1、查看隐藏文件

|  |
| --- |
| ls -al |

2、删除目录

|  |
| --- |
| rm -rf 目录名 |

3、查看linux时间

|  |
| --- |
| date |

4、查询安装程序(java)

|  |
| --- |
| rpm -qa |grep java |

5、查找可执行文件

|  |
| --- |
| which java |

6、重新加载配置

|  |
| --- |
| source ~/.bash\_profile  source .bash\_profile |

7、yum卸载软件

|  |
| --- |
| yum remove maven |

vi编辑命令模式下，

用/搜索，下一个用n，上一个用大写N（shift+n或者大写锁定键）

用问号？从文章结尾处搜索

用/搜索，搜索的内容保持原样，不需要添加单引号，双引号。

比如下面的字符串：

Caused by: java.sql.SQLSyntaxErrorException: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'from p\_orders po

我们打开别的文件，发现也被高亮了，怎么关闭高亮？

nohlsearch是（no highlight search缩写）

set nohlsearch

简写：noh或者set noh

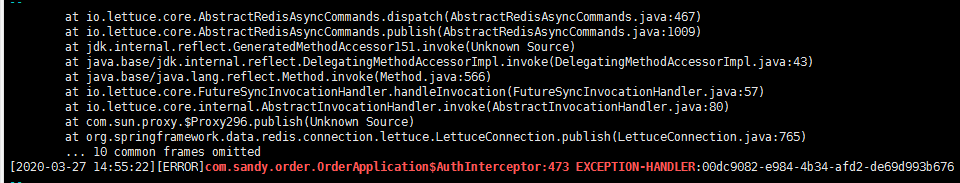
vim 命令模式

gg 回到文档最开始

G 到文件末尾

从文件末尾查询com.sandy.order.OrderApplication$AuthInterceptor:473 EXCEPTION-HANDLER，多显示前面的10行

tac service-order.log | grep -B 10 'com.sandy.order.OrderApplication$AuthInterceptor:473 EXCEPTION-HANDLER'



从文件前面往后查询

cat service-order.log | grep -B 10 'com.sandy.order.OrderApplication$AuthInterceptor:473 EXCEPTION-HANDLER'

grep的更多选项 使用 grep --help

grep -C 5 foo file 显示file文件里匹配foo字串那行以及上下5行

grep -B 5 foo file 显示foo及前5行

grep -A 5 foo file 显示foo及后5行

配置好环境变量 .bash\_profile /etc/profile等文件

source .bash\_profile

ssh -p 2222 user@host

ssh命令登录远程机器

假定经过风险衡量以后，用户决定接受这个远程主机的公钥。  
    　　Are you sure you want to continue connecting (yes/no)? yes  
系统会出现一句提示，表示host主机已经得到认可。  
    　　Warning: Permanently added 'host,12.18.429.21' (RSA) to the list of known hosts.  
然后，会要求输入密码。  
    　　Password: (enter password)  
如果密码正确，就可以登录了。

sudo -u userprofile -i

debian 安装软件用apt-get

apt-get update

apt-get install openssh-server -y

apt-get install net-tools

apt-get install vim

ssh命令

ssh -p 58958 userprofile@169.44.23.158

ssh长传文件命令

rz

sz 文件名

复制文件夹

cp -rf 源文件夹目标地址

删除文件夹

cp -rf 文件夹

cp -rf elasticsearch-6.4.2-cp1/ ./elasticsearch-6.4.2-cp2

修改文件执行、读写权限

chmod [a|u|g]+[w|r|x] 文件

修改文件所有者

chown[[-f](http://study.chyangwa.com/IT/AIX/aixcmds1/chown.htm#ph3200wmh3)][[-h](http://study.chyangwa.com/IT/AIX/aixcmds1/chown.htm#d722dea943jani)][[-R](http://study.chyangwa.com/IT/AIX/aixcmds1/chown.htm#dh3240wmh3)] Owner [ :Group ] { File ... | Directory ... }  
chown[-R](http://study.chyangwa.com/IT/AIX/aixcmds1/chown.htm#dh3240wmh3)[[-f](http://study.chyangwa.com/IT/AIX/aixcmds1/chown.htm#ph3200wmh3)][[-H](http://study.chyangwa.com/IT/AIX/aixcmds1/chown.htm#d722dea001)| [-L](http://study.chyangwa.com/IT/AIX/aixcmds1/chown.htm#d722dea002)| [-P](http://study.chyangwa.com/IT/AIX/aixcmds1/chown.htm#d722dea003)] Owner [ :Group ] { File ... | Directory ... }

查看端口状态

|  |
| --- |
| **netstat -an | grep 3306   //查看所有3306端口使用情况** |

1、w命令查看当前用户

[root@localhost /]# w  
04:31:52 up  1:45,  2 users,  load average: 0.00, 0.01, 0.05  
USER    TTY        FROM            LOGIN@  IDLE    JCPU  PCPU  WHAT  
root      tty1                                02:46      1:45m  0.02s  0.02s  -bash  
root      pts/0    192.168.146.1    03:19      0.00s    0.10s  0.02s  w

