注:

NoteDate:2017-03-24

System:centos7

Reference:《鸟叔的私房菜》

# (一)防火墙firewalld的使用

1、firewalld的基本使用

启动： systemctl start firewalld

查看状态： systemctl status firewalld

停止： systemctl disable firewalld

禁用： systemctl stop firewalld

2.systemctl是CentOS7的服务管理工具中主要的工具，它融合之前service和chkconfig的功能于一体。

启动一个服务：systemctl start firewalld.service  
关闭一个服务：systemctl stop firewalld.service  
重启一个服务：systemctl restart firewalld.service  
显示一个服务的状态：systemctl status firewalld.service  
在开机时启用一个服务：systemctl enable firewalld.service  
在开机时禁用一个服务：systemctl disable firewalld.service  
查看服务是否开机启动：systemctl is-enabled firewalld.service  
查看已启动的服务列表：systemctl list-unit-files|grep enabled  
查看启动失败的服务列表：systemctl --failed

3.配置firewalld-cmd

查看版本： firewall-cmd --version

查看帮助： firewall-cmd --help

显示状态： firewall-cmd --state

查看所有打开的端口： firewall-cmd --zone=public --list-ports

更新防火墙规则： firewall-cmd --reload

查看区域信息:  firewall-cmd --get-active-zones

查看指定接口所属区域： firewall-cmd --get-zone-of-interface=eth0

拒绝所有包：firewall-cmd --panic-on

取消拒绝状态： firewall-cmd --panic-off

查看是否拒绝： firewall-cmd --query-panic

那怎么开启一个端口呢

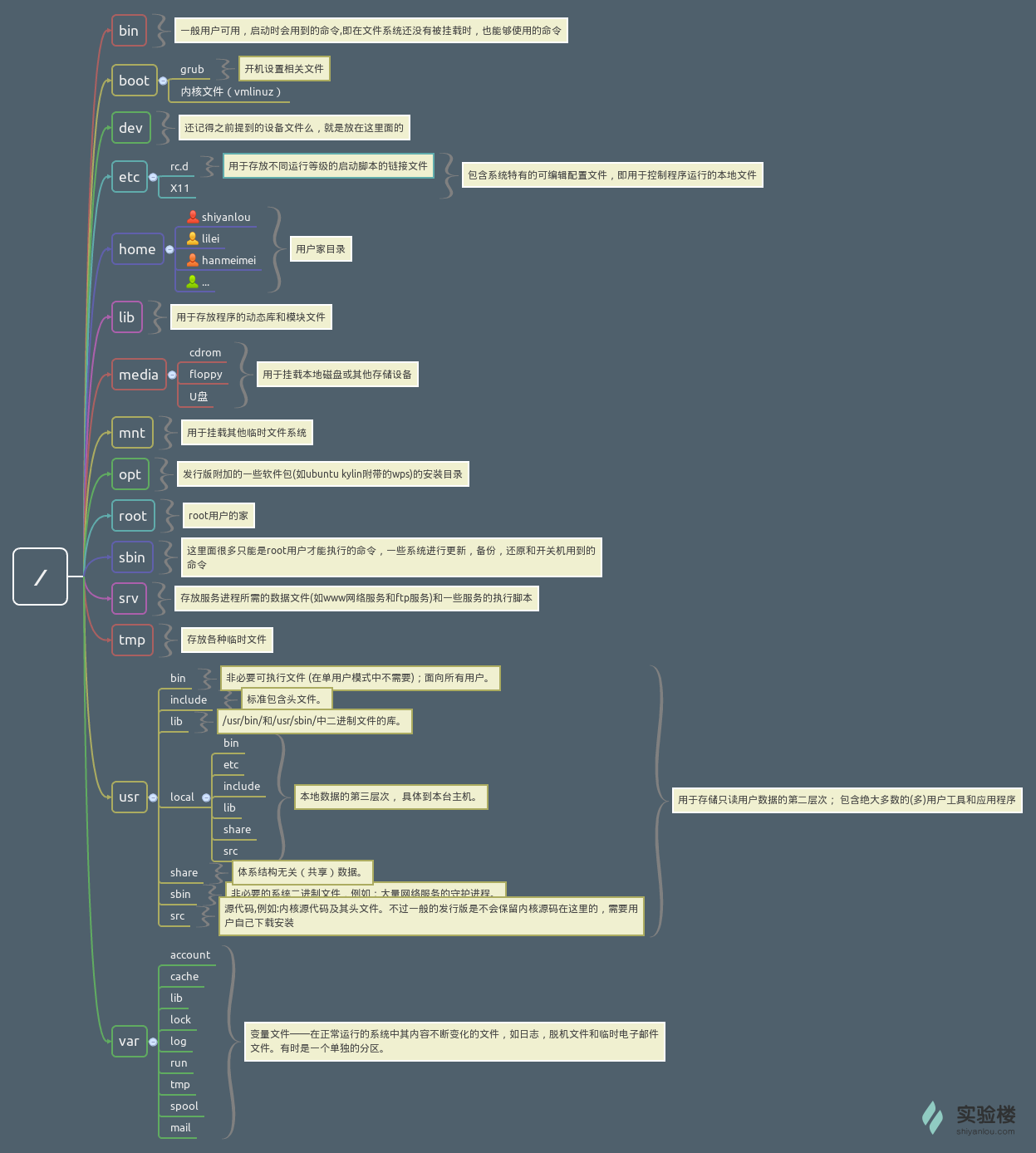
添加firewall-cmd --zone=public --add-port=80/tcp --permanent （--permanent永久生效，没有重启后失效）

