Jar uf a.jar com/xx/a.class更新class

**Jstat -gc pid 3000 10//每3s统计一下gc，共十次**

-class：统计class loader行为信息

-compile：统计编译行为信息

-gc：统计jdk gc时heap信息

-gccapacity：统计不同的generations（不知道怎么翻译好，包括新生区，老年区，permanent区）相应的heap容量情况

-gccause：统计gc的情况，（同-gcutil）和引起gc的事件

-gcnew：统计gc时，新生代的情况

-gcnewcapacity：统计gc时，新生代heap容量

-gcold：统计gc时，老年区的情况

-gcoldcapacity：统计gc时，老年区heap容量

-gcpermcapacity：统计gc时，permanent区heap容量

-gcutil：统计gc时，heap情况

-printcompilation：不知道干什么的，一直没用过。

**jmap -histo:live 2364|sort -k 2 -n -r|less**

Jmap 观察内存和对象

参数如下：

-heap：打印jvm heap的情况

-histo：打印jvm heap的直方图。其输出信息包括类名，对象数量，对象占用大小。

-histo：live ：同上，但是只答应存活对象的情况

-permstat：打印permanent generation heap情况

jstack打印栈信息

**jstack -l 4089 > jstack.txt**

**Jinfo pid**

观察运行中的java程序的运行环境参数：参数包括Java System属性和JVM命令行参数

jconsole分析线程、堆栈内存信息

---------------------

Oracle连接会使用随机数加密token，Linux随机数和熵池有关系，下面的命令加大熵池

for i in {1..100000};do cat /proc/sys/kernel/random/entropy\_avail;done;

解决方案：

安装一个为提供提供熵的程序包

sudo apt-get install haveged

step1. 加入epel源(实际就是添加yum源配置)

[root@cdh-03 ~]# yum install epel-release -y

yum install haveged -y

vim /usr/java/jdk1.8/jre/lib/security/java.security

kill -0 $pid中的-0表示不发送任何信号给PID对应的进程，但是仍会对变量值PID对应的进程是否存在进行检查，如果$pid对应的进程存在，则返回0，不存在返回1

if kill -0 `cat $pid` > /dev/null 2>&1; then  
 echo Agent started !  
 exit 0  
fi

netstat -ano

-tpl

**lsof -i:端口号**

查看显卡nvidia-smi

两台机子A/B

A: nc -l 9999 在A机器上启动一个端口监听，比如 9999端口,默认tcp

telnet B 9999测试连接

nc -vzw 2 B 9998-9999 扫描两个端口

-w<超时秒数> 设置等待连线的时间。

端口扫描经常被系统管理员和黑客用来发现在一些机器上开放的端口，帮助他们识别系统中的漏洞。

$ nc -z -v -n 172.31.100.7 21-25

可以运行在TCP或者UDP模式，默认是TCP，-u参数调整为udp.

z 参数告诉netcat使用0 IO,连接成功后立即关闭连接， 不进行数据交换

v 参数指使用冗余选项

n 参数告诉netcat 不要使用DNS反向查询IP地址的域名

这个命令会打印21到25 所有开放的端口。

$ nc -v 172.31.100.7 21

netcat 命令会连接开放端口21并且打印运行在这个端口上服务的banner信息。

1. 通过 kill `ps -ef | grep 'ddd'`      
#这种形式，这个时候实际上等同于拼接字符串得到的命令，其效果类似于  kill $pid

2. for procid in $(ps -aux | grep "some search" | awk '{print $2}'); do kill -9 $procid; done     
#其实与第一种原理一样，只不过需要多次kill的时候是循环处理的，每次处理一个

3. ps -ef | grep 'ddd' | xargs kill    
#OK，使用了xargs命令，铺垫了这么久终于铺到了主题上。xargs命令可以通过管道接受字符串，并将接收到的字符串通过空格分割成许多参数(默认情况下是通过空格分割) 然后将参数传递给其后面的命令，作为后面命令的命令行参数，kill不直接接受管道