第一行分别是：当前时间、从开机到现在有多久、负载均衡  
      第二行是：每列的说明  
      第三行是：root用户的登录信息

2、who命令查看主机上的用户

[root@localhost /]# who  
root    tty1          2017-04-19 02:46  
root    pts/0        2017-04-19 03:19 (192.168.146.1)  
【附】：查看当前登录用户是谁  
whoami  （字母之间没有空格）

3、lastlog命令查看主机上的用户

[root@localhost /]# lastlog | head -n 15  
Username        Port        From                      Latest  
root                  pts/0      192.168.146.1        Wed Apr 19 03:19:47 -0400 2017  
bin                                                                \*\*Never logged in\*\*  
daemon                                                        \*\*Never logged in\*\*  
adm                                                              \*\*Never logged in\*\*  
lp                                                                  \*\*Never logged in\*\*  
sync                                                              \*\*Never logged in\*\*

tar压缩命令

参数：

-c ：create 建立压缩档案的参数；

-x ：解压缩压缩档案的参数；

-z ：是否需要用gzip压缩；

-v：压缩的过程中显示档案；

-f：置顶文档名，在f后面立即接文件名，不能再加参数

1 将tgz文件解压到指定目录

tar   zxvf    test.tgz  -C  指定目录

比如将/source/kernel.tgz解压到  /source/linux-2.6.29 目录

tar  zxvf  /source/kernel.tgz  -C /source/ linux-2.6.29

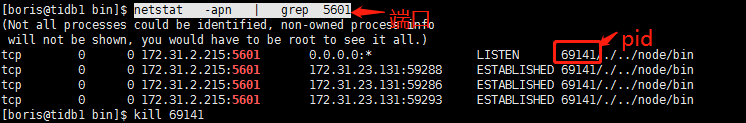
2 将指定目录压缩到指定文件

比如将linux-2.6.29 目录压缩到  kernel.tgz

 tar czvf   kernel.tgz   linux-2.6.29

查看端口占用情况

netstat -apn | grep 5601



杀掉进程

kill pid

日志分析：

查看日志|搜索cod免验证算法接口总耗时所在的行|用冒号分隔在屏幕数据第二个字段|用秒分隔在屏幕输出第一个字段|排序

cat jolly-service-risk-busi.log | grep "cod免验证算法接口总耗时" | awk -F "：" '{print $2}' | awk -F "秒" '{print $1}' | sort |

cat jolly-service-risk-busi.log | grep "cod免验证算法接口总耗时" | wc -l

cat jolly-service-risk-busi.log | grep "cod免验证算法接口总耗时" | sort -k 14 19

cat jolly-service-risk-busi.log | grep "cod免验证算法接口总耗时" | awk -F '[：秒]' '$2>0.16 {print $2}' | wc -l

cat jolly-service-risk-busi.log | grep "cod免验证算法接口总耗时" | awk -F '[：秒]' '$2>0.16 {print $2}' | sort | wc -l

cat jolly-service-risk-busi.log | grep "cod免验证算法接口总耗时" | awk -F '[：秒]' '$2>0.16'

cat jolly-service-risk-busi.log | grep "cod免验证算法接口总耗时" | awk -F '[：秒]' '$2>0.16' | sort -k 2

根据分隔符分出字段，在取第二个字段的平均值，显示

grep "调用算法http耗时" jolly-service-risk-busi.log.copy |awk -F '[：秒]' '{sum += $2;} END { print "avg=" sum/NR}'

输出匹配字符串的行号

awk '/cod免验证算法接口总耗时/ {print NR}' jolly-service-risk-busi.log

跟grep -n ' cod免验证算法接口总耗时' jolly-service-risk-busi.log 相似

grep -niR '想要查找的字符' xxx

注：n，显示行号　R，查找所有文件包含子目录　i,忽略大小写 最后的“xxx”表示想要查找的文件名

grep -rn "hello,world!" ./  
  
./ : 表示路径为当前目录.  
  
-r 是递归查找  
  
-n 是显示行号

将一个文件的部分数据copy到另一个文件，相当于拆分文件

tail -n 10000 jolly-service-risk-busi.log>jolly-service-risk-busi.log.copy

也可以做合并

tail jolly-service-risk-busi.log>>jolly-service-risk-busi.log.copy

**日志分析查看——grep,sed,sort,awk运用**

https://blog.csdn.net/ultrani/article/details/6750434#content\_t4

# 在linux后台运行脚本的方法和命令

置顶 2018年05月03日 18:15:55 [izar](https://me.csdn.net/ruiyelp) 阅读数：6971 标签： [后台运行脚本](http://so.csdn.net/so/search/s.do?q=%E5%90%8E%E5%8F%B0%E8%BF%90%E8%A1%8C%E8%84%9A%E6%9C%AC&t=blog)[linux脚本](http://so.csdn.net/so/search/s.do?q=linux%E8%84%9A%E6%9C%AC&t=blog)[后台运行](http://so.csdn.net/so/search/s.do?q=%E5%90%8E%E5%8F%B0%E8%BF%90%E8%A1%8C&t=blog) 更多

