号给该作业，暂停作业，同时通知用户(PS1输出时显示)
2. 后台作业缺省可输出到终端，通过stty tostop禁止后会发信号SIGTTOU给该作业，并暂停...

终端的输入及终端产生的信号

Ctrl-C SIGINT
Ctrl-\ SIGQUIT
Ctrl-Z SIGTSTP

控制终端

进程可通过/dev/tty来标识控制终端

显示进程树pstree

- 进程树的根为init或者systemd进程(pid=1)

pstree [options] [PID | USER]

查看进程ID为PID开始的进程树，如果不传递参数，缺省为1（即整个进程树）。如果传递为名字，表示查看该用户的进程对应的进程树

-p 显示进程ID

-g 显示进程组ID

-n 子节点排序为按照PID而不是名字排序

-a 显示进程的命令行参数

-l (long) 显示时不要截取内容

-s 显示PID的先辈进程

查看整个进程树，包含进程ID和进程组ID，按PID排序

pstree –pgn

```
demo@mars:~$ pstree -pgn | less
systemd(1,1)-+-systemd-journal(220,220)
             |-systemd-udevd(242,242)
             |-ModemManager(568,568)-+-{gmain}(598,568)
             |                       `-{gdbus}(615,568)
             |-rsyslogd(585,585)-+-{in:imuxsock}(603,585)
             |                   |-{in:imklog}(604,585)
             |                   `-{rs:main Q:Reg}(605,585)
```

查看当前shell开始的进程树，但也包括到根的那一段

pstree –pgns

```
demo@mars:~$ pstree -pgns $$
systemd(1,1)---sshd(750,750)---sshd(17325,17325)---sshd(17435,17325)---bash(17436,17436)-+-cat(177+
                                                                                          `-pstree(+
```

```
demo@mars:~$ pstree -pga demo
VBoxClient,2296,2294 --clipboard
  └─VBoxClient,2297,2294 --clipboard
      └─{SHCLIP},2303,2294
...
sshd,17435,17325
  └─bash,17436,17436
      ├─cat,17783,17783
      └─pstree,17993,17993 -pga demo
```

pstree –pga demo

查看用户demo的所有进程组成的进程树，包含进程ID和进程组ID和命令行参数

作业(job)

- shell如何通过控制终端来进行进程组的前台后台等切换？引入作业控制
- 内核维护进程表来纪录每个进程的信息
- shell维护一个作业表来纪录正在执行的作业
  - 一个作业通过唯一的作业号job ID标识
  - 把命令放在后台执行时（命令后加&）会返回一个作业ID和进程ID
  - 一般一个进程可能对应一个作业，返回的进程ID就是该进程的ID
  - 但是也会有多个进程对应一个作业
    - 比如由多个程序构造的管道线，返回的进程ID为最后一个程序对应的进程ID
    - 通过子shell运行多个命令，返回子shell的进程ID
  - 后台进程缺省可以输出到控制终端，也可通过stty tostop禁止后台进程输出到该控制终端
  - 后台作业的结束、停止等信息缺省会等待下一个shell提示出现时才显示
    - 避免干扰当前用户的输入
    - 可通过set –o notify开启作业结束时信息立即通知选项

```
demo@mars:~$ (sleep 5; head /etc/passwd)&
[2] 18035
demo@mars:~$ pstree -pgn $$
bash(17436,17436)─┬─cat(17783,17783)
                  ├─bash(18035,18035)───sleep(18036,18035)
                  └─pstree(18037,18037)
demo@mars:~$ root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin

[2]-  Done                    ( sleep 5; head /etc/passwd )
demo@mars:~$
```

后台进程输出

```
demo@mars:~$ cat &
[1] 18140
demo@mars:~$ who | cut -c 1-8 | sort | uniq -c &
[2] 18144

[1]+  Stopped                 cat
demo@mars:~$

[2]-  Done                    who | cut -c 1-8 | sort | uniq -c
```

后台的cat要从终端读，收到信号SIGTTIN而暂停运行

后台进程暂停、结束信息（[2]-Done…)缺省等待下一个shell提示时显示

作业控制

- 每个shell有不同的作业：编号从1开始
- 作业状态：
  - 前台运行: 正在运行的作业，输入连接到控制终端
  - 后台运行：[Running]，没有因为等待输入而停止
  - 暂停运行：后台暂时停止，等待SIGCONT信号恢复执行
- 查看作业状态： jobs [options] {-l 显示进程ID}
  - 当前shell的输入连接到控制终端，接收shell命令 jobs
  - + 表示当前作业，即最近切换到后台的那个作业
  - - 表示前一个作业
- Ctrl-Z/fg/bg命令: 暂停、切换前台和切换后台

