[**linux之sed用法**](http://www.cnblogs.com/dong008259/archive/2011/12/07/2279897.html)

sed -n '/deploy/p' logs/catalina.out |sed 's/deploy/~guagua~/g'

 sed '1,3a drink tea' ab #第一行到第三行后增加字符串"drink tea"

sed '2,$d' ab           #删除第二行到最后一行

sed '1,2c Hi' ab             #第一行到第二行代替为Hi

sed -n '$p' ab           #显示最后一行

sed -n '/ruby/p' ab | sed 's/ruby/bird/g'    #替换ruby为bird

sed -n '/ruby/p' ab | sed 's/ruby//g'        #删除ruby

-n, --quiet, --silent

suppress automatic printing of pattern space

正则删除匹配部分

sed -e 's/[()]//g

-e script, --expression=script

add the script to the commands to be executed

删除匹配到的模式行

 sed /PATTERN/d filename

示例：在脚本中删除/data/apache-tomcat-7.0.68进程，但不能删除脚本进程

所以用sed /$$/d 剔除,$$代表脚本的进程id

pid=`ps -ef |more |grep /data/apache-tomcat-7.0.68 | awk '{print $2}' | sed /$$/d`

echo $pid

echo "PID of this script: $$"

kill -9 $pid

combo

sed -n '1p' `find logs -name cata\*log -print |sort -r |head -1| awk '{print $1}'`

windows如下脚本经常问题，必须在linux下编辑保存

#!/bin/bash

判断是否传了第一个参数，[ ]表示条件测试。注意这里的空格很重要。要注意在'['后面和']'前面都必须要有空格

if [ -z ${1} ];then

echo "set200"

DISLINE=200

else

echo "remain"

fi

文件表达式

-e filename 如果 filename存在，则为真 文件目录下是否不为空  
-d filename 如果 filename为目录，则为真 -d是判断$!这个目录存不存在  
-f filename 如果 filename为常规文件，则为真 判断一个普通文件是不是存在  
-L filename 如果 filename为符号链接，则为真  
-r filename 如果 filename可读，则为真   
-w filename 如果 filename可写，则为真   
-x filename 如果 filename可执行，则为真  
-s filename 如果文件长度不为0，则为真 文件存在且内容非空 即必须要一个空格以上的字符

-h filename 如果文件是软链接，则为真  
filename1 -nt filename2 如果 filename1比 filename2新，则为真。  
filename1 -ot filename2 如果 filename1比 filename2旧，则为真。

整数变量表达式  
-eq 等于  
-ne 不等于  
-gt 大于  
-ge 大于等于  
-lt 小于  
-le 小于等于

字符串变量表达式  
If  [ $a = $b ]                 如果string1等于string2，则为真  
                                字符串允许使用赋值号做等号  
if  [ $string1 !=  $string2 ]   如果string1不等于string2，则为真         
if  [ -n $string  ]             如果string 非空(非0），返回0(true)    
if  [ -z $string  ]             如果string 为空，则为真  
if  [ $sting ]                  如果string 非空，返回0 (和-n类似)   
  
    逻辑非 !                   条件表达式的相反  
if [ ! 表达式 ]  
if [ ! -d $num ]               如果不存在目录$num  
  
    逻辑与 –a                   条件表达式的并列  
if [ 表达式1  –a  表达式2 ]  
  
    逻辑或 -o                   条件表达式的或  
if [ 表达式1  –o 表达式2 ]