Xargs -d分隔符默认为空格

-p 询问命令是否执行

-n 每次传递几个参数给其后面的命令执行

-E 只取该参数对应字符之前的数据

find . -name "\*.txt" -print0 | xargs -0 echo

xargs的 -0 和 -d '\0' 表示其从标准输入中读取的内容使用 '\0' 来分割

command || { echo "command failed"; exit 1; }

将一个已有用户 cnzhx 增加到一个已有用户组 apache 中，使此用户组成为该用户的附加用户组，可以使用带 -a 参数的 usermod

# usermod -a -G apache cnzhx

指令。-a 代表 append， 也就是将用户添加到新用户组中而不必离开原有的其他用户组。不过需要与 -G 选项配合使用：

如果要同时将 cnzhx 的主要用户组改为 apache，则直接使用 -g 选项：

# usermod -g apache cnzhx

如果要将一个用户从某个组中删除，则

gpasswd -d user group

======================

====================

你写的每个脚本都应该在文件开头加上set -e,这句语句告诉bash如果任何语句的执行结果不是true则应该退出。

这样的好处是防止错误像滚雪球般变大导致一个致命的错误，而这些错误本应该在之前就被处理掉。如果要增加可读性，

可以使用set -o errexit，它的作用与set -e相同。

command

if [ "$?" -ne 0]; then

echo "command failed";

exit 1;

fi

1. **基础知识。**

登陆：

#!bin/bash

su root<<EOF

12qwaszx

whoami

pwd

ifconfig

whoami

EOF

#!/usr/bin/expect -f

#不可以去掉spawn，英文expect中没有ssh命令

spawn ssh root@mu4

expect "\*password:"

send "123456\r"

expect "\*#"

#有interact才可以停留在mu4主机，否则会退回mu3主机

interact

2，模式-动作。

单一分支模式语法：

expect "hi" {send "You said hi"}

匹配到hi后，会输出"you said hi"

多分支模式语法：

expect "hi" { send "You said hi\n" } \

"hello" { send "Hello yourself\n" } \

"bye" { send "That was unexpected\n" }

匹配到hi,hello,bye任意一个字符串时，执行相应的输出。等同于如下写法：

expect {

"hi" { send "You said hi\n"}

"hello" { send "Hello yourself\n"}

"bye" { send "That was unexpected\n"}

}

文件描述符

#打开test.sh可读写操作，与文件描述符3绑定

exec 3<>test.sh;

while read line < **&3**

do

echo $line;

done

#循环读取文件描述符3（读取的是test.sh内容）

exec 3>&-

exec 3<&-

#关闭文件的，输入，输出绑定

exec 6>&1

#将标准输出与fd 6绑定

$ ls /proc/self/fd/

0 1 2 3 6

#出现文件描述符6

$ exec 1>suc.txt

#将接下来所有命令标准输出，绑定到suc.txt文件（输出到该文件）

$ ls -al

#执行命令，发现什么都不返回了，因为标准输出已经输出到suc.txt文件了

$ exec 1>&6

#恢复标准输出

--------------------------------

-------------------------------------

#!bin/bash

PRONUM=10 #进程个数

tmpfile="$$.fifo" #临时生成管道文件

mkfifo $tmpfile

exec 6<>$tmpfile

rm $tmpfile

for(( i=0; i<$PRONUM; i++ ))

do

echo "init."

done >&6

while read line

do

read -u6

{ table\_name="$line"

if [[ "$table\_name" == $prefix\* ]]

then

out\_table\_name=$table\_name

else

out\_table\_name=$prefix$table\_name

fi

sqoop import \

--connect jdbc:srdbsql://${databaseip[0]}:${databaseport[0]}/${databasetable[0]} \

--driver org.srdbsql.Driver \

--username ${databaseuser[0]} \

--password ${databasepwd[0]} \

--query "select \

\* \

from $database\_model.$table\_name where \$CONDITIONS" \

-z \

--null-string '' --null-non-string '' \

--delete-target-dir --target-dir /ods/srdb/jk/$out\_table\_name \

--fields-terminated-by '\001' -m 1

if [ $? -ne 0 ]

then

echo $table\_name >> /txdb/ods/offline/sqoop/srdb/jk/impl/BUG\_table

fi

echo >& 6

echo "$table\_name finished"

} &

done < /txdb/ods/offline/sqoop/srdb/jk\_table\_list

wait

----------------------------------

------------------------------

#!/bin/bash

#导入到hdfs

#table\_name=表名

#-z 开启压缩，GZip

#--compression-codec org.apache.hadoop.io.compress.BZip2Codec BZip格式

databaseip=(`awk -F '|' '{print $1}' /txdb/ods/offline/sqoop/srdb/database\_config.cfg`)

databaseport=(`awk -F '|' '{print $2}' /txdb/ods/offline/sqoop/srdb/database\_config.cfg`)

databasetable=(`awk -F '|' '{print $3}' /txdb/ods/offline/sqoop/srdb/database\_config.cfg`)

databaseuser=(`awk -F '|' '{print $4}' /txdb/ods/offline/sqoop/srdb/database\_config.cfg`)

databasepwd=(`awk -F '|' '{print $5}' /txdb/ods/offline/sqoop/srdb/database\_config.cfg`)

database\_model=(`awk -F '|' '{print $6}' /txdb/ods/offline/sqoop/srdb/database\_config.cfg`)

prefix="t\_"

out\_table\_name=""

for table\_name in `cat /txdb/ods/offline/sqoop/srdb/jk\_table\_list`

do

{

if [[ "$table\_name" == $prefix\* ]]

then

out\_table\_name=$table\_name

else

out\_table\_name=$prefix$table\_name

fi

sqoop import \

--connect jdbc:srdbsql://${databaseip[0]}:${databaseport[0]}/${databasetable[0]} \