命令	含义
echo $$	显示当前shell的PID
echo $!	显示最近切换到后台的进程PID
set -/+o monitor	开启或关闭作业控制
set -/+o notify	后台作业完成时是否立刻通知

```
demo@mars:~$ (sleep 180; echo 'done')
^Z
[3]+  Stopped                 ( sleep 180; echo 'done' )
demo@mars:~$ jobs
[1]   Stopped                 cat
[2]-  Stopped                 cat | sort
[3]+  Stopped                 ( sleep 180; echo 'done' )
demo@mars:~$ bg
[3]+ ( sleep 180; echo 'done' ) &
```

```
demo@mars:~$ jobs –l    # 查看进程ID
[1]- 18276 Stopped (tty input)     cat
[2]+ 18283 Stopped (tty input)     cat
     18284                         | sort
[3]  18378 Running                 ( sleep 180; echo 'done' ) &
```

Ctrl-Z暂停当前作业
bg: 当前作业切换到后台运行
fg: 当前作业切换到前台运行

作业控制: fg和bg

- 将当前作业或者指定的作业切换到前台

  fg [%job]

  fg %2(也可fg 2) 将作业号为2的作业切换到前台

  fg %make 将命令名前面为make的作业切换到前台

  fg %?game 将命令中包含game的作业切换到前台

- bg [%job]将当前作业或指定的作业切换到后台

命令	含义
jobs	列出当前的作业列表, -l 显示进程ID
ps	列出当前的进程列表
fg [%job]	将作业切换到前台
bg [%job]	将作业切换到后台
suspend	暂停当前shell，也可kill –STOP $$
^Z	发送信号TSTOP从而暂停当前前台作业
kill	发送信号到进程，缺省为终止(TERM)进程

```
demo@mars:~$ vi tty.txt
[4]+  Stopped                 vi tty.txt
demo@mars:~$ jobs
[1]   Stopped                 cat
[2]-  Stopped                 cat | sort
[3]   Running                 ( sleep 180; echo 'done' ) &
[4]+  Stopped                 vi tty.txt
demo@mars:~$ fg %2
cat | sort
^Z
[2]+  Stopped                 cat | sort
demo@mars:~$ fg %3
( sleep 180; echo 'done' )

[3]+  Stopped                 ( sleep 180; echo
'done' )
demo@mars:~$ bg %3
[3]+ ( sleep 180; echo 'done' ) &
demo@mars:~$ fg %vi
vi tty.txt

[4]+  Stopped                 vi tty.txt
demo@mars:~$ bg
[4]+ vi tty.txt &

[4]+  Stopped                 vi tty.txt
```

作业控制：nohup

- 控制终端断开(比如网络连接断开）或用户输入exit或logout退出shell时，会话的控制终端要退出
  - 发送SIGHUP给会话中的前台作业
  - 如果仍然有后台作业时会发送SIGHUP信号给所有的后台进程（一般收到SIGHUP信号时会终止）
- 守护进程(没有控制终端)在收到SIGHUP信号时一般重新读取配置文件
- shopt –s checkjobs打开checkjobs选项，输入exit或logout时不是马上退出，而是有后台作业时会提示有后台作业存在。继续退出时会终止(发送SIGHUP信号）那些与控制终端（输入或者输出）连接的所有作业

nohup COMMAND [ARG]

- nohup命令会执行COMMAND，如果该命令的标准输入为终端，则重定向到/dev/null，如果标准输出为终端，重定向到nohup.out，**忽略SIGHUP信号**
- 如果将nohup命令放到后台，在终端退出时也不会终止

  nohup wget url … &

作业控制：disown

- nohup cmd args... 重定向输入为/dev/null，输出为nohup.out，忽略SIGHUP信号，避免控制终端退出而结束命令的执行
- 如果一个命令已经开始执行了，可通过jobs找到对应的作业号，然后使用bash内置命令disown –h %job命令
  - 表示在控制进程退出时不给其发送SIGHUP信号
  - 注意该作业的标准输入和输出并没有改变

disown [-h] [-ar] [jobspec ... | pid ...]