[ -a FILE ] 如果 FILE 存在则为真。

[ -b FILE ] 如果 FILE 存在且是一个块特殊文件则为真。

[ -c FILE ] 如果 FILE 存在且是一个字特殊文件则为真。

[ -d FILE ] 如果 FILE 存在且是一个目录则为真。

[ -e FILE ] 如果 FILE 存在则为真。

[ -f FILE ] 如果 FILE 存在且是一个普通文件则为真。

[ -g FILE ] 如果 FILE 存在且已经设置了SGID则为真。

[ -h FILE ] 如果 FILE 存在且是一个符号连接则为真。

[ -k FILE ] 如果 FILE 存在且已经设置了粘制位则为真。

[ -p FILE ] 如果 FILE 存在且是一个名字管道(F如果O)则为真。

[ -r FILE ] 如果 FILE 存在且是可读的则为真。

[ -s FILE ] 如果 FILE 存在且大小不为0则为真。

[ -t FD ] 如果文件描述符 FD 打开且指向一个终端则为真。

[ -u FILE ] 如果 FILE 存在且设置了SUID (set user ID)则为真。

[ -w FILE ] 如果 FILE 如果 FILE 存在且是可写的则为真。

[ -x FILE ] 如果 FILE 存在且是可执行的则为真。

[ -O FILE ] 如果 FILE 存在且属有效用户ID则为真。

[ -G FILE ] 如果 FILE 存在且属有效用户组则为真。

[ -L FILE ] 如果 FILE 存在且是一个符号连接则为真。

[ -N FILE ] 如果 FILE 存在 and has been mod如果ied since it was last read则为真。

[ -S FILE ] 如果 FILE 存在且是一个套接字则为真。

[ FILE1 -nt FILE2 ] 如果 FILE1 has been changed more recently than FILE2, or 如果 FILE1 exists and FILE2 does not则为真。

[ FILE1 -ot FILE2 ] 如果 FILE1 比 FILE2 要老, 或者 FILE2 存在且 FILE1 不存在则为真。

[ FILE1 -ef FILE2 ] 如果 FILE1 和 FILE2 指向相同的设备和节点号则为真。

[ -o OPTIONNAME ] 如果 shell选项 “OPTIONNAME” 开启则为真。

[ -z STRING ] “STRING” 的长度为零则为真。

[ -n STRING ] or [ STRING ] “STRING” 的长度为非零 non-zero则为真。

[ STRING1 == STRING2 ] 如果2个字符串相同。 “=” may be used instead of “==” for strict POSIX compliance则为真。

[ STRING1 != STRING2 ] 如果字符串不相等则为真。

[ STRING1 < STRING2 ] 如果 “STRING1” sorts before “STRING2” lexicographically in the current locale则为真。

[ STRING1 > STRING2 ] 如果 “STRING1” sorts after “STRING2” lexicographically in the current locale则为真。

[ ARG1 OP ARG2 ] “OP” is one of -eq, -ne, -lt, -le, -gt or -ge. These arithmetic binary operators return true if “ARG1” is equal to, not equal to, less than, less than or equal to, greater than, or greater than or equal to “ARG2”, respectively. “ARG1” and “ARG2” are integers.

7、使用-z或者-n来检查长度的时候，没有定义的变量也为0  
8、空变量和没有初始化的变量可能会对shell脚本测试产生灾难性的影响，因此在不确定变量的内容的时候，在测试号前使用-n或者-z测试一下  
9、? 变量包含了之前执行命令的退出状态（最近完成的前台进程）（可以用于检测退出状态）

vim 显示行号

在/etc/vimrc 中最下面加入一行set number

/xx 执行正常查找后 有时需要高亮

:set hls 打开高亮  
:set nohls 关闭高亮

:nohlsearch　　关闭当前的高亮显示，如果再次搜索或者按下n或N键，则会再次高亮。

:set incsearch　　逐步搜索模式，对当前键入的字符进行搜索而不必等待键入完成。

:set wrapscan　　重新搜索，在搜索到文件头或尾时，返回继续搜索，默认开启。

:set ignorecase　　忽略大小写的查找

:set noignorecase　　不忽略大小写的查找

自动tab替换为空格是  
set expandtab

查看系统信息 内核

uname -a

cat /proc/version

系统信息 版本

lsb\_release -a，即可列出所有版本信息：

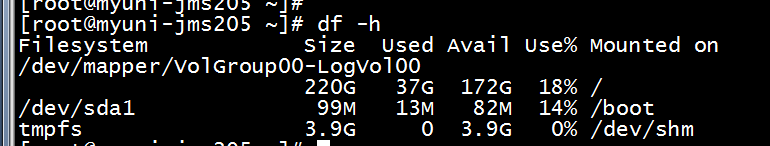
cat /etc/redhat-release，这种方法只适合Redhat系的Linux：