个人分类： [linux](https://blog.csdn.net/ruiyelp/article/category/6742632)

版权声明：本文为博主izar原创文章，未经博主izar允许也可转载。 https://blog.csdn.net/ruiyelp/article/details/80184249

## 后台运行脚本

1. 执行脚本test.sh:./test.sh
2. 中断脚本test.sh：ctrl+c
3. 在1的基础上将运行中的test.sh，切换到后台并暂停：ctrl+z
4. 执行ctrl+z后，test.sh在后台是暂停状态（stopped）,使用命令：bg number让其在后台开始运行（“number”是使用jobs命令查到的 [ ]中的数字，不是pid）
5. **直接在后台运行脚本test.sh**:./test.sh &
6. 查看当前shell环境中已启动的任务情况：jobs
7. 将test.sh切换到前台运行：fg %number（”number”为使用jobs命令查看到的 [ ] 中的数字，不是pid）
8. 中断后台运行的test.sh脚本：先fg %number切换到前台，再ctrl+c；或是直接kill %number

以上两种在后台运行test.sh的方法，当遇到退出当前shell终端时，后台运行的test.sh也就结束了。这是因为以上两种方法使得test.sh在后台运行时，运行test.sh进程的父进程是当前shell终端进程，关闭当前shell终端时，父进程退出，会发送hangup信号给所有子进程，子进程收到hangup以后也会退出。所以要想退出当前shell终端时test.sh继续运行，则需要使用nohup忽略hangup信号。

1. **不中断的在后台运行test.sh**：nohup ./test.sh &（test.sh的打印信息会输出到当前目录下的nohup.out中）
2. 使用jobs可看到test.sh处于running状态
3. 使用ps -ef |grep test.sh可查看到正在运行的test.sh脚本进程
4. 退出当前shell终端，再重新打开，使用jobs看不到正在运行的test.sh，但使用ps -ef可以看到

在后台不中断的运行test.sh，可以使用nohup忽略hangup信号，或者使用setsid将其父进程改为init进程（进程号为1）

1. **不中断的在后台运行test.sh另一个命令**：setsid ./test.sh &
2. 使用ps -ef |grep test.sh可看到test.sh进程的父进程id为1

## 测试脚本

#！/bin/bash

int=1

while(( $int<=100 ))

do

echo$int

let"int++"

sleep 1

done

jps | grep [appname]

netstat -anp | grep [pid]

ping hostname, telnet hostname port

vim

gg:命令将光标移动到文档开头  
  
G:命令将光标移动到文档末尾

* 0 Move to the begining of the line
* $ Move to the end of the line
* 1G Move to the first line of the file
* G Move to the last line of the file
* nG Move to nth line of the file
* :n Move to nth line of the file
* H Move to top of screen
* M Move to middle of screen
* L Move to botton of screen

**一、tail命令语法**

tail [ -f ] [ -c Number | -n Number | -m Number | -b Number | -k Number ] [ File ]  
参数解释：  
-f 该参数用于监视File文件增长。  
-c Number 从 Number 字节位置读取指定文件  
-n Number 从 Number 行位置读取指定文件。  
-m Number 从 Number 多字节字符位置读取指定文件，比方你的文件假设包括中文字，假设指定-c参数，可能导致截断，但使用-m则会避免该问题。  
-b Number 从 Number 表示的512字节块位置读取指定文件。  
-k Number 从 Number 表示的1KB块位置读取指定文件。  
File 指定操作的目标文件名称  
上述命令中，都涉及到number，假设不指定，默认显示10行。Number前面可使用正负号，表示该偏移从顶部还是从尾部開始计算。  
tail可运行文件一般在/usr/bin/以下。

**二、tail命令使用方法演示例子**

1、tail -f filename  
说明：监视filename文件的尾部内容（默认10行，相当于增加参数 -n 10），刷新显示在屏幕上。退出，按下CTRL+C。