将指定的作业从作业列表中移走，注意进程本身并不会被kill

-h 不是从作业列表中移走，而是标记不给该作业发送SIGHUP信号

-a 如果后面参数没有时表示所有作业

-r 如果后面参数没有时表示所有运行的作业

建议大家使用终端模拟器软件screen或tmux(更加流行），这样只要Linux系统不关机，再次登录时仍然可回到以前的环境

tmux初步使用

- sudo apt install tmux
- 创建一个会话(session)
  - tmux new –s name 创建一个名为name的会话，如果没有-s选项，则创建一个名为0的会话
- 加入一个会话
  - tmux ls列出当前的会话
  - tmux at –t name 加入名为name的会话，没有-t选项时加入最后一个会话
- 退出会话:
  - tmux detach 退出当前会话
  - Prefix + d 退出当前会话，缺省的prefix为Ctrl-b
  - Prefix + ? 可以列出可以使用的快捷键

结束会话:
tmux kill-session –t name

- 在会话中创建一个新的窗口： Prefix + c 或tmux new-window
- 在会话中切换窗口(window):
  - Prefix + 0/1/2，切换到对应编号的窗口
  - Prefix + n/p 切换到下一个或前一个窗口 Prefix + l 上一个活跃窗口
  - Prefix + w 列出当前窗口后选择切换到某个窗口
- 关闭窗口： 对应窗口的命令退出时自动关闭，也可以Prefix + &关闭当前窗口

tmux初步使用

- 一个窗口可以切割成多个pane
  - Prefix + % 当前的空间垂直分割成两个pane
  - Prefix + " 当前的空间水平分割成两个pane
- 切换pane
  - Prefix + o 切换下一个pane
  - Prefix + 箭头： 切换到指定方向的pane
- 最大化： Prefix + z 当前pane最大化，再次Prefix + z恢复原来的布局
- 关闭pane： Prefix + x 关闭当前pane

查看进程状态ps

Linux短选项：ps [-aefFly] [-t tty] [-p pid] [-u userid]
BSD短选项：ps [ajluvx] [t tty] [p pid] [U userid]

- 选项：BSD短选项(前无连字符) 、Linux短选项 (前有连字符) 和GNU长选项(前有两个连字符)

查看哪些进程？缺省为当前用户且和当前控制终端控制的进程

- 当前用户还是所有用户的进程？
- 由某个或者任意终端控制还是不要求由控制终端控制？

Linux选项	BSD选项	含义两种选项确定范围有时稍有区别
ps	ps	当前用户且当前控制终端控制的进程，缺省
ps –a	ps a	所有用户(a取消当前用户限制)且和某个终端相关的进程。-a选项不包括session leader进程
ps –e/-A	ps ax	所有进程(包括守护进程)。x取消终端限制, 包括守护进程
ps –p pidlist	ps p pidlist	与pidlist指定的进程ID相关的进程，以逗号分割
ps –u uidlist	ps U uidlist	与uidlist指定的用户相关的进程，以逗号分割
ps –t ttylist	ps t ttylist	与终端ttylist相关的进程
ps –w	ps w	宽输出(不会截取字符)
ps –o format	ps o format	用户自定义输出的格式，比如 o uid,pid,user,args
ps –f/-F/-l/-ly		full-format/extra full format/long format/不显示flags
ps j/l/u		job control/long format(多了CPU详细信息)/面向用户格式
ps e		显示用到的环境变量 e=environment
ps jf		查看进程树, j =job，f=forest
ps –C command		命令command相关进程

查看进程状态ps

- 查看当前用户和当前终端进程 ps
- **查看当前用户的所有进程ps x**
- 查看所有用户与终端相关的进程:
  - ps a、ps au 或者ps –af
- 查看守护进程（没有终端) ps –t -
- 查看所有进程：ps ax或者 ps –e
  - 长格式： ps axj、ps aux、ps auxw
    或者ps –ef、ps –eF等
- 查看进程树: ps axjf
- 查看某个服务相关的进程 ps -C sshd u

状态	含义
R	正在运行或等待运行
S	睡眠状态，可被唤醒；等待事件结束
T	停止状态，作业控制信号而挂起或traced
Z	僵尸进程，进程已结束但无父进程
D	不可唤醒的睡眠状态，等待事件结束（磁盘I/O)

```
$ ps auxw
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.1 225452  9132 ?        Ss   May02   0:02 /sbin/init splash
root         2  0.0  0.0      0     0 ?        S    May02   0:00 [kthreadd]
root         4  0.0  0.0      0     0 ?        I<   May02   0:00 [kworker/0:0H]
root         6  0.0  0.0      0     0 ?        I<   May02   0:00 [mm_percpu_wq]
root         7  0.0  0.0      0     0 ?        S    May02   0:00 [ksoftirqd/0]
root         8  0.0  0.0      0     0 ?        I    May02   0:34 [rcu_sched]
root         9  0.0  0.0      0     0 ?        I    May02   0:00 [rcu_bh]
```