重新载入firewall-cmd --reload

查看firewall-cmd --zone= public --query-port=80/tcp

删除firewall-cmd --zone= public --remove-port=80/tcp –permanent

# (二)Linux-FHS



# (三)Shell Scripts

## 1.Shell Scripts的运行方式:

1. sh ./filename.sh
2. bash ./ filename.sh
3. source ./ filename.sh
4. exec ./filename.sh
5. ./ filename.sh

sh和bash的执行方法是在当前的shell下创建新的子进程来运行对应的Scipts.

Source和./的运行方式的一样的不过的是source不需要文件的执行权限

Source和exec都是在当前的shell下运行Scripts,不同的是exec命令在执行时会把当前的shell process关闭,然后换到后面的命令继续执行

例子:

#!/bin/bash

# Program:

# My study project

# PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin:~/bin

# export PATH

##########################例#################################

echo -e "My test!"

read -p "Please input your first name: " firstname # 提示使用者输入

read -p "Please input your last name: " lastname # 提示使用者输入

echo -e "Your full name is: $firstname $lastname" # 结果由屏幕输出

##########################例#################################

echo -e "Use touch command to create different file by date"

read -p "Please input your last name: " filename # 提示使用者输入

date1=$(date --date='1 days ago' +%Y%m%d)

date2=$(date +%Y%m%d)

touch "$filename$date1"

touch "$filename$date2"

exit 0 #说明这个脚本离开的时候回传一个0

# (四)Linux的账户管理

## 1.账号管理相关文件

1. /etc/passwd :保存系统的用户信息

2. /etc/shadow :保存系统中用户的密码

3. /etc/group :保存系统种群组的信息

## 2.账号管理相关指令

1.useradd:添加用户

2.passwd:修改用户密码.如果命令后面不加账户名则为当前账号

3.chage:这个命令个跟passwd差不多.

4.usermod:修改账户细部信息

5.userdel:删除用户

## 3.组管理相关指令

1.groupadd

2.groupmod

3.groupdel

4.gpasswd

# (五)进程管理与系统服务

## 1.PS和top指令.

两个指令都能够观察当前系统中正在运行的进程,其中top指令能够动态观察进程的变化情况

## 2.pstree指令

这个指令可以找出进程之间的相关性

## 3.netstat

这个指令常用于网络的监控方面,其中分为两部分,分别是网络与系统自己的程序相关性部分.

# (六)网络配置

## 1.输入指令查看设备

ifconfig查看当前系统识别的网卡设备,以CentonOS7为例eno16777736

## 2.修改配置文件:

vim /etc/sysconfig/network-scripts/ifcfg-eno16777736编辑网络设置

## 3.设置配置文件:

TYPE=Ethernet

BOOTPROTO=static //将“DHCP”改为“static”.如果是DHCP则为自动获取IP

DEFROUTE=yes

IPV4\_FAILURE\_FATAL=no

IPV6INIT=yes

IPV6\_AUTOCONF=yes

IPV6\_DEFROUTE=yes

IPV6\_FAILURE\_FATAL=no

NAME=eth0

UUID=2e09b0f4-ec1a-44b0-8066-7da7ce04ff02

ONBOOT=yes //将“no”改为“yes”,设置为启动设备

HWADDR=00:0C:29:46:8C:C0

PEERDNS=yes

PEERROUTES=yes

IPV6\_PEERDNS=yes

IPV6\_PEERROUTES=yes

IPADDR=192.168.28.101 //设置IP地址

NETMASK=255.255.255.0 //增加子网掩码

GATEWAY=192.168.1.1 //增加网关

DNS1=8.8.8.8 //增加DNS

## 4.重启网络

输入指令: service network restart

# (七)daemons

## 1.daemons的分类

Daemons有两类:stand alone和super daemons.

