link: null
title: 珠峰架构师成长计划
description: sleep.sh
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=124 sentences=76, words=1140

1. 进程管理

- 进程是正在执行的一个程序或命令，每一个进程都是一个运行的实体，都有自己的地址空间并且用一定的系统资源
- 进程就是正在执行的某个程序
- 判断服务器的状态
- 查看系统中的所有进程
- 杀死进程，只有无法关闭才要杀死进程
- `ps aux` 查看系统中所有进程，使用BSD操作系统格式
- `ps -le` 查看系统中所有进程，使用Linux标准格式
- TTY是TeleType的一个缩写，原来指的是电传打字机，是通过串行线用打印机键盘通过阅读和发送信息的东西
- pts(pseudo['su:dou]-terminal slave)是所谓的伪终端或虚拟终端

参数 含义 -a 显示一个终端的所有进程 -u 显示进程的归属用户及内存的使用情况 -x 显示没有控制终端的进程 -l 长格式显示，显示更详细的信息

数据 含义 USER 该进程是由哪个用户创建的 PID 进程的ID号 %CPU 该进程占用CPU资源的百分比，占用越高说明越消耗系统资源 %MEM 该进程占用物理内存的百分比，占用越高说明越消耗系统资源 VSZ 该进程占用虚拟内存的百分比，单位是KB RSS 该进程占用实际物理内存大小，单位是KB TTY 该进程在哪个终端中运行。tty1~tty7表示本地控制终端，tty1~tty6是字符终端，tty7是图形终端。pts/0~255代表虚拟终端，?表示此终端是系统启动的 STAT 进程状态 START 该进程的启动时间 TIME 该进程占用CPU的运算时间,数值越高说明越消耗系统资源 COMMAND 产生此进程的命令名

参数 含义 R(Runing) 运行 S(Sleep) 休眠 T(Terminated) 停止 S(Son) 包含子进程 + 位于后台

- pstree [选项]
    - -p 显示进程PID
    - -u 显示进程的所属用户
- top

```
top -b -n 1 > top.txt
```

选项 含义 -b 使用批处理模式输出，一般和

配合使用 -n 次数，指定top命令执行的次数。一般了

选项配合使用 -d 秒数，指定top命令每隔几秒更新。默认是3秒

选项 含义 ?或h 显示交互模式的帮助 P 按CPU使用率排序，默认就是此选项 M 以内存的使用率排序 N 以PID排序 q 退出top

内容 说明 12:12:12 系统的当前时间 up 1 day 5:33 系统的运行时间，本机已经运行了1天5小时33分 2 users 当前登录了二个客户端 load average 0 0 0 系统在之前1分钟、5分钟、15分钟的平均负载。一般认为小于1小时负载较小，大于1超过负载

内容 说明 Tasks: 100 total 系统中的进程总数 1 running 正在运行的进程数 94 sleeping 睡眠的进程 0 stopped 正在停止的进程 0 zombie 僵尸进程。如果不是0的话要进行检查

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 空闲CPU的CPU百分比 0.1%wa 等待输入/输出的进程的占用CPU百分比 0.1%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)虚拟时间百分比，就是当有

- 硬中断 由与系统相连的外设(比如网卡、硬盘)自动产生的。主要是用来通知操作系统系统外设状态的变化。比如当网卡收到数据包的时候，就会发出一个中断。我们通常所说的中断指的是硬中断(hardirq)。
- 软中断是通讯进程之间用来模拟硬中断的 一种信号通讯方式,软中断是执行中断指令产生的，无面外部施加中断请求信号，因此中断的发生不是随机的而是由程序安排好的。
- 硬中断是可屏蔽的，软中断不可屏蔽。软中断是由程序调用发生的,而硬中断是由外设引发的

内容 说明 Mem: 1030720k total 物理内存的问题，单位是KB 551860k used 已经使用的物理内存数量 478860k free 空闲的物理内存数量，虚拟机分配了1024M内存，使用了538M,空闲467M 43180k buffers 作为缓冲的内存数量，可以存放需要写入硬盘的数据，用来加速数据的写入

内容 说明 Swap: 2047992k total 总计的交换分区(虚拟内存)大小 536k used 已经使用的交换分区大小 2047456k free 空闲的交换分区大小 368164k cached 把需要经常读取的数据从硬盘读到内存中，加速了数据的读取

- httpd采用的是worker模式，是一种多进程与多线程混合的模式
- `kill -l` 查看可用的进程信号

信号 代码 信号名称 说明 示例 1 SIGHUP 该信号让进程立即关闭，然后重写读取配置文件后重启,平滑重启 kill -1 -HUP 进程号 2 SIGINT 程序终止信号，用于关闭前台进程,相当于ctrl+c 9 SIGKILL 用来立刻结束程序的运行，本信号不能阻塞、处理和忽略，一般用于强制中止 15 SIGTERM 正常结束进程的信号，kill命令的默认信号。如果不能正常中止，才会尝试SIGKILL信号

```
kill -9 进程号
```