监视系统进程top

top [-b] [-d delay] [-u user] [-n count] [-p pid[,pid]…]

实时查看系统中当前运行的进程的所有信息，每隔delay(-d secs）刷新状态信息，-n count指定刷新几次后退出，-p pid指出只查看哪些进程的状态，-b 表示批处理执行(可重定向到文件)，缺省屏幕状态执行

和less/vi类似，以原始模式运行，完全接管屏幕

- h或者?查看帮助
- q退出
- b:打开/关闭加亮效果(状态为运行R的行y或者某排序列x)
- x和y: 打开/关闭排序列/运行态进程高亮效果
- "shift + >"或"shift + <" 改变排序列
- f: 更改显示列
- 箭头和翻页键确定显示的内容
- u 查看哪些用户，s设置刷新间隔，n设置屏幕显示的行数
- k 发送信号给某个进程

```
top - 16:27:12 up 4 days,  7:57,  3 users,  load average: 0.00, 0.00, 0.00
Tasks: 215 total,   1 running, 210 sleeping,   4 stopped,   0 zombie
%Cpu(s):  0.0 us,  0.3 sy,  0.0 ni, 99.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  2048444 total,   250336 free,   910500 used,   887608 buff/cache
KiB Swap:  2095100 total,  2010200 free,    84900 used.   896968 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
 1324 root      20   0  364360   1648   1584 S  0.3  0.1   2:25.65 VBoxService
 2325 demo      20   0  232264    644    644 S  0.3  0.0  21:15.44 VBoxClient
19274 demo      20   0   97468   4184   3212 S  0.3  0.2   0:01.33 sshd
19936 demo      20   0  157860   3792   3236 R  0.3  0.2   0:00.37 top
    1 root      20   0  119776   5388   3516 S  0.0  0.3   0:17.80 systemd
    2 root      20   0       0      0      0 S  0.0  0.0   0:00.17 kthreadd
    3 root      20   0       0      0      0 S  0.0  0.0   0:37.50 ksoftirqd/0
    5 root       0 -20       0      0      0 S  0.0  0.0   0:00.00 kworker/0:0H
    7 root      20   0       0      0      0 S  0.0  0.0   0:56.02 rcu_sched
    8 root      20   0       0      0      0 S  0.0  0.0   0:00.00 rcu_bh
    9 root      rt   0       0      0      0 S  0.0  0.0   0:00.00 migration/0
   10 root      rt   0       0      0      0 S  0.0  0.0   0:05.08 watchdog/0
```

向进程发送信号

kill [-signal] pid… | jobid… 缺省SIGTERM信号
kill -l 列出支持的信号，不同系统支持信号以及编号可能有所不同

- 进程如何退出?
  - 正常退出
  - 在前台时可Ctrl-C发送中断信号(INT)结束进程或Ctrl-\发送QUIT信号强制终止进程
  - 控制终端退出时发送信号SIGHUP给所有前台和后台作业
- kill命令：发送信号给进程，可以通过进程ID来指出发信号给哪些进程，也可通过作业号（%n %command %?name)等指定发信号给哪个作业的进程
  - pid=0表示当前进程组内的所有进程，pid=-1（**谨慎使用!**）表示允许发送信号的所有进程(init除外)，pid<-1表示进程组为-pid的所有进程
- 信号0不实际发送信号，但是进行错误检查，是否存在进程等

man 7 signal查看信号

- 当前有效用户为超级用户;当前进程的有效用户ID或真实用户ID与接收信号的进程的真实用户ID相同时才允许发送信号
- 执行程序一般会定义自己的信号处理程序，但是KILL和STOP信号不能忽略

编号	名称	缩写	缺省含义
1	SIGHUP	HUP	控制终端退出，也常用于重启程序，重新读取配置文件
2	SIGINT	INT	终端中断信号，用户按了Ctrl-C
3	SIGQUIT	QUIT	终端退出信号，按Ctrl-\，保存core文件
9	SIGKILL	KILL	立即终止进程，不能忽略
15	SIGTERM	TERM	缺省终止信号，请求终止
20	SIGTSTP	TSTP	交互式停止信号，用户按Ctrl-Z
19	SIGSTOP	STOP	挂起进程，不能忽略
18	SIGCONT	CONT	恢复挂起进程