--driver org.srdbsql.Driver \

--username ${databaseuser[0]} \

--password ${databasepwd[0]} \

--query "select \

\* \

from $database\_model.$table\_name where \$CONDITIONS" \

-z \

--null-string '' --null-non-string '' \

--delete-target-dir --target-dir /ods/srdb/jk/$out\_table\_name \

--fields-terminated-by '\001' -m 1

if [ $? -ne 0 ]

then

echo $table\_name >> /txdb/ods/offline/sqoop/srdb/jk/impl/BUG\_table

fi

}&

done{split}

**Awk命令:**

$0 代表整行

NF 每一行($0)拥有字段总数

NR 目前awk处理的第几行

FS 目前分隔符，默认是空白

$ cat /etc/passwd|awk '{FS=":"} $3<10 {print $0}'

root:x:0:0:root:/root:/bin/zsh

bin:x:1:1:bin:/bin:/sbin/nologin

daemon:x:2:2:daemon:/sbin:/sbin/nologin

awk -v # 设置变量

$ awk -va=1 '{print $1,$1+a}' log.txt

使用多个分隔符.先使用空格分割，然后对分割结果再使用","分割

$ awk -F '[ ,]' '{print $1,$2,$5}' log.txt

awk 'BEGIN{   
test=100;  
if(test>90)  
{  
    print "very good";  
}  
else if(test>60)  
{  
    print "good";  
}  
else  
{  
    print "no pass";  
}

}'

过滤条件求和：

$ cat test

aaa 123.52

bbb 125.54

aaa 123.52

aaa 123.52

ccc 126.36

对文件test中 第一列为aaa的行求和

$ awk '/aaa/ {sum += $2};END {print sum}' test370.56

awk ' BEGIN { print "文件中出现 tecmint.com 的次数是：" ; }

/^tecmint.com/ { counter+=1 ; }

END { printf "%s\n", counter ; } ' $file

**Sed语法**

sed [-hnV][-e<script>][-f<script文件>][文本文件]

参数说明：

-e<script>或--expression=<script> 以选项中指定的script来处理输入的文本文件。

-f<script文件>或--file=<script文件> 以选项中指定的script文件来处理输入的文本文件。

-h或--help 显示帮助。

-n或--quiet或--silent 仅显示script处理后的结果。

-V或--version 显示版本信息。

动作说明：

a ：新增， a 的后面可以接字串，而这些字串会在新的一行出现(目前的下一行)～

c ：取代， c 的后面可以接字串，这些字串可以取代 n1,n2 之间的行！

d ：删除，因为是删除啊，所以 d 后面通常不接任何咚咚；

i ：插入， i 的后面可以接字串，而这些字串会在新的一行出现(目前的上一行)；

p ：打印，亦即将某个选择的数据印出。通常 p 会与参数 sed -n 一起运行～

s ：取代，可以直接进行取代的工作！通常这个 s 的动作可以搭配正规表示法！例如 1,20s/old/new/g 就是啦！

在testfile文件的第四行后添加一行newLine，并将结果输出到标准输出，在命令行提示符下输入如下命令：

sed -e 4a\newLine testfile

将 /etc/passwd 的内容列出并且列印行号，同时，请将第 2~5 行删除！

[root@www ~]# nl /etc/passwd | sed '2,5d'

在第二行后(亦即是加在第三行)加上『drink tea?』字样！

[root@www ~]# nl /etc/passwd | sed '2a drink tea'

那如果是要在第二行前

nl /etc/passwd | sed '2i drink tea'

如果是要增加两行以上，在第二行后面加入两行字，例如『Drink tea or .....』与『drink beer?』

[root@www ~]# nl /etc/passwd | sed '2a Drink tea or ......\

> drink beer ?'