- 按照进程名杀死进程
- killall [选项][信号] 进程名
    - -i 交互式，询问是否要杀死某个进程
    - -I 进程名忽略大小写
- /etc/httpd/conf/httpd.conf
    - ServerName localhost:80
- 按照进程名杀死进程
- pkill [选项][信号] 进程名 -t 按终端号踢出用户

```
pkill -9 -t "pts/2"
```

选项 含义 USER 显示登陆用户帐号名 TTY 用户登陆所用的终端 FROM 显示用户在何处登陆系统 LOGIN@ 是LOGIN AT的意思，表示登陆进入系统的时间 IDLE 用户空闲时间，从用户上一次任务结束后，开始记时 JCPU 终端代号来区分，表示在这段时间内，所有与该终端相关的进程任务所耗费的CPU时间 PCPU 指WHAT域的任务执行后耗费的CPU时间 WHAT 表示当前执行的任务

- Linux操作系统是一个多用户、多任务的操作系统，Linux系统同时管理着非常多的进程，但是CPU在同一个时间周期内只能运算一个指令
- 进程的优先级决定了每个进程处理的先后顺序
- `ps -le` 可以查看进程优先级
- PRI表示Priority,NI表示Nice。这两个值都是优先级，数字越小代理进程优先有越高
- NI的值范围是-20~19
- 普通用户调整NI值的范围是0~19,而且只能调整自己的进程
- 普通用户只能调高NI值，但不能调低。比如原来是0，则只能调为大于0的数字

- root用户才能设定进程NI值为负值，而且可以调整任何用户的进程
- PRI(最终值)=PRI(原始值)+NI
- 用户只能修改NI的值，不能直接修改PRI的值
- nice命令可以给新执行的命令直接赋NI值，但不能修改已经存在进程的NI值
- nice [选项] 命令
- 选项 -n NI值 给命令赋新的NI值

```
nice -n -5 service httpd start
```

- 修改已经存在的进程的NI的值
- renice [优先级] PID

```
renice -10 30054
0054: old priority -5, new priority -10
# ps -le | grep httpd
1 S     0 30054     1  0  70 -10 -  2792 -      ?        00:00:00 httpd
5 S    48 30055 30054  0  75  -5 -  2792 -      ?        00:00:00 httpd
```

2. 工作管理

- 工作管理就是指的是单个登录终端中同时管理多个工作的行为
- 有时候有些命令会卡住我们的操作界面，我们就需要把它放入后台，比如拷贝大型文件
- 当前的登录终端只能管理当前终端的工作，而不能管理其它终端工作
- 放入后台的命令必须是还要持续一段时间的，这样我们才能去捕捉和操作这个动作
- 放入后台的命令不能和前台用户有交互或者需要前台输入，否则放入后台只能暂停，而不会执行
- & 在命令后面加可以把命令放入后台，并在后台执行
- ctrl+z 把工作放在后台暂停
- jobs
    - -l 显示工作的PID
- 加号代表最近一个放入后台的工作，也就工作恢复时默认恢复的工作
- 减号代表倒数第二个放入后台的工作

```
[1]+  Stopped                 top
# jobs -l
[1]+ 30562 停止 (信号)          top

# sleep 100s
^Z
[2]+  Stopped                 sleep 100s
# jobs -l
[1]- 30562 停止 (信号)          top
[2]+ 30588 停止                  sleep 100s

vim hello.txt &

find / -name hello
```

- fg %工作号
    - -%工作号 %可以省略，注意工作号和PID是不同的
- bg %工作号
    - -%工作号 %可以省略，注意工作号和PID是不同的
- 后台恢复执行的命令，不能和前台有交互

```
# bg 2
[2]+ sleep 100s &
# jobs -l
[1]+ 30562 停止 (信号)          top
[2]- 30601 Running                 sleep 100s &
```

- 所有的后台程序默认跟终端绑定，终端消失后台程序也会退出
- 当终端退出的时候，系统会向终端里所有的进程发送一个 SIGHUB的信号,终止后台进程
- 把需要后台执行的命令加入到 /etc/rc.local文件中
- 使用系统定时任务，让系统在指定的时间执行某个后台命令
- 使用 nohub命令
- nohup [命令] &

sleep.sh

```
for((i=0;i<10000;i++))
do
   echo `date` >> /root/date.log
   sleep 1s
done
```

```
nohup ./sleep.sh &
tail -f /root/date.log
ps -ef | grep sleep.sh
```