向进程发送信号killall

- killall与kill类似，只是通过名字来指定进程，如果名字为绝对路径名，则仅仅给该程序对应的进程发送信号

killall [options] [-signal] name

-g 对应进程所在进程组内的所有进程

-i 交互式模式

-r 正则表达式来匹配进程名称

-I 忽略大小写

-u user 只给用户user的进程发送信号

-w 发送信号后等待进程全部结束，每秒检查一次是否已全部退出

```
demo@mars:~$ ps
  PID TTY          TIME CMD
…
20742 pts/9    00:00:00 sort
demo@mars:~$ kill 20742
demo@mars:~$ ps
  PID TTY          TIME CMD
20742 pts/9    00:00:00 sort
demo@mars:~$ kill -9 20742

[2]+  Stopped                 cat | sort
demo@mars:~$ jobs -l
[4]  18707 Stopped (tty output)    vi
tty.txt
[5]  20737 Stopped (tty input)     cat
[6]- 20738 Stopped (tty input)     cat -
[7]+ 20741 Stopped (tty input)     cat
     20742 Killed                  | sort
demo@mars:~$ killall -9 cat
[5]   Killed                  cat
[6]-  Killed                  cat -
[7]+  Killed                  cat | sort
```

查找进程或者给进程发信号pgrep和pkill, pidof

pidof program 查看命令名为program的进程pid

pgrep [options] pattern　基于pattern(扩展正则表达式）查找匹配的活跃进程，返回进程的pid

pkill [options] pattern 与pgrep类似，只是给进程发送信号

-signal	pkill发送的信号
-u /-g	指定用户和用户组的进程
-d delimiter	pgrep返回的进程PID之间的分隔符，缺省为\n
-l /-a	除了pid外还包含进程名或者完整的命令行
-c	仅仅返回进程个数

```
demo@mars:~$ pgrep -lu demo sshd
1861 sshd
3440 sshd
demo@mars:~$ renice +4 $(pgrep -u demo firefox)
4514 (process ID) old priority 0, new priority 4
demo@mars:~$ pkill -SIGSTOP -u demo firefox
demo@mars:~$ pkill -SIGCONT -u demo firefox
demo@mars:~$  pidof sshd
29379 29272 3440 1861 1345
```

demo用户的sshd进程，输出还包括进程名

设置进程优先级nice

```
demo@mars:~/bin$ cat mysleep
sleep 45000
demo@mars:~/bin$ nice mysleep&
demo@mars:~/bin$ nice -n 4
mysleep&
```

- 调度器基于进程优先级为各个进程动态分配资源

nice [-n niceness] command

renice priority [-g|-p|-u] identifier...

nice表示以一个低优先级(-n选项指定NI值，缺省为10）来执行外部命令，这些命令是那些需要大量CPU且可在后台运行的命令，一般会在后面添加&以后台运行

- renice改变已经运行的进程的优先级， -p(缺省)表示后面参数为pid, -g表示后面为进程组ID, -u表示后面为用户名或用户ID。普通用户一般只能设置更低的优先级
- 进程的niceness值(NI)为[-20,19]，值越低，优先级越高。普通命令的NI为0

ps l可以查看NI值

```
demo@mars:~/bin$ ps axl | egrep 'UID|loop|sleep|demo'
F   UID   PID  PPID PRI  NI    VSZ   RSS WCHAN  STAT TTY        TIME COMMAND
4     0  1541   750  20   0  97464  5968 -      Ss   ?          0:00 sshd: demo [priv]
5  1000  1597  1541  20   0  97464  3320 -      S    ?          0:08 sshd: demo@notty
1  1000  2487  2270  20   0 282600   292 -      Ss   ?          0:15 gpg-agent --homedir /home/demo/.gnupg --use-s
tandard-socket --daemon
1     0  6105     2   0 -20      0     0 -      S<   ?          0:00 [loop0]
1     0 13352     2   0 -20      0     0 -      S<   ?          0:00 [loop1]
4     0 17325   750  20   0  97468  6964 -      Ss   ?          0:00 sshd: demo [priv]
5  1000 17435 17325  20   0  97468  4124 -      S    ?          0:03 sshd: demo@pts/9
4     0 19165   750  20   0  97468  6972 -      Ss   ?          0:00 sshd: demo [priv]
5  1000 19274 19165  20   0  97468  4184 -      S    ?          0:10 sshd: demo@pts/8
0  1000 20901 19275  30  10   4508   788 wait   SN   pts/8      0:00 /bin/sh /home/demo/bin/mysleep
0  1000 20902 20901  30  10 123352   648 hrtime SN   pts/8      0:00 sleep 45000
0  1000 20918 19275  24   4   4508   712 wait   SN   pts/8      0:00 /bin/sh /home/demo/bin/mysleep
0  1000 20919 20918  24   4 123352   652 hrtime SN   pts/8      0:00 sleep 45000
0  1000 20998 19275  20   0 130288  1032 pipe_w S+   pts/8      0:00 grep -E --color=auto UID|loop|sleep|demo
```

