## **1. 进程管理**

### **1.1 进程**

* 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

### **1.2 进程管理简介**

* 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

### **1.3 进程的查看命令(ps)**

* 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:doʊ]-terminal slave)是所谓的伪终端或虚拟终端

#### **1.3.1 选项**

| **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 |

#### **1.3.2 结果字段含义**

| **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 |

#### **1.3.3 进程状态(STAT)**

| **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 |

### **1.4 pstree**

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

### **1.5 进程的查看(top)**

* top

top -b -n 1 > top.txt

#### **1.5.1 选项**

| **选项** | **含义** |
| --- | --- |
| -b | 使用批处理模式输出，一般和-n配合使用 |
| -n | 次数，指定top命令执行的次数。一般了-b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 |

#### **1.5.2 交互模式的命令**

| **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 |

#### **1.5.3 状态栏**

##### **1.5.3.1 第一行为任务队列信息**

| **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 |

##### **1.5.3.2 第二行为进程信息**

| **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 |

##### **1.5.3.3 第三行为CPU信息**

| **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 |

##### **1.5.3.4 第四行为物理内存信息**

| **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 |

##### **1.5.3.5 第四行为交换分区信息**

| **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 |

### **1.6 杀死进程(kill)**

#### **1.6.1 进程信号**

* 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 |  |

#### **1.6.2 杀死进程**

sleep.sh

#!/bin/bash

i=0**while** [ $i -le 1000 ]

**do**

echo $(date)

sleep 1s

done

sh sleep.sh &

ps -ef | grep sleep.sh

kill 进程号

## **2. 系统资源查看**

### **2.1 vmstat**

* 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

#### **2.1.1 (procs)进程信息字段**

| **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 |

#### **2.1.2 memory(内存信息字段)**

| **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 |

#### **2.1.3 swap(交换分区信息)**

* 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 |

#### **2.1.4 io(磁盘读写)**

* 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 |

#### **2.1.5 system(系统信息字段)**

* 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 |

#### **2.1.6 CPU(CPU信息字段)**

| **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 |

### **2.3 free**

* 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

#### **2.3.1 第一行**

| **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 |  |

#### **2.3.2 第二行**

| **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 |

#### **2.3.3 第三行**

| **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 |  |

### **2.6 查看内核相关信息**

* 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

### **2.7 查看操作系统位数**

file /bin/ls

### **2.8 查看发行版本**

lsb\_release -a

### **2.9 查看发行版本**

yum install redhat-lsb -y

lsb\_release -v

## **3. 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

### **3.2 crontab**

* 可以循环定时执行定时任务
* [cron](http://www.bejson.com/othertools/cron/)

systemctl restart crond.service

#### **3.2.3 crontab设置**

* 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

##### **3.2.3.1 语法**

# .---------------- 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

##### **3.2.3.2 位置**

| **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 |

##### **3.2.3.3 特殊符号**

| **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 "*/10*" 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 |  |

##### **3.2.3.4 案例**

| **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 |
| */10 10* | 每天凌晨10点钟，每隔10分钟执行一次 |

##### **3.2.3.5 注意事项**

* 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

### **3.3 系统定时任务**

* 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

### **3.3.1 /etc/crontab**

* 修改/etc/crontab配置文件

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

## **4. 实战任务**

### **4.1 监控nginx**

nginx.sh

#!/bin/bash

local nginx

nginx=`ps -ef |grep nginx|grep -v grep|wc -l`

**if** [ $nginx -gt 2 ];then

echo "your nginx is running"

exit 0

**else**

/bin/systemctl start nginx.service

exit 1

fi

### **4.2 监控mysql状态**

mysql.sh

#!/bin/sh

PortNum=`netstat -lnt|grep 3306|wc -l`**if** [ $PortNum -eq 1 ]

then

echo "mysqld is running."**else**

echo "mysqld is stoped."

fi

### **4.3 mysql备份脚本**

mysql\_backup.sh

#!/bin/bash

DATE=$(date +%F\_%H-%M-%S)

HOST=127.0.0.1

DB=test

USER=root

PASS=abcd1#EFG

MAIL="83687401@qq.com"

BACKUP\_DIR=/data/db\_backup

if [ ! -d "$BACKUP\_DIR" ];then

mkdir -p $BACKUP\_DIR

fi

SQL\_FILE=${DB}\_FULL\_$DATE.sql

BAK\_FILE=${DB}\_FULL\_$DATE.zip

cd $BACKUP\_DIR

if mysqldump -h$HOST -u$USER -p$PASS -B $DB > $SQL\_FILE; then

zip $BAK\_FILE $SQL\_FILE && rm -rf $SQL\_FILE

if [ ! -s $BAK\_FILE ]; then

echo "$DATE 备份失败" | mail -s "备份失败" $MAIL

fi

else

echo "$DATE 备份失败" | mail -s "备份失败" $MAIL

fi

find $BACKUP\_DIR -name '\*.zip' -ctime +14 -exec rm {} \;