Stand alone:这种是可以自行单独启动服务,这种类型不必透过其他几只来管理,daemon启动并加载到内存后就一直占用内存与系统资源.

Super daemon:这种服务的启动方式统一由super daemon来管理.当没有客户端的要求是,各项服务都是未启动的情况,等有来自客户端的要求是,super daemon才唤醒相对应的服务.当客户端的要求结束后,被唤醒的这个服务也会关闭并释放系统资源.

## 2.daemons的启动脚本与启动方式

通常服务的启动脚本放置在/etc/init.d/\*

各项服务的初始化环境配置文件放置在:/etc/sysconfig/\*

Super daemon配置文件放置在:/etc/xinetd.conf,/etc/xinetd.d/\*

## 3.设定开机服务启动

可以通过ntsysv 指令来设置开机启动的服务

Linux开机流程:

1. 打开计算机电源，开始读取 BIOS 并迚行主机的自我测试；

2. 透过 BIOS 取得第一个可开机装置，读取主要开机区 (MBR) 取得开机管理程序；

3. 透过开机管理程序的讴定，取得 kernel 并加载内存丏侦测系统硬件；

4. 核心主劢呼叫 init 程序；

5. init 程序开始执行系统初始化 (/etc/rc.d/rc.sysinit)

6. 依据 init 的讴定迚行 daemon start (/etc/rc.d/rc[0-6].d/\*)

7. 加载本机讴定 (/etc/rc.d/rc.local)

从上面的开机流程可以看出,我们进入linux系统的时候可以进入不同的模式,这个模式可以称为执行等级(run level).不同的执行等级有不同的功能与服务,目前知道的执行等级有两个,一个是X-window的是level 5,另一个则是纯文本界面的level3.

## 4.设定自己的系统服务

## 5. systemd和systemctl

1.systemd是linux启动的第一个进程.pid为1.

2.systemctl

# (八)VIM指令

## 1.VIM的模式:

正常模式（按Esc或Ctrl+[进入） 左下角显示文件名或为空  
插入模式（按i键进入） 左下角显示--INSERT--  
可视模式（不知道如何进入） 左下角显示--VISUAL—

## 2.VIM退出(只能在正常模式退出)

:wq 保存并退出

ZZ 保存并退出

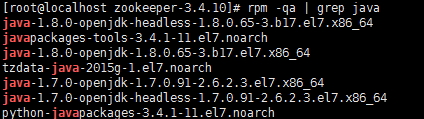
:q! 强制退出并忽略所有更改

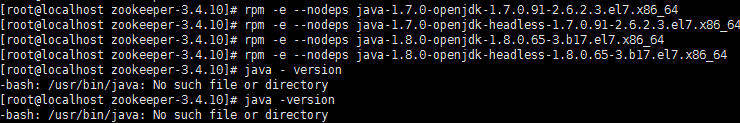
:e! 放弃所有修改，并打开原来文件。

# (九)环境变量设置

例:配置JAVA

1. 删除CentOS自带的OpenJDK

使用rpm -qa | grep java 查看安装位置

1. rpm -e –nodeps或者yum -y remove指令删除全部，noarch文件可以不用删除

rpm -e --nodeps java-1.7.0-openjdk-1.7.0.91-2.6.2.3.el7.x86\_64

rpm -e --nodeps java-1.7.0-openjdk-headless-1.7.0.91-2.6.2.3.el7.x86\_64

rpm -e --nodeps java-1.8.0-openjdk-1.8.0.65-3.b17.el7.x86\_64

rpm -e --nodeps java-1.8.0-openjdk-headless-1.8.0.65-3.b17.el7.x86\_64

1. 修改/etc/profile文件或者.bash\_profile文件,其中/etc/profile是全局的环境变量,对所有用户生效,而. bash\_profile是只对当前用户生效