查找正在使用文件或socket的进程fuser

fuser [-umv] [-k[i] [-signal]] NAME

查找打开了名字为NAME的文件、目录或socket的进程，显示的结果为文件名: 进程的ID(后面可能还包括访问模式)

-u 输出中包括用户名

-v verbose模式，列出进程详细信息

-m 显示正使用文件或目录所在文件系统中的文件的进程

-k 给进程发信号，缺省SIGKILL， -i选项表示询问用户是否发信号

```
demo@mars:~/bin$ fuser .
/home/demo/bin:      19275c 20901c 20902c 20918c 20919c
demo@mars:~/bin$ fuser ~/.bashrc
/home/demo/.bashrc: 12292
```

查找正在使用文件或socket的进程fuser

- 在移走移动存储前，应该通过umount卸载文件系统，如果该文件系统中已经有文件打开时，无法卸载
- 可以通过-m选项找到有哪些进程在打开文件系统中的文件，当然也可合并使用-ki选项来结束那些进程

```
demo@mars:~/bin$ sudo umount /mnt
umount: /mnt: target is busy
        (In some cases useful info about processes that use the device is
found by lsof(8) or fuser(1).)
demo@mars:~$ fuser /mnt/tmp/1.txt
demo@mars:~$ fuser -m /mnt/tmp/1.txt
/mnt/tmp/1.txt:       21659c
demo@mars:~$ fuser -mv /mnt/tmp/1.txt
                      USER         PID ACCESS COMMAND
/mnt/tmp/1.txt:       root     kernel mount /mnt
                      demo       21659 ..c.. bash
demo@mars:~$ fuser -mki /mnt/tmp/1.txt
/mnt/tmp/1.txt:       21659c
Kill process 21659 ? (y/N) y
```

打开了1.txt所在文件系统(即/mnt)上的文件或目录的进程

查找正在使用文件或socket的进程fuser

fuser [-umv] [-k[i] [-signal]] NAME

NAME也可是socket，格式为port/space, space可取值tcp或者udp

选项-4 表示IPv4， -n tcp表示后面的NAME(ssh)属于TCP名字空间

fuser不会列出当前用户没有权限查看的那些文件的进程，sudo用于获得超级用户权限

```
demo@mars:~$ fuser ssh/tcp    # 也可 fuser 22/tcp
demo@mars:~$ sudo fuser ssh/tcp
ssh/tcp:              750  1541  1597 17325 17435 19165
19274 21184 21294
demo@mars:$ sudo fuser -v -4 -n tcp ssh
...
```

列出(进程)打开的文件lsof

lsof [options] [name]…

- lsof 缺省列出所有进程打开的文件，如果有参数name，则表示仅仅查看打开了name对应的文件的进程所打开的与name相关的文件
  - name为普通文件，显示哪些进程打开了该文件
  - **name为块设备文件或挂载点，显示哪些进程打开了设备上的文件**
- -c command 列出执行的程序以command开头的进程打开的文件，可多个-c选项，之间为OR关系
- -u [^]user 列出这个(些)用户或(前面有^，则为)这个(些)用户以外的进程打开的文件
- -g [^]group列出这个(些)用户组或者这个(些)用户组以外(前面有^)的进程打开的文件
- -p [^]pid 列出这个(些)进程或者这个(些)进程以外(前面有^)的进程打开的文件
- -a 表示后面的选择选项之间为AND关系，缺省为OR关系
- +d dir 只包含该目录以及该目录第一层的文件或目录
- +D dir 包含该目录以及(递归包括)该目录子目录下的文件