2、tail -n 20 filename  
说明：显示filename最后20行。

3、tail -r -n 10 filename  
说明：逆序显示filename最后10行。

补充：  
跟tail功能相似的命令还有：  
cat 从第一行開始显示档案内容。  
tac 从最后一行開始显示档案内容。  
more 分页显示档案内容。  
less 与 more 相似，但支持向前翻页  
head 仅仅显示前面几行  
tail 仅仅显示后面几行  
n 带行号显示档案内容

grep -n "2018-11-16" jolly-service-risk-busi.log

mysqldump  -h 10.142.170.177  -uusertag -pusertag##3  userprofile  表名 > 导出的文件名

grep

# [Linux后台运行命令，nohup和&的区别](https://www.cnblogs.com/chenkeyu/p/8544289.html)

&的意思是在后台运行， 什么意思呢？  意思是说， 当你在执行 ./a.out & 的时候， 即使你用ctrl C,  那么a.out照样运行（因为对SIGINT信号免疫）。 但是要注意， 如果你直接关掉shell后， 那么， a.out进程同样消失。 可见， &的后台并不硬（因为对SIGHUP信号不免疫）。

      nohup的意思是忽略SIGHUP信号， 所以当运行nohup ./a.out的时候， 关闭shell, 那么a.out进程还是存在的（对SIGHUP信号免疫）。 但是， 要注意， 如果你直接在shell中用Ctrl C, 那么, a.out进程也是会消失的（因为对SIGINT信号不免疫）

      所以， &和nohup没有半毛钱的关系， 要让进程真正不受shell中Ctrl C和shell关闭的影响， 那该怎么办呢？ 那就用nohua ./a.out &吧， 两全其美。

       如果你懂守护进程， 那么nohup ./a.out &颇有点让a.out成为守护进程的感觉。

查看软件安装位置

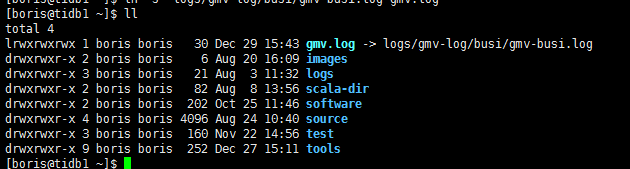
whereis redis

3.如果是rpm包安装的话，可以使用rpm -qal查询一下

rpm -qf /路径/程序名可以查到一个程序属于哪个rpm包，然后rpm -ql那个包可以看到那个包的所有文件和路径等等。还可以解开成一个cpio文件等等之类的，仔细研究一下rpm的参数吧。

linux中的l类型文件，类似windows中的快捷方式。可以通过命令：ln -s 源文件新文件。例如：

ln -s logs/gmv-log/busi/gmv-busi.log gmv.log



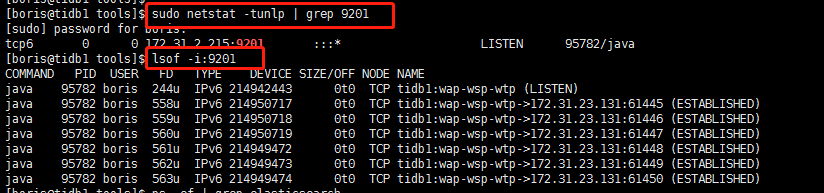
下载文件的命令很多 curl、prozilla、myget、axel

wget url

查看端口占用情况：

sudo netstat -tunlp |grep 端口号

lsof -i:端口号



netstat -ntlp

后台启动nohup &使用

nohup ./tools/elasticsearch-6.4.2-cp1/bin/elasticsearch &

Contos7 远程文件copy

# [Centos7-两台Centos机器间复制文件](https://www.cnblogs.com/happyflyingpig/p/8076150.html)

我又两台Centos机器，一台192.168.1.1；另一台192.168.1.2 现在在将192.168.1.1上的一个文件复制到192.168.1.2。登陆到192.168.1.1然后运行命令

**命令：**

**scp /home/look/dev-software/alibaba-rocektmq-3.2.6.tar.gz 192.168.1.2:/home/look/dev-software/**

然后输入192.168.1.2的密码就可以了

CentOS下可能有n个账户，让vim显示行号有两种方法：仅让当前用户显示行号和让所有用户显示行号

一.仅让当前用户显示行号

输入命令：vim ~/.vimrc

写入：set nu

保存：wq

手动加载配置： source .vimrc

二.让所有用户显示行号

输入命令：vim /etc/vimrc

在vimrc文件的最后添加：set nu

保存：wq

手动加载配置：source /etc/bashrc

这样不管是哪个用户在vim下都显示行号

原文：https://blog.csdn.net/jas0n\_liu/article/details/72377310

查看用户用户组

|  |
| --- |
| 1. 查看用户：cut -d : -f 1 /etc/passwd 2. 查看用户组：cut -d : -f 1 /etc/group 3. 删除xiaoluo这个用户：userdel -r xiaoluo |

1、查找文件

find / -name ‘filename’

2、查找目录

find / -name ‘path’ -type d

3、查找内容

find . | xargs grep -ri ‘content’

//find . | xargs grep -ril ‘content’只显示文件名称

# centos7

## 1、文件搜索

find 可以找到你想要的文件

格式： find [目录] [选项] [选项值]