Vim /etc/profile 并且添加下面语句

#add JDK path

export PATH=$PATH:/opt/jdk1.8.0\_131/bin

#add ActiveMQ path

export PATH=$PATH:/opt/apache-activemq-5.14.5/bin

#add Zookeeper path

export PATH=$PATH:/opt/zookeeper-3.4.10/bin

1. source /etc/profile 命令对配置进行更新

# (十)指令总结

## Unix命令行程序和内建指令(更多)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 文件系统 | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  |  |  | | --- | --- | --- | --- | | ▪ [cat](http://baike.baidu.com/view/250626.htm) | ▪ [cd](http://baike.baidu.com/view/22623.htm) | ▪ [chmod](http://baike.baidu.com/view/1229012.htm) | ▪ [chown](http://baike.baidu.com/view/1229019.htm) | | ▪ [chgrp](http://baike.baidu.com/view/1188586.htm) | ▪ [cksum](http://baike.baidu.com/view/2761036.htm) | ▪ [cmp](http://baike.baidu.com/view/631858.htm) | ▪ [cp](http://baike.baidu.com/view/25561.htm) | | ▪ [du](http://baike.baidu.com/view/43913.htm) | ▪ [df](http://baike.baidu.com/view/111942.htm) | ▪ [fsck](http://baike.baidu.com/view/1757895.htm) | ▪ [fuser](http://baike.baidu.com/view/3123051.htm) | | ▪ [ln](http://baike.baidu.com/view/1156839.htm) | ▪ [ls](http://baike.baidu.com/view/370184.htm) | ▪ [lsattr](http://baike.baidu.com/view/4811972.htm) | ▪ [lsof](http://baike.baidu.com/searchword/?word=lsof&pic=1&sug=1&enc=utf8) | | ▪ [mkdir](http://baike.baidu.com/view/1279338.htm) | ▪ [mount](http://baike.baidu.com/view/1374830.htm) | ▪ [mv](http://baike.baidu.com/view/17861.htm) | ▪ [pwd](http://baike.baidu.com/view/920712.htm) | | ▪ [rm](http://baike.baidu.com/view/6779.htm) | ▪ [rmdir](http://baike.baidu.com/view/1081230.htm) | ▪ [split](http://baike.baidu.com/view/356808.htm) | ▪ [touch](http://baike.baidu.com/view/345328.htm) | | ▪ [umask](http://baike.baidu.com/view/1867757.htm) |  |  |  | |  |  |  |  | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 程序 | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  |  |  | | --- | --- | --- | --- | | ▪ [at](http://baike.baidu.com/view/27835.htm) | ▪ [bg](http://baike.baidu.com/view/194376.htm) | ▪ [chroot](http://baike.baidu.com/view/2977396.htm) | ▪ [cron](http://baike.baidu.com/view/660015.htm) | | ▪ [exit](http://baike.baidu.com/view/619980.htm) | ▪ [fg](http://baike.baidu.com/view/228000.htm) | ▪ [jobs](http://baike.baidu.com/view/1374855.htm) | ▪ [kill](http://baike.baidu.com/view/22085.htm) | | ▪ [killall](http://baike.baidu.com/searchword/?word=killall&pic=1&sug=1&enc=utf8) | ▪ [nice](http://baike.baidu.com/view/620110.htm) | ▪ [pgrep](http://baike.baidu.com/searchword/?word=pgrep&pic=1&sug=1&enc=utf8) | ▪ [pidof](http://baike.baidu.com/searchword/?word=pidof&pic=1&sug=1&enc=utf8) | | ▪ [pkill](http://baike.baidu.com/view/6624478.htm) | ▪ [ps](http://baike.baidu.com/view/7556.htm) | ▪ [pstree](http://baike.baidu.com/view/1229127.htm) | ▪ [sleep](http://baike.baidu.com/view/66232.htm) | | ▪ [time](http://baike.baidu.com/view/10368.htm) | ▪ [top](http://baike.baidu.com/view/184937.htm) | ▪ [wait](http://baike.baidu.com/view/11068.htm) |  | |  |  |  |  | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 使用环境 | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  |  |  | | --- | --- | --- | --- | | ▪ [env](http://baike.baidu.com/view/6397758.htm) | ▪ [finger](http://baike.baidu.com/view/137128.htm) | ▪ [id](http://baike.baidu.com/view/371.htm) | ▪ [logname](http://baike.baidu.com/searchword/?word=logname&pic=1&sug=1&enc=utf8) | | ▪ [mesg](http://baike.baidu.com/view/1229115.htm) | ▪ [passwd](http://baike.baidu.com/view/1116984.htm) | ▪ [su](http://baike.baidu.com/view/843034.htm) | ▪ [sudo](http://baike.baidu.com/view/1138183.htm) | | ▪ [uptime](http://baike.baidu.com/view/1228993.htm) | ▪ [w](http://baike.baidu.com/view/59877.htm) | ▪ [wall](http://baike.baidu.com/view/126810.htm) | ▪ [who](http://baike.baidu.com/view/52104.htm) | | ▪ [whoami](http://baike.baidu.com/view/1116259.htm) | ▪ [write](http://baike.baidu.com/view/1081203.htm) |  |  | |  |  |  |  | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 文字编辑 | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  |  |  | | --- | --- | --- | --- | | ▪ [awk](http://baike.baidu.com/view/209681.htm) | ▪ [comm](http://baike.baidu.com/view/1374860.htm) | ▪ [cut](http://baike.baidu.com/view/443540.htm) | ▪ [ed](http://baike.baidu.com/view/16488.htm) | | ▪ [ex](http://baike.baidu.com/view/345491.htm) | ▪ [fmt](http://baike.baidu.com/view/2131324.htm) | ▪ [head](http://baike.baidu.com/view/463665.htm) | ▪ [iconv](http://baike.baidu.com/view/1437030.htm) | | ▪ [join](http://baike.baidu.com/view/598388.htm) | ▪ [less](http://baike.baidu.com/view/1229034.htm) | ▪ [more](http://baike.baidu.com/view/29762.htm) | ▪ [paste](http://baike.baidu.com/view/1374816.htm) | | ▪ [sed](http://baike.baidu.com/view/432091.htm) | ▪ [sort](http://baike.baidu.com/view/1374853.htm) | ▪ [strings](http://baike.baidu.com/view/7923.htm) | ▪ [talk](http://baike.baidu.com/view/112883.htm) | | ▪ [tac](http://baike.baidu.com/view/754049.htm) | ▪ [tail](http://baike.baidu.com/view/1053222.htm) | ▪ [tr](http://baike.baidu.com/view/316588.htm) | ▪ [uniq](http://baike.baidu.com/view/1374849.htm) | | ▪ [vi](http://baike.baidu.com/view/27682.htm) | ▪ [wc](http://baike.baidu.com/view/24408.htm) | ▪ [xargs](http://baike.baidu.com/view/2285387.htm) |  | |  |  |  |  | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Shell 程序 | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  |  |  | | --- | --- | --- | --- | | ▪ [alias](http://baike.baidu.com/view/49616.htm) | ▪ [basename](http://baike.baidu.com/view/2978587.htm) | ▪ [dirname](http://baike.baidu.com/view/2978588.htm) | ▪ [echo](http://baike.baidu.com/view/32757.htm) | | ▪ [expr](http://baike.baidu.com/view/1229144.htm) | ▪ [false](http://baike.baidu.com/view/2152638.htm) | ▪ [printf](http://baike.baidu.com/view/410546.htm) | ▪ [test](http://baike.baidu.com/view/1234.htm) | | ▪ [true](http://baike.baidu.com/view/743997.htm) | ▪ [unset](http://baike.baidu.com/view/4879915.htm) |  |  | |  |  |  |  | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 网络 | |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  |  |  | | --- | --- | --- | --- | | ▪ [inetd](http://baike.baidu.com/view/500836.htm) | ▪ [netstat](http://baike.baidu.com/view/28008.htm) | ▪ [ping](http://baike.baidu.com/view/709.htm) | ▪ [rlogin](http://baike.baidu.com/view/611069.htm) | | ▪ **netcat** | ▪ [traceroute](http://baike.baidu.com/view/70478.htm) |  |  | |  |  |  |  | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 搜索 | |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  |  |  | | --- | --- | --- | --- | | ▪ [find](http://baike.baidu.com/view/801452.htm) | ▪ [grep](http://baike.baidu.com/view/1057278.htm) | ▪ [locate](http://baike.baidu.com/view/1229038.htm) | ▪ [whereis](http://baike.baidu.com/view/1793607.htm) | | ▪ [which](http://baike.baidu.com/view/1793602.htm) |  |  |  | |  |  |  |  | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 杂项 | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  |  |  | | --- | --- | --- | --- | | ▪ [apropos](http://baike.baidu.com/searchword/?word=apropos&pic=1&sug=1&enc=utf8) | ▪ [banner](http://baike.baidu.com/view/53292.htm) | ▪ [bc](http://baike.baidu.com/view/104163.htm) | ▪ [cal](http://baike.baidu.com/view/56183.htm) | | ▪ [clear](http://baike.baidu.com/view/611255.htm) | ▪ [date](http://baike.baidu.com/view/555251.htm) | ▪ [dd](http://baike.baidu.com/view/21965.htm) | ▪ [file](http://baike.baidu.com/view/32758.htm) | | ▪ [help](http://baike.baidu.com/view/310222.htm) | ▪ [info](http://baike.baidu.com/view/105555.htm) | ▪ [size](http://baike.baidu.com/view/126727.htm) | ▪ [lp](http://baike.baidu.com/view/136896.htm) | | ▪ [man](http://baike.baidu.com/view/2671.htm) | ▪ [history](http://baike.baidu.com/view/325910.htm) | ▪ [tee](http://baike.baidu.com/view/912941.htm) | ▪ [tput](http://baike.baidu.com/searchword/?word=tput&pic=1&sug=1&enc=utf8) | | ▪ [type](http://baike.baidu.com/view/360307.htm) | ▪ [yes](http://baike.baidu.com/view/74673.htm) | ▪ [uname](http://baike.baidu.com/view/1374878.htm) | ▪ [whatis](http://baike.baidu.com/view/2129683.htm) | | |