3. 系统资源查看

- 监控系统资源使用状态
- vmstat [刷新延时 刷新次数]

```
vmstat 1 3
procs -----------memory---------- ---swap-- -----io---- --system-- -----cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 1  0    532 329932  99388 459768    0    0    16    81   59   50  3  1 96  0  0
```

分类 参数 含义 procs r 等待运行的进程数，数量越大，系统就越繁忙 procs b 不可被唤醒的进程数量，数量越大，系统越繁忙

分类 参数 含义 memory swpd 使用的Swap空间的大小，单位KB memory free 空闲的内存容量，单位KB memory buff 缓冲的内存容量，单位KB memory cache 缓存的内存容量，单位KB

- 如果说si和so数越大说明数据经常要在磁盘和内存之间数据交换，系统性能就会越差

分类 参数 含义 swap si(in) 从磁盘中交换到内存中的数据的数量，单位KB swap so(out) 从内存中交换到硬盘中的数据的数量，单位KB

- bi和bo数越大，说明磁盘的I/O越繁忙

分类 参数 含义 io bi(in) 从块设备读入数据的问题，单位是块 io bo(out) 写到块设备的数据的总量，单位是块

- in和cs数越大，说明系统与接口设备的通信越繁忙

分类 参数 含义 system in(interrupt) 每秒被中断的进程次数 system cs(switch) 每秒钟进行的事件切换次数

分类 参数 含义 CPU us(user) 非内核进程消耗CPU运算时间的百分比 CPU sy(system) 内核进程消耗CPU运算时间的百分比 CPU id(idea) 空闲CPU的百分比 CPU wa(wait) 等待I/O所消耗的CPU百分比 CPU st(steal) 被虚拟机偷走的CPU百分比

- 开机时内核检测信息

```
dmesg | grep CPU
```

- 查看内存使用状态
- free [-b|-k|-m|-g]
- 选项
    - -b 以字节为单位
    - -k 以KB字节为单位
    - -m 以MB字节为单位
    - -g 以GB字节为单位

```
# free -m
             total       used       free     shared    buffers     cached
Mem:          1006        687        319          0         98        449
-/+ buffers/cache:        139        866
Swap:         1999          0       1999
```

分类 参数 含义 total 内存总数 used 已经使用的内存数 free 空闲的内存数 shared 多个进程共享的内存数 buffers 缓冲区内存数 cached 缓存内存数

参数 算法 含义 - buffers/cache 第一行的used-buffers-cached 已经使用的要减去缓存和缓冲的内存量 + buffers/cache 第一行的free+buffers+cached 空闲的要加上缓存和缓冲的内存量

分类 参数 含义 total swap总数，默认单位是K used 已经使用的swap数，默认单位是K free 空闲的swap数，默认单位是K

- 查看CPU的信息
- `cat /proc/cpuinfo`
- 显示系统的启动时间和平均负载，也就是top的第一行
- 通过 w也可以看到
- uname

```
# uname -a
Linux localhost 2.6.32-279.el6.i686 #1 SMP Fri Jun 22 10:59:55 UTC 2012 i686 i686 i386 GNU/Linux
# uname -s
Linux
# uname -r
2.6.32-279.el6.i686
```

```
file /bin/ls
```

```
lsb_release -a
```

```
yum install redhat-lsb -y
lsb_release -v
```

- lsof [选项]
- 列出进程调用或打开的文件的信息
- 选项
    - -c 字符串: 只列出字符串开头的进程打开的文件
    - -u 用户名：只列出某个用户的进程打开的文件
    - -p pid: 列出某个PID进程打开的文件

```
lsof | more
```

```
lsof /sbin/init
```

```
lsof -c httpd
```

```
lsof -u root
```

4. 系统定时任务

- 有些任务比如备份数据库等操作需要在系统空闲的时候执行
- 一次性定时任务**4.1.1 启动服务**

```
chkconfig --list | grep atd
service atd status
service atd start
```

- 如果系统中有 `/etc/at.allow`文件，那么只有写入 `/etc/at.allow`(白名单)中的用户可以使用 `at`命令(优先级更高，会忽略 `/etc/deny`文件)
- 如果系统中没有 `/etc/at.allow`文件，只有 `/etc/at.deny`文件，那么写入 `/etc/at.deny`文件中的用户不能使用at命令，但这个对 `root`用户并没有作用
- 如果这两个文件都不存在，那么只有 `root`用户可以使用 `at`命令
- at 选项 时间
- 选项
    - -m 当at工作完成后，无论是否命令有输出，都用 `email`通知执行 `at`命令的用户
    - -c 工作号 显示该at工作的实际内容
- 时间
    - HH:MM 例如 10:10
    - HH:MM YYYY -MM-DD 10:10 2018-08-08
    - HH:MM[am|pm] [month] [date] 10:10 May 11
    - HH:MM[am|pm] + [minutes|hours|days|weeks] now + 5 minutes