每一行之间都必须要以反斜杠『 \ 』来进行新行的添加

将第2-5行的内容取代成为『No 2-5 number』呢？

[root@www ~]# nl /etc/passwd | sed '2,5c No 2-5 number'

仅列出 /etc/passwd 文件内的第 5-7 行

[root@www ~]# nl /etc/passwd | sed -n '5,7p'

删除/etc/passwd所有包含root的行，其他行输出

nl /etc/passwd | sed '/root/d'

搜索/etc/passwd,找到root对应的行，执行后面花括号中的一组命令，每个命令之间用分号分隔，这里把bash替换为blueshell，再输出这行：

nl /etc/passwd | sed -n '/bash/{s/bash/blueshell/;p;q}'

1 root:x:0:0:root:/root:/bin/blueshell

一条sed命令，删除/etc/passwd第三行到末尾的数据，并把bash替换为blueshell

nl /etc/passwd | sed -e '3,$d' -e 's/bash/blueshell/'

将 IP 前面的部分予以删除

[root@www ~]# sed 's/^.\*addr://g'

### 直接修改文件内容(危险动作)

sed 可以直接修改文件的内容，不必使用管道命令或数据流重导向！ 不过，由於这个动作会直接修改到原始的文件，所以请你千万不要随便拿系统配置来测试！

利用 sed 将 regular\_express.txt 内每一行结尾若为 . 则换成 !

[root@www ~]# sed -i 's/\.$/\!/g' regular\_express.txt

利用 sed 直接在 regular\_express.txt 最后一行加入『# This is a test』

[root@www ~]# sed -i '$a # This is a test' regular\_express.txt

**服务：**

1.按一定的规则编写服务脚本，比如：myserviced

[plain] view plain copy print?

#!/bin/sh

#chkconfig: 2345 80 05

#description: myservice

case $1 in

start)

echo "myservice startup" #将该行替换成你自己的服务启动命令

;;

stop)

echo "myservice stop" #将该行替换成你自己服务的启动命令

;;

restart)

echo "myservice stop" #...

echo "myservice startup" #...

;;

\*)

;;

esac

命令解析：

# chkconfig: 2345 80 5

2345表示服务的运行级别，80代表Start的顺序，05代表Kill（Stop）的顺序；

# description: service\_description

该服务的描述

2.将编写的脚本放到/etc/init.d/，将myserviced的访问权限加上“可执行”

[plain] view plain copy print?

chmod +x myserviced

3.增加服务

[plain] view plain copy print?

chkconfig --add myserviced

4.启停服务

[plain] view plain copy print?

service myserviced start

service myserviced stop

服务添加完成

可以用：chkconfig --list查看当前系统的服务

可以用：chkconfig --del myserviced删除服务

**系统**  
  
# uname -a               # 查看内核/操作系统/CPU信息  
# head -n 1 /etc/issue   # 查看**操作系统**版本  
# cat /proc/cpuinfo      # 查看CPU信息  
# hostname               # 查看计算机名  
# lspci -tv              # 列出所有PCI设备  
# lsusb -tv              # 列出所有USB设备  
# lsmod                  # 列出加载的内核模块  
# env                    # 查看环境变量

**资源**  
  
# free -m                # 查看内存使用量和交换区使用量  
# df -h                  # 查看各分区使用情况  
# du -sh <目录名>        # 查看指定目录的大小  
# grep MemTotal /proc/meminfo   # 查看内存总量  
# grep MemFree /proc/meminfo    # 查看空闲内存量  
# uptime                 # 查看系统运行时间、用户数、负载  
# cat /proc/loadavg      # 查看系统负载

**磁盘和分区**  
  
# mount | column -t      # 查看挂接的分区状态  
# fdisk -l               # 查看所有分区  
# swapon -s              # 查看所有交换分区  
# hdparm -i /dev/hda     # 查看磁盘参数(仅适用于IDE设备)  
# dmesg | grep IDE       # 查看启动时IDE设备检测状况

**网络**  
  
# ifconfig               # 查看所有网络接口的属性  
# iptables -L            # 查看防火墙设置  
# route -n               # 查看路由表  
# netstat -lntp          # 查看所有监听端口  
# netstat -antp          # 查看所有已经建立的连接  
# netstat -s             # 查看网络统计信息

**进程**  
  
# ps -ef                 # 查看所有进程  
# top                    # 实时显示进程状态  
 **用户**  
  
# w                      # 查看活动用户  
# id <用户名>            # 查看指定用户信息  
# last                   # 查看用户登录**日志**  
# cut -d: -f1 /etc/passwd   # 查看系统所有用户  
# cut -d: -f1 /etc/group    # 查看系统所有组  
# crontab -l             # 查看当前用户的计划任务  
 **服务**  
  
# chkconfig --list       # 列出所有系统服务  
# chkconfig --list | grep on    # 列出所有启动的系统服务

**程序**

# rpm -qa                # 查看所有安装的软件包

**查看本机信息dmidecode | more**