列出打开的文件lsof

FD： 文件描述字或cwd当前工作目录、rtd(根目录)、txt（程序）等

```
demo@mars:~$ sudo lsof | more
[sudo] password for demo:
lsof: WARNING: can't stat() fuse.gvfsd-fuse file system /run/user/1000/gvfs
      Output information may be incomplete.
COMMAND     PID  TID       USER   FD      TYPE             DEVICE  SIZE/OFF       NODE NAME
systemd       1            root  cwd       DIR               8,17      4096          2 /
systemd       1            root  rtd       DIR               8,17      4096          2 /
systemd       1            root  txt       REG               8,17   1577232     468770 /lib/systemd/systemd
systemd       1            root  mem       REG               8,17     18976     398901 /lib/x86_64-linux-gnu/libuu
id.so.1.3.0
systemd       1            root  mem       REG               8,17    262408     398709 /lib/x86_64-linux-gnu/libbl
kid.so.1.1.0
systemd       1            root  mem       REG               8,17     14608     398741 /lib/x86_64-linux-gnu/libdl
-2.23.so
```

- 查看谁在使用文件/bin/bash

```
demo@mars:~$ lsof /bin/bash
COMMAND   PID USER  FD   TYPE DEVICE SIZE/OFF   NODE NAME
bash    19275 demo txt    REG   8,17  1037464 917513 /bin/bash
bash    21295 demo txt    REG   8,17  1037464 917513 /bin/bash
```

- lsof +d ~ 查看用户主目录以及第一层文件的使用情况
- lsof +D ~ 查看用户主目录以及各级子目录下文件的使用情况：
- lsof –u demo 查看用户demo打开的文件
- lsof –u ^root 查看非root用户打开的文件
- lsof -u demo -a -c ssh -c bash 列出用户为demo且运行的命令以ssh或者bash开头的进程打开的文件
- lsof -p 19275,20918 列出进程19275和20918打开的文件

列出打开的文件lsof： 网络相关

- -i [46][tcp|udp][@hostname|hostaddr][:service|port] 列出与指定的socket地址匹配的文件。比如TCP:25 @1.2.3.4 UDP:dns等
- -n 不要将IP地址转变为主机名，加快速度

```
demo@mars:~$ sudo lsof -i
COMMAND     PID   USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
avahi-dae   682  avahi   12u  IPv4  13061      0t0  UDP *:mdns
avahi-dae   682  avahi   13u  IPv6  13062      0t0  UDP *:mdns
avahi-dae   682  avahi   14u  IPv4  13063      0t0  UDP *:49697
avahi-dae   682  avahi   15u  IPv6  13064      0t0  UDP *:34416
sshd        750   root    3u  IPv4 522147      0t0  TCP *:ssh (LISTEN)
sshd        750   root    4u  IPv6 522149      0t0  TCP *:ssh (LISTEN)
cups-brow   751   root    8u  IPv4  14000      0t0  UDP *:ipp
dnsmasq     858 nobody    4u  IPv4  14847      0t0  UDP mars:domain
dnsmasq     858 nobody    5u  IPv4  14848      0t0  TCP mars:domain (LISTEN)
xrdp       1506   xrdp    6u  IPv4  17952      0t0  TCP *:3389 (LISTEN)
xrdp-sesm  1508   root    6u  IPv4  17683      0t0  TCP localhost:3350 (LISTEN)
sshd       1541   root    3u  IPv4  18014      0t0  TCP 10.0.4.15:ssh->10.0.4.2:51963 (ESTABLISHED)
sshd       1597   demo    3u  IPv4  18014      0t0  TCP 10.0.4.15:ssh->10.0.4.2:51963 (ESTABLISHED)
vino-serv  2776   demo   13u  IPv6  28307      0t0  TCP *:5900 (LISTEN)
vino-serv  2776   demo   14u  IPv4  28308      0t0  TCP *:5900 (LISTEN)
dhclient  13977   root    6u  IPv4 518969      0t0  UDP *:bootpc
dhclient  14202   root    6u  IPv4 519852      0t0  UDP *:bootpc
dhclient  14662   root    6u  IPv4 521635      0t0  UDP *:bootpc
```

sudo lsof -n -i4tcp:ssh
表示仅仅列出IPv4中TCP连接中有端口为ssh(22)号的TCP socket

定期执行命令crontab

```
$ pgrep -la cron
836 /usr/sbin/cron -f
```

- cron程序允许用户设置在某些时刻执行某些命令
  - cron程序每分钟检查相应的crontab设置，看是否要执行相应的命令
  - /etc/crontab.allow和/etc/crontab.deny文件用于控制谁可以执行crontab
- 如何进行crontab设置？
  - 超级用户的crontab设置在/etc/crontab文件中
  - 普通用户需要通过crontab命令来设置自己的crontab文件