输出日志

```
# at now +5 minutes
at> /root/hello.sh > /root/hello.log
job 3 at 2019-04-10 22:48
[root@localhost ~]# atq
3    2019-04-10 22:48 a root

at -c 3
```

指定的时间重启服务器

- 查询当前服务器上的 at工作
- 删除指定的 at任务

```
service crond restart
chkconfig crond on
```

- 如果系统中有 /etc/cron.allow文件，那么只有写入 /etc/cron.allow(白名单)中的用户可以使用 at命令(优先级更高，会忽略 /etc/cron.deny文件)
- 如果系统中没有 /etc/cron.allow文件，只有 /etc/cron.deny文件，那么写入 /etc/cron.deny文件中的用户不能使用at命令，但这个对 root用户并没有作用
- 如果这两个文件都不存在，那么只有 root用户可以使用 at命令
- crontab [选项]
- 选项
    - -e 编辑crontab定时任务
    - -l 查询crontab任务
    - -r 删除当前用户所有的crontab任务

```
* * * * *  执行的任务
```

```
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...

# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name command to be executed
```

项目 含义 范围 第1个星 1个小时中的第几分钟 0~59 第2个星 1天当中的第几小时 0~23 第3个星 1月当中的第几天 1~31 第4个星 1年当中的第几月 1~12 第5个星 1周当中的星期几 0~6

符号 含义 例子 * 代表任意时间 比如第一个星就代表一个小时中每分钟都执行一次 , 代表不连续的时间 比如"1,2,3

",就代表每小时的1分、2分、3分执行命令 - 代表连续的时间范围 比如 " 1-5

- ** ",代表每小时的第1分到第5分执行命令 */n 代表每隔多久执行一次 比如 "

" 就代表每隔10分钟就执行一次命令 0 0 1,10 * 1 每月1号和10号，每周1的0点0分都会执行

符号 含义 10 22

- 在每天的22点10分执行 0 15

1 每周1的15点0分执行 0 5 5,10

每月5号和10号的凌晨5点整执行 10 5

1-5 每周一到周五的凌晨5点10分执行命令

每天凌晨10点钟，每隔10分钟执行一次

- 所有选项不能为空，必须填写
- crontab最小单位是分钟,最大单位是天
- 不管写命令还是脚本都要使用绝对路径
- crontab -e是用户执行的命令，不同的用户身份可以执行自己的定时任务
- 如果需要系统执行定时任务，可以编辑 /etc/crontab文件
- /etc/crontab可以指定shell、路径、邮件发送和家目录
- 修改 /etc/crontab配置文件

```
`5 5 * * * echo `date` >> /root/date.log`
```

- 把需要定时执行的脚本复制到/etc/cron.{daily,weekly,monthly}目录中的任意一个

```
root@localhost log]# ls /etc/cron*
/etc/cron.daily:
cups  logrotate  makewhatis.cron  mlocate.cron  prelink  readahead.cron  tmpwatch
/etc/cron.hourly:
/etc/cron.monthly:
/etc/cron.weekly:
```

- anacron是用来保证在系统关机的时候错过的定时任务(/etc/cron.daily)，可以在系统开机后自动执行
- anacron会使用1天、7天和1个月作为检测周期
- 在系统的 /var/spool/anacron目录中存在 cron.{daily,weekly,monthly}文件，用于记录上次执行cron的时间
- 和当前的时间做比较，如果两个时间的差值超过了 anacron指定的时间差，那就证明有 crontab未执行
- /etc/cron.{daily,weekly,monthly}只会被anacron调用

```
# /etc/anacrontab: configuration file for anacron

# See anacron(8) and anacrontab(5) for details.

SHELL=/bin/sh
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
# the maximal random delay added to the base delay of the jobs
# 最大随机延迟时间(分钟)
RANDOM_DELAY=45
# the jobs will be started during the following hours only
# anacron的执行时间范围是 3:00 ~ 22:00
START_HOURS_RANGE=3-22

#period in days   delay in minutes   job-identifier   command
每隔几天执行 强制延迟时间(分钟) 修改优先级 执行目录下面的所有脚本
1    5    cron.daily        nice run-parts /etc/cron.daily
7    25    cron.weekly        nice run-parts /etc/cron.weekly
@monthly 45    cron.monthly        nice run-parts /etc/cron.monthly
```

- 首先读取/var/spool/anacron/cron.daily中的上一次执行时间
- 和当前时间对比，如果两个时间相差超过1天，说明漏执行了，就执行cron.daily任务
- 执行任务的时间只能在3-22点之间
- 执行的时强制延迟时间为5分钟，再随机延迟0~45分钟
- 使用nice命令指定默认的优先级，再使用 run-parts脚本执行 /etc/cron.daily目录中所有的可执行文件