cat /etc/issue，此命令也适用于所有的Linux发行版。

查看磁盘

df - h

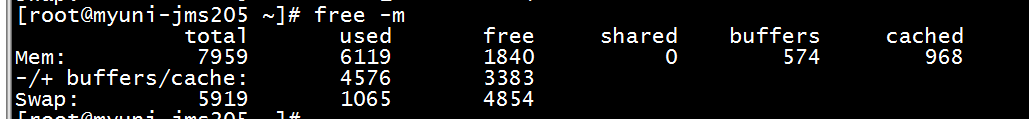
-h, --human-readable print sizes in human readable format (e.g., 1K 234M 2G)



查看内存

cat /proc/meminfo

free -g



查找文件 递归寻找

find / -name treenity

find / -name \*treenity\*.war

service命令，顾名思义，就是用于管理Linux操作系统中服务的命令。

1. 声明：这个命令不是在所有的linux发行版本中都有。主要是在redhat、fedora、mandriva和centos中。

2. 此命令位于/sbin目录下，用file命令查看此命令会发现它是一个脚本命令。

3. 分析脚本可知此命令的作用是去/etc/init.d目录下寻找相应的服务，进行开启和关闭等操作。

比如

**service jboss log**

**service jboss f-restart**

在/etc/init.d目录下有jboss文件

cat jboss

#!/bin/bash

#guohouyu 2015021 script for manage jboss

# chkconfig: 2345 10 90 # Source function library.

. /etc/rc.d/init.d/functions

#Definition

tcName=jboss

basedir=/usr/local/jboss

RETVAL=0

export JAVA\_HOME="/usr/local/jdk1.6.0\_33/"

project\_log=${basedir}/server/default/log/server.log

time=`/bin/date +'%y%m%d%H%M%S'`

key\_word="Started in"

log() {

tail -f $project\_log

}

case分支语句的格式如下：

**case** $变量名 **in**

                模式1**）**

            命令序列1

**;;**

                模式2**）**

            命令序列2

**;;**

**\*）**

            默认执行的命令序列  **;;**

**esac**

case "$1" in

start)

checkexist

start

exit 0

;;

stop)

checkexist

stop

exit 0

;;

restart)

checkexist

stop

start

exit 0

;;

f-restart)

checkexist

kill

start

exit 0

;;

status)

checkexist

status

exit 0

;;

log)

checkexist

log

exit 0

;;

kill)

checkexist

status

kill

exit 0

;;

clear)

checkexist

status

clear

exit 0

;;

\*)

echo "Usage: $0 {start|stop|restart|status|log|kill|clear|f-restart}"

echo " service jboss {0|1|..} {start|stop|restart|status|log|kill|clear|f-restart}"

exit

;;

esac

读取输入

read -p "press some key ,then press return :" KEY

**使用w命令查看登录用户正在使用的进程信息，**

w命令用于显示已经登录系统的用户的名称，以及他们正在做的事。

该命令所使用的信息来源于/var/run/utmp文件。w命令输出的信息包括：

* 用户名称
* 用户的机器名称或tty号
* 远程主机地址
* 用户登录系统的时间
* 空闲时间（作用不大）
* 附加到tty（终端）的进程所用的时间（JCPU时间）
* 当前进程所用时间（PCPU时间）
* 用户当前正在使用的命令

$ **w**

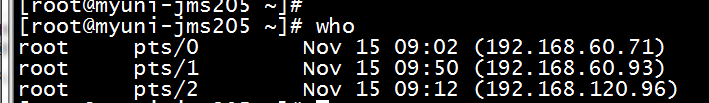
23:04:27 up 29 days, 7:51, 3 users, load average: 0.04, 0.06, 0.02

USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT

ramesh pts/0 10.1.80.56 22:57 8.00s 0.05s 0.01s sshd: ramesh [priv]

jason pts/1 10.20.48 23:01 2:53 0.01s 0.01s -bash

john pts/2 10.1.80.7 23:04 0.00s 0.00s 0.00s w





history

cat ~/.bash\_history

history 添加操作时间

修改/etc/profile文件，在末尾添加：export HISTTIMEFORMAT="`whoami` : | %F | %T: | "

或在用户目录下，修改文件 .bash\_profile，添加export histtimeformat=”%f %t `whoami` ”

source

显示当前ip

USER\_IP=`who -u am i 2>/dev/null| awk '{print $NF}' |sed -e 's/[()]//g'`

hostname

主机名字

awk

NF代表：浏览记录的域的个数  
$NF代表 ：最后一个Field(列)

* /etc/profile  
  这个文件是每个用户登录时都会运行的环境变量设置  
  当用户第一次登录时,该文件被执行. 并从/etc/profile.d目录的配置文件中搜集shell的设置。
* /etc/bashrc: 当 bash shell 被打开时，该文件被读取。也就是说，每次新打开一个终端 shell，该文件就会被读取。

接着是与上述两个文件对应，但只对单个用户生效：

* ~/.bash\_profile 或 ~/.profile: 只对单个用户生效，当用户登录时该文件仅执行一次。用户可使用该文件添加自己使用的 shell 变量信息。另外在不同的LINUX操作系统下，这个文件可能是不同的，可能是 ~/.bash\_profile， ~/.bash\_login 或 ~/.profile 其中的一种或几种，如果存在几种的话，那么执行的顺序便是：~/.bash\_profile、 ~/.bash\_login、 ~/.profile。比如 Ubuntu 系统一般是 ~/.profile 文件。
* ~/.bashrc: 只对单个用户生效，当登录以及每次打开新的 shell 时，该文件被读取。

此外，修改 /etc/environment 这个文件也能实现环境变量的设置。/etc/environment 设置的也是全局变量，从文件本身的作用上来说， /etc/environment 设置的是整个系统的环境，而/etc/profile是设置所有用户的环境。有几点需注意：

* 系统先读取 etc/profile 再读取 /etc/environment（还是反过来？）
* /etc/environment 中不能包含命令，即直接通过 VAR="..." 的方式设置，不使用 export 。
* 使用 source /etc/environment 可以使变量设置在当前窗口立即生效，需注销/重启之后，才能对每个新终端窗口都生效。

.bashfile  
是单用户登录时比如root会运行的

alias vvv=vim

alias vi=vim

定义别名 这时候可以用vi了

有点c中typedef的意思

1. **typedef** **int** (\*FP\_CALC)(**int**, **int**);

!!是执行和显示上一条命令的结果

开启ssh

 rpm -qa |grep ssh 查找当前系统是否已经安装

如果没有安装SSH软件包，可以通过yum  或rpm安装包进行安装

service sshd start 可以启动

E：could not get lock /var/lib/dpkg/lock -open等

解决方法：输入以下命令

sudo rm /var/cache/apt/archives/lock

sudo rm /var/lib/dpkg/lock

解压

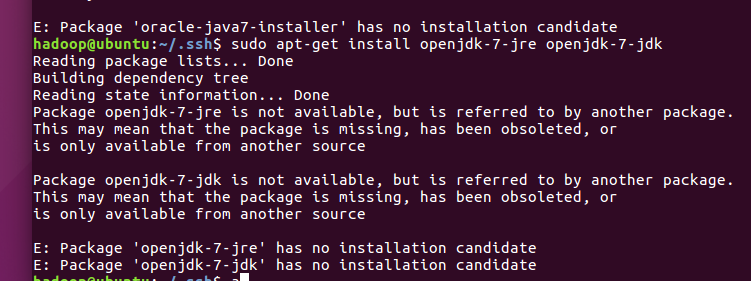
sudo tar -zxf ~/下载/hadoop-2.6.0.tar.gz -C /usr/local # 解压到/usr/local中