crontab [-u user] file　更新用户(缺省当前用户)的crontab设置，如果没有file，则从标准输入读取

crontab [-u user] [-i] { -e | -l | -r }　-e表示编辑，-l表示列出，-r表示移走，-i表示移走时询问

```
demo@mars:~$ crontab –l  # 列出crontab设置
no crontab for demo
demo@mars:~$ cat crontab.demo
*/5 * * * * echo "`date`" >> /tmp/crontab.txt
demo@mars:~$ crontab crontab.demo  # 更新crontab设置
demo@mars:~$ crontab -l
*/5 * * * * echo "`date` >> /tmp/crontab.txt
demo@mars:~$ crontab –r    # 删除crontab设置
```

crontab文件

run-parts 执行目录中的脚本
test –x /usr/sbin/anacron || (cd / && run-parts…)
首先检查anacron是否可执行，如果不可执行时手写切换到目录/，然后执行run-parts

- 包括环境设置和cron命令
- #为注释，但注意不要出现在环境设置和cron命令中
- cron服务会设置缺省环境变量PATH、SHELL(/bin/sh)、HOME、LOGNAME
- crontab文件允许修改环境变量设置，但注意不允许参数展开(如$PATH等)，MAILTO环境变量为执行有问题时发送的通知邮件的接收者
- 当前面的分钟、小时、月份都匹配，且**天或者星期几之一**满足时执行后面的命令
- 字段描述可采用：
  - range：比如小时7-9，表示小时为7、8、9
  - *: 表示合法的range范围，比如分钟的*相当于[0-59]

/etc/crontab的内容

```
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
# minute(0-59) hour(0-23) day-of-month(1-31) month-of-year(1-12,名字) day-of-week(0-7,名字) command
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
```

crontab文件

- 当前面的分钟、小时、月份都匹配，且天或者星期几之一满足时执行后面的命令
- 命令如果不是shell内置命令，应该采用绝对路径
- 字段描述支持：
  - list：比如小时的 2,4,7-9，表示小时为2、4、7、8、9
  - range：比如小时7-9，表示小时为7、8、9
  - *：表示合法的range范围，比如分钟的*相当于[0-59]
  - step可以与range结合，格式为range/n。比如 4-14/3表示4,7,10,13。

```
# 每天12:05分执行
5 0 * * *       $HOME/bin/daily.job >> $HOME/tmp/out 2>&1
# 每月第一天14:15分执行
15 14 1 * *     $HOME/bin/monthly
# 工作日晚上10点执行
0 22 * * 1-5   echo "it's late... `date`" >>$HOME/tmp/out 2>&1
23 0-23/2 * * * echo "run 23 minutes after midn, 2am, 4am ...,
everyday"
5 4 * * sun     echo "run at 5 after 4 every sunday"
# Run on every second Saturday of the month
0 4 8-14 * *    test $(date +%u) -eq 6 && echo "2nd Saturday"
```

异步周期执行： anacron

- cron针对的是机器一直运行的情况，如果定期执行任务的时刻到来时，机器已经关机了，启动后并不会执行之前的任务
- anacron是cron的一个补充，允许以天数或者月份为单位定期执行作业
  - 需要手动地执行anacron命令，当然管理员可设置为启动时运行anacron，以及定期运行anacron
  - 检查作业列表，上次运行该作业到当前时刻是否超过了作业的指定间隔，如果是的话就运行该作业
- 配置文件在 /etc/anacrontab
  - period delay job-identifier command 表示给要执行的命令command一个标识job-identifier(邮件中会列出该名字)，该作业的间隔为period天，delay表示启动anacron之后等待delay分钟执行作业
  - @monthly delay job-identify command 表示间隔为一个月（会考虑到不同月份天数变动的情况）
  - 作业执行完成之后会在相应目录(缺省/var/spool/anacron) 中通过一个文件(job-identifier)纪录一个时间戳(即作业完成的日期)

```
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
HOME=/root
LOGNAME=root
1       5       cron.daily      run-parts --report /etc/cron.daily
7       10      cron.weekly     run-parts --report /etc/cron.weekly
@monthly        15      cron.monthly    run-parts --report /etc/cron.monthly
```

普通用户可通过-t选项指定anacrontab，-S选项指定时间戳保存目录来执行自己的anacron作业：

```
anacron -t ~/.anacrontab -S ~/.anacron/
```