目录：去哪找，可以不写，默认代表当前目录

选项：怎么找

>> -name 按照名字找

可以使用通配符

-size 按照大小找

单位为 kmg 10k（等于10k） +10k（大于10k） -10k（小于10k）

-user 按照用户名

-group 按照组名

-maxdepth -mindepth 限制查找的目录层级，默认递归查找所有

-ctime 按照创建时间查找

单位是天

选项值：找什么

find / -name demo.txt

find / -name \\*.txt

find / -size +10k

find / -user demo.txt

find / -group demo.txt

find / -mindepth 4 -name \\*.txt

find / -mindepth 3 -maxdepth 5 -name \\*.txt

## 2、文件内容搜索

grep 查找的内容文件路径

grep movie demo.txt

grep that ~/\*.txt

选项

--color=auto 将颜色高亮显示

给 grep 指令起一个别名 vi ~/.bashrc

添加一行 alias grep='grep --color=auto'

让配置文件立即生效 source ~/.bashrc

-c 得到内容的个数

-i 不区分大小写的查找

-n 显示在文档中的行号

-r 递归查找，但是不能限制后缀，只能遍历所有

grep -r that ~/\*

-l 只显示文件名，不显示内容

grep -l 你好 ~/test/\*.txt

正则表达式进行查找(少用)

\w(数字字母下划线)

\W(除了上面)

\d(数字)

\D(非数字)

.(除了换行符)

\*(任意多个)

+(至少1个)

?(0个或者1个)

te-st@163.com abc\_def@qq.com lala@sina.cn benben@meme.net

grep -E .\*? demo.txt

-E 使用正则表达式来进行匹配

前一个管道的输出是后一个管道的输入

前一个指令的输出是后一个指令的输入

格式：

指令1 | 指令2

指令1的结果作为指令2的输入，然后将指令2的输出显示到屏幕中

常用的管道指令

*#ll=select /etc=table | less = page*

ll /etc | less

ll /etc | grep sudo 常用格式

ll /etc | head -10 | tail -5显示前10条的后5条

2019年7月16日星期二

将可执行脚本，编程全局可以执行的命令

#ln –s 需要设置的文件路径 /usr/local/bin/

例如：将/usr/local/webserver/nginx/sbin/nginx 做成可以全局执行的命令

#ln –s /usr/local/webserver/nginx/sbin/nginx /usr/local/bin/nginx

应该就是加入环境变量的操作。

往文件快速写入多行

使用EOF 来处理

<< EOF

. . .

EOF

完整的，比如往test.txt 文件中写入下面多行内容

Kkjdkfjkjfkd

Skfjdkfjslkdf

Skdfjdkfdfkj

可以使用命令cat >test.sh << EOF

Kkjdkfjkjfkd

Skfjdkfjslkdf

Skdfjdkfdfkj

EOF

EOF是end of file缩写，是自定义终止符。在上面的用法中EOF可以用其他字符代替，可以是小写字符，也可以是大写字符

比如

<< BBB

. . .

BBB

< :输入重定向

> :输出重定向

>> :输出重定向,进行追加,不会覆盖之前内容

<< :标准输入来自命令行的一对分隔号的中间内容.

一下执行多条命令。可以通过分号（;）、&&、||来进行分割，但是意义有点差异。

;分割的命令，不管前面的命令执行结果是失败还是成功，会执行完所有的命令。

&&分割的命令，前面的命令执行成功了，才会执行后面的命令；前面命令执行失败了，后面的命令就不会执行了。

||分割的命令，前面的命令执行失败了，才会执行后面的命令；前面的命令执行失败了，后面的命令不会执行了。

一条命令太长可以使用反斜杠\来多行书写命令。

反斜线符号 \ 在Bash中被解释为转义字符，用于去除一个单个字符的特殊意义，它保留了跟随在之后的字符的字面值，换行符除外。在反斜线之后紧接着一个换行字符，转义字符使行得以继续，遇到命令很长时使用反斜线很有效。

反斜杠\让[enter]不具有开始执行的功能。

Windows：“/”是表示参数，“\”是表示本地路径。

Linux和Unix：“/”表示路径，“\”表示转义，“-”和“--”表示参数。

网络：由于网络使用Unix标准，所以网络路径用“/”。

Yum安装openjdk的时候，用\*号，不然会安装不完全，使用的时候会有问题。

yum -y install java-1.8.0-openjdk\*

centos7时间

# date // 查看系统时间

#hwclock // 查看硬件时间

# yum -y install ntp ntpdate 安装ntpdate工具

# ntpdate cn.pool.ntp.org  设置系统时间与网络时间同步

# hwclock --systohc  将系统时间写入硬件时间

timedatectl # 查看系统时间方面的各种状态

timedatectl list-timezones # 列出所有时区

timedatectl set-local-rtc 1 # 将硬件时钟调整为与本地时钟一致, 0 为设置为 UTC 时间

timedatectl set-timezone Asia/Shanghai # 设置系统时区为上海

>/dev/null2>&1

这里的1，2表示都是文件描述符，1表示标准输出stdout，2表示标准错误stderr， /dev/null表示消息垃圾桶，放在里面的消息相当于删掉。

标准输入0    从键盘获得输入 /proc/self/fd/0   
标准输出1    输出到屏幕（即控制台） /proc/self/fd/1   
错误输出2    输出到屏幕（即控制台） /proc/self/fd/2

| **命令** | **介绍** |
| --- | --- |
| command <filename | 以filename文件作为标准输入 |
| command 0<filename | 同上 |
| command <<delimiter | 从标准输入中读入，直到遇到delimiter分隔符 |