cd /usr/local/

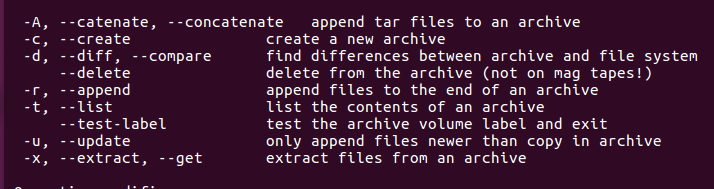
sudo mv ./hadoop-2.6.0/ ./hadoop # 将文件夹名改为hadoop

sudo chown -R hadoop ./hadoop # 修改文件权限

ubuntu安装jdk问题

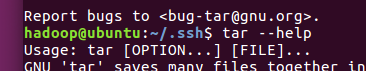


tar -xvf









重命名

mv既可以重命名，又可以移动文件或文件夹.

mv A B

将/a目录移动到/b下，并重命名为c

mv /a /b/c

chown 修改文件和文件夹的用户和用户组属性   
1。要修改文件hh.c的所有者.修改为sakia的这个用户所有   
chown sakia hh.c   
这样就把hh.c的用户访问权限应用到sakia作为所有者   
  
2。将目录 /tmp/sco 这个目录的所有者和组改为sakia和组net   
chown -R sakia:net /tmp/sco   
  
  
chmod 修改文件和文件夹读写执行属性   
1。把hh.c文件修改为可写可读可执行   
chmod 777 hh.c   
要修改某目录下所有的文件属性为可写可读可执行   
chmod 777 \*.\*   
把文件夹名称与后缀名用\*来代替就可以了。   
同理若是要修改所有htm文件的属性   
chmod 777 \*.htm   
2。把目录 /tmp/sco修改为可写可读可执行   
chmod 777 /tmp/sco   
  
要修改某目录下所有的文件夹属性为可写可读可执行   
chmod 777 \*   
把文件夹名称用\*来代替就可以了   
  
要修改/tmp/sco下所有的文件和文件夹及其子文件夹属性为可写可读可执行   
chmod -R 777 /tmp/sco   
  
可写 w=4   
可读 r=2   
可执行 x=1   
777就是拥有全权限。根据需要可以自由组合用户和组的权限

拷贝文件 \* 通配符

cp ./etc/hadoop/\*.xml ./input

. fileName

执行fileName脚本文件

# 安装redis

# apt-get install redis-server,

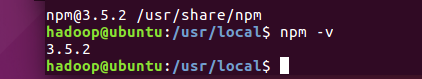
# ps -aux|grep redis

# netstat -nlt|grep 6379

# redis-cli

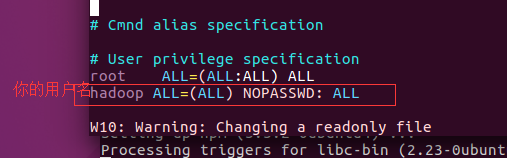
# 安装npm

sudo apt install npm



linux sudo 去密码

sudo vim /etc/sudoers



your\_user\_name ALL=(ALL) NOPASSWD: ALL

gpasswd -a jiangli root

# 安装mysql

**sudo apt-get install mysql-server**

**sudo service mysql status**

**sudo service mysql start**

**sudo service mysql stop**

client:

mysql -uroot -proot

# 如果是deb格式

1、dpkg -i <package.deb>  
安装一个 Debian 软件包，如你手动下载的文件。

sudo dpkg -i \*.deb

# dpkg -i 包名 （结果遇到依赖性问题，需要安装其他的包）  sudo apt-get -f install （使用它就可以自动下载上面所需要的包了）

* **安装（注意安装顺序）**

**安装顺序如下：**

**1.**[**MySQL**](http://lib.csdn.net/base/14)**-common\_5.7.10-1ubuntu14.04\_amd64.deb**

**2.libmysqlclient20\_5.7.10-1ubuntu14.04\_amd64.deb**

**3.libmysqlclient-dev\_5.7.10-1ubuntu14.04\_amd64.deb**

**4.libmysqld-dev\_5.7.10-1ubuntu14.04\_amd64.deb**

**5.mysql-community-client\_5.7.10-1ubuntu14.04\_amd64.deb**

**6.mysql-client\_5.7.10-1ubuntu14.04\_amd64.deb**

**7.mysql-community-server\_5.7.10-1ubuntu14.04\_amd64.deb**

# Dpkg与Apt两种包管理工具的比较

# 两者的区别就是： dpkg安装.deb文件,但不会解决模块的依赖关系,且不会关心ubuntu的软件仓库内的软件,可以用于安装本地的deb文件 apt会解决和安装模块的依赖问题,并会咨询软件仓库, 但不会安装本地的deb文件, apt是建立在dpkg之上的软件管理工具 如果在用dpkg安装.deb遇到依赖问题，那就需要通过 apt-get depends package 来查询依赖关系再用dpkg逐个去安装

# 当然安装本地包用dpkg命令是大家常用的 但是遇到依赖就会很麻烦 大家可以用gdebi来安装本地下载的deb包 它会自动下载所需要的依赖 sudo gdebi package

# apt-get -f install去解决本地deb包的依赖问题了

aptitude 是 Debian GNU/Linux 系统中, 非常神奇的的软件包管理器,基于大名鼎鼎的 APT 机制, 整合了 dselect 和 apt-get的所有功能, 并提供的更多特性,特别是在依赖关系处理上。

aptitude与 apt-get 一样，是 Debian 及其衍生系统\*\*\*能极其强大的包管理工具。与 apt-get 不同的是，aptitude在处理依赖问题上更佳一些。举例来说，aptitude在删除一个包时，会同时删除本身所依赖的包。这样，系统中不会残留无用的包，整个系统更为干净。

# 下载文件

# 如tomcat

wget http://mirrors.cnnic.cn/apache/tomcat/tomcat-7/v7.0.73/bin/apache-tomcat-7.0.73.tar.gz

set "targetName=SpringMVCLearn.war"

scp target/%targetName% hadoop@192.168.208.130:/usr/local/apache-tomcat-7.0.73/webapps/%targetName%

bin/startup.sh

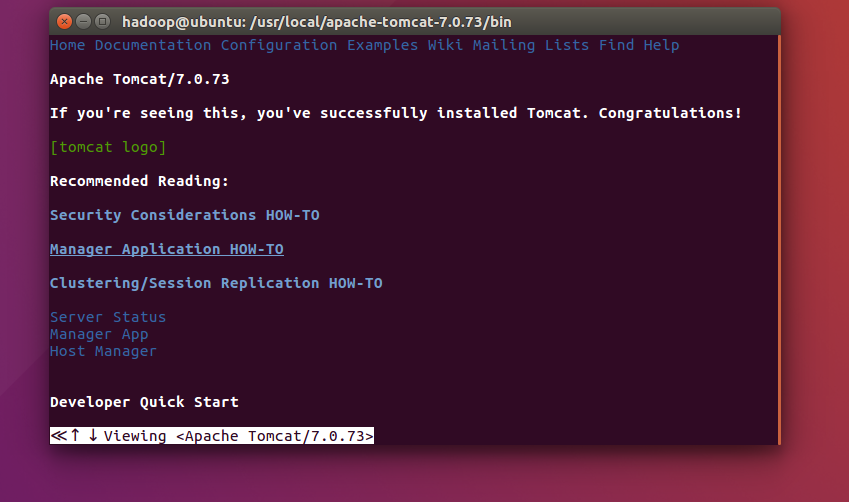
localhost:8080/SpringMVCLearn

tail -f logs/catalina.out

# 浏览器命令

安装w3m





安装字符终端下的浏览器，比如lynx 更好用，有前进后退