我们使用<对输入做重定向，如果符号左边没有写值，那么默认就是0。

<<后面跟的是分割符

例如cat >file << EOF

…

EOF

<后面跟的是文件

参考文档：<https://blog.csdn.net/mengzuchao/article/details/82378769>

timedatectl set-ntp no //关闭时间动态更新

timedatectl set-time "YYYY-MM-DD HH:MM:SS" //设置时间和日期

timedatectl set-ntp yes //开启时间动态更新

#netcat

#yum install netcat –y

在A机器上使用下面命令，开启一个tcp服务器

#nc –l 9000

在B机器上用下面命令和A机器建立连接

#nc 10.10.2.184 9000

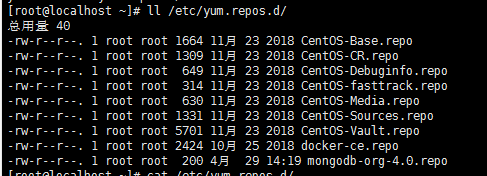
这样A、B机器进行消息传输

服务设置开机启动

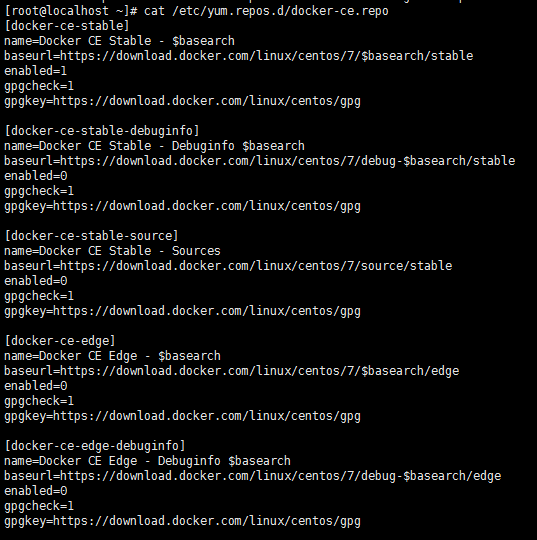
Centos systemctl enable servicename

Created symlink from /etc/systemd/system/multi-user.target.wants/docker.service to /usr/lib/systemd/system/docker.service.

Centos的yum源在目录/etc/yum.repos.d/下面，



Yum源文件的的内容如下：



更新yum源

#yum clean all

#yum makecache

参考：

<https://www.fcwys.cc/archives/256.html>

将数据库7中AUTH:TOKEN\_USER\_前缀的key 放入tokens.txt文件

redis-cli -n 7 keys "AUTH:TOKEN\_USER\_\*" | xargs -n1 echo >> tokens.txt

rz

sz

ll显示文件大小，以M（K，G）显示

--block-size=M

#top 查看服务器资源占用情况

#w

第一位0.31：表示最近1分钟平均负载  
第二位0.30：表示最近5分钟平均负载  
第三位0.31：表示最近15分钟平均负载

**2. load average值的含义**

**1） 单核处理器**

假设我们的系统是单CPU单内核的，把它比喻成是一条单向马路，把CPU任务比作汽车。当车不多的时候，load <1；当车占满整个马路的时候 load=1；当马路都站满了，而且马路外还堆满了汽车的时候，load>1

**2） 多核处理器**

我们经常会发现服务器Load > 1但是运行仍然不错，那是因为服务器是多核处理器（Multi-core）。  
假设我们服务器CPU是2核，那么将意味我们拥有2条马路，我们的Load = 2时，所有马路都跑满车辆。

**注**：查看cpu 核数命令：

grep 'model name' /proc/cpuinfo | wc -l

**3. 什么样的Load average值要提高警惕**

* 0.7 < load < 1: 此时是不错的状态，如果进来更多的汽车，你的马路仍然可以应付。
* load = 1: 你的马路即将拥堵，而且没有更多的资源额外的任务，赶紧看看发生了什么吧。
* load > 5: 非常严重拥堵，我们的马路非常繁忙，每辆车都无法很快的运行

**4. 三种Load值，应该看哪个**

通常我们先看15分钟load，如果load很高，再看1分钟和5分钟负载，查看是否有下降趋势。  
1分钟负载值 > 1，那么我们不用担心，但是如果15分钟负载都超过1，我们要赶紧看看发生了什么事情。所以我们要根据实际情况查看这三个值。

查看进程中线程占用资源情况：

top -Hp 进程pid

查看java栈信息： jstack 进程pid | grep 线程pid（十六进制）

用命令printf %x 十进制数 可以显示其16进制数

①物理cpu数：主板上实际插入的cpu数量，可以数不重复的 physical id 有几个（physical id）

cat /proc/cpuinfo| grep "cpu cores"| uniq

②cpu核数：单块CPU上面能处理数据的芯片组的数量，如双核、四核等 （cpu cores）

cat /proc/cpuinfo| grep "physical id"| sort| uniq| wc -l

③逻辑cpu数：简单来说，它可使处理器中的1颗内核，如2颗内核那样在操作系统中发挥作用。

cat /proc/cpuinfo| grep "processor"| wc -l

④查看cpu信息（型号）

cat /proc/cpuinfo | grep name | cut -f2 -d: | uniq -c

# 总核数 = 物理CPU个数 X 每颗物理CPU的核数

# 总逻辑CPU数 = 物理CPU个数 X 每颗物理CPU的核数 X 超线程数

# 查看物理CPU个数

cat /proc/cpuinfo| grep "physical id"| sort| uniq| wc -l  
或grep 'physical id' /proc/cpuinfo | sort -u | wc -l

# 查看每个物理CPU中core的个数(即核数)

cat /proc/cpuinfo| grep "cpu cores"| uniq  
或者grep 'core id' /proc/cpuinfo | sort -u | wc -l

# 查看逻辑CPU的个数

cat /proc/cpuinfo| grep "processor"| wc -l  
或者grep 'processor' /proc/cpuinfo | sort -u | wc -l

# 查看CPU信息（型号）  
cat /proc/cpuinfo | grep name | cut -f2 -d: | uniq -c

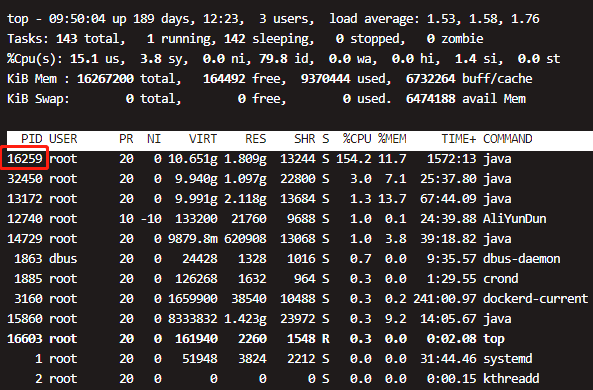
或者dmidecode -s processor-version

#查看内 存信息  
cat /proc/meminfo

解压jar包

unzip xx.jar -d 目录

top 可以看进程消耗的资源，只能看出来是java程序



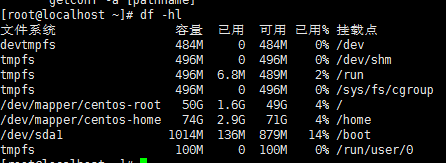
不知道究竟是哪一个java程序。

查看java是哪一个程序，可通过ll /proc/16259/cwd mvn:springboot 方式启动的程序



ps -ef|grep 29307 java -jar方式启动

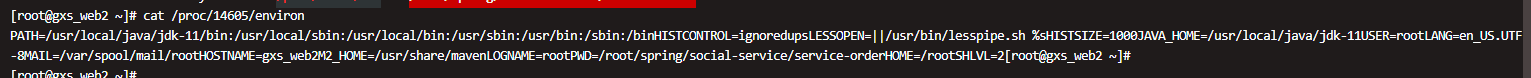
查看磁盘文件



查看与终端相关的所有环境变量

env

查看程序运行使用的环境变量 cat /proc/$PID/environ



查看全局环境变量：env

查看所有环境变量：set

删除一个变量：unset [变量名] #只针对当前会话

设置一个变量：name=songjiankang

export name #将这个变量提升为全局的，注意只能是当前会话可以访问，要想下次也可以访问必须写到文件里

环境变量的配置文件：

全局：/etc/profile

局部：~/.bash\_profile

查看内存

free

-k 以kB为单位

-m 以兆为单位

-g 以g为单位

1、查看当前用户所属组

#groups

2、查看指定用户所属组

#groups 用户名

#id 用户名

3、查看系统中用户组文件

#cat /etc/group

#cat /etc/group | grep 用户名

dig命令安装

yum install bind-utils -y

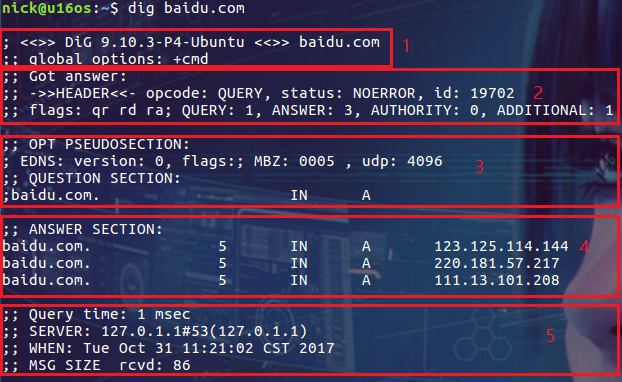
dig从DNS域名服务查询主机信息

查询单个主机：

dig www.baidu.com

dig 命令最典型的用法就是查询单个主机的信息。

$ dig baidu.com



dig 命令默认的输出信息比较丰富，大概可以分为 5 个部分。  
第一部分显示 dig 命令的版本和输入的参数。  
第二部分显示服务返回的一些技术详情，比较重要的是 status。如果 status 的值为 NOERROR 则说明本次查询成功结束。  
第三部分中的 "QUESTION SECTION" 显示我们要查询的域名。  
第四部分的 "ANSWER SECTION" 是查询到的结果。  
第五部分则是本次查询的一些统计信息，比如用了多长时间，查询了哪个 DNS 服务器，在什么时间进行的查询等等。

默认情况下 dig 命令查询 A 记录，上图中显示的 A 即说明查询的记录类型为 A 记录。在尝试查询其它类型的记录前让我们先来了解一下常见的 DNS 记录类型。

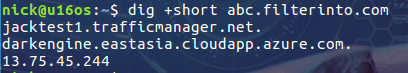
**常见 DNS 记录的类型**

|  |  |
| --- | --- |
| 类型 | 目的 |
| A | 地址记录，用来指定域名的 IPv4 地址，如果需要将域名指向一个 IP 地址，就需要添加 A 记录。 |
| AAAA | 用来指定主机名(或域名)对应的 IPv6 地址记录。 |
| CNAME | 如果需要将域名指向另一个域名，再由另一个域名提供 ip 地址，就需要添加 CNAME 记录。 |
| MX | 如果需要设置邮箱，让邮箱能够收到邮件，需要添加 MX 记录。 |
| NS | 域名服务器记录，如果需要把子域名交给其他 DNS 服务器解析，就需要添加 NS 记录。 |
| SOA | SOA 这种记录是所有区域性文件中的强制性记录。它必须是一个文件中的第一个记录。 |
| TXT | 可以写任何东西，长度限制为 255。绝大多数的 TXT记录是用来做 SPF 记录(反垃圾邮件)。 |

显示精简结果 用+short参数

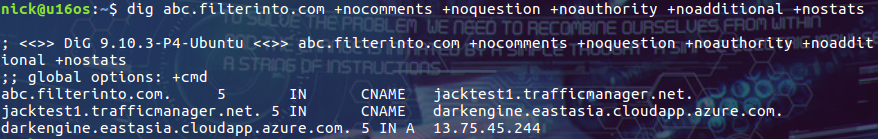
dig 命令默认返回的结果展示详细的信息，如果要获得精简的结果可以使用 +short 选项：

$ dig +short abc.filterinto.com



其实我们还可以通过更多选项来控制输出的内容，比如只想显示 "ANSWER SECTION" 的内容：

$ dig abc.filterinto.com +nocomments +noquestion +noauthority +noadditional +nostats



这个结果很不错，就是使用的选项太多了(dig 命令有很多这样的选项，详情请参考使用手册)。我们可以换一种优雅一些的方式来实现和上面相同的结果：

$ dig abc.filterinto.com +noall +answer

**跟踪整个查询过程**

如果你好奇 dig 命令执行查询时都经历了哪些过程，你可以尝试使用 +trace 选项。它会输出从根域到最终结果的所有信息：

$ dig +trace abc.filterinto.com

根据ip反查域名

**反向查询**

在前面的查询中我们指定了查询服务器为 8.8.8.8，这是谁家的 DNS 服务器？其实我们可以使用 dig 的 -x 选项来反向解析 IP 地址对应的域名：

$ dig -x 8.8.8.8 +short

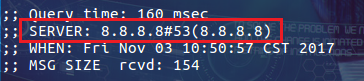
https://images2017.cnblogs.com/blog/952033/201711/952033-20171103134826998-622046289.png

好吧，应该是谷歌家的，可以放心使用了。

**从指定的 DNS 服务器上查询**

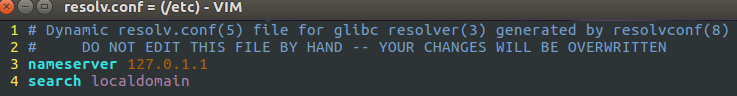
由于一些原因，希望从指定的 DNS 服务器上进行查询(从默认的 DNS 服务器上获得的结果可能不准确)。指定 DNS 服务器的方式为使用 @ 符号：

$ dig @8.8.8.8 abc.filterinto.com



从上图可以看到本次查询的 DNS 服务器为 8.8.8.8。

如果不指定 DNS 服务器，dig 会依次使用 /etc/resolv.conf 里的地址作为 DNS 服务器：



$ dig abc.filterinto.com

上面查询的 DNS 服务器就变成了：