## 1. 文件权限

$ls   -l

**1 .文件类型**  
文件权限位前面的那个字符,我们例子中的横杠就是文件的类型，即普通文件类型。  
文件类型有七种，它可以从 ls -l命令所列出的结果的第一位看出:  
d   目录。l  符号链接(指向另一个文件)。s  套接字文件。b  块设备文件。c  字符设备文件。p  命名管道文件。  
-    普通文件，或者更准确地说，不属于以上几种类型的文件。

**2.文件的权限：**

文件的权限可分为三类：  
1) 文件属主，创建该文件的用户。如 -rw：文件属主权限 这是前面三位  
2) 同组用户，拥有该文件的用户组中的任何用户。如 -r-： 同组用户权限 这是中间三位  
3) 其他用户，即不属于拥有该文件的用户组的某一用户。如 -r-：其他用户权限 这是最后三位

**3.修改权限: chmod**

1）符合模式：chmod [who]  operator [permission]  filename

who的含义是：**u**文件属主权限。**g**同组用户权限。**o**其他用户权限。**a**所有用户(文件属主、同组用户及其他用户)。  
operator的含义：**+** 增加权限。**-**取消权限。**=** 设定权限。  
permission的含义：**r**读权限。**w**写权限。**x**执行权限。**s** 文件属主和组set -ID。t 粘性位\*。**l**给文件加锁，使其他用户无法访问。

u,g,o   针对文件属主、同组用户及其他用户的操作。

chmod u+x myfile  //赋予文件属主执行权限

2) 绝对模式:  chmod  [mode]   file  
其中mode是一个八进制数:  
0**4**00 文件属主可读        00**4**0 同组用户可读          000**4** 其他用户可读  
0**2**00 文件属主可写        00**2**0 同组用户可写          000**2** 其他用户可写  
0**1**00 文件属主可执行    00**1**0 同组用户可执行       000**1** 其他用户可执行  
myfile文件具有这样的权限：  
  rw-         r--         r--  
  4+2        4         4

把相应权限位所对应的值加在一起，就是644。

**4.chown和chgrp**

chown命令的一般形式为：  
chown -R -h owner file  
- R选项意味着对所有子目录下的文件也都进行同样的操作。

 - h选项意味着在改变符号链接文件的属主时不影响该链接所指向的目标文件。

如：

$ chown [**MySQL**](http://lib.csdn.net/base/mysql) hiscore 把 hiscore文件的属主改为[**mysql**](http://lib.csdn.net/base/mysql)  
$ chown -R mysql hiscore 递归地把hiscore（[**数据库**](http://lib.csdn.net/base/mysql)名称）目录下的所有文件属主改成mysql

chgrp命令和chown命令的格式差不多

chgrp sybadmin project

把project文件所属的组由admin变为sybadmin（系统中的另外一个用户组） 。

## 2. ls、find和xargs

ls使用技巧

以找到当前目录中最大的那个文件：

ls -lSrh

“r”的作用是将大的文件列在后面，而“h”则是给出易于人们阅读的输出(MB或者诸如此类)。你也可以搜寻最大的MP3/MPEG文件：

ls -lSrh \*.mp\*

你也可以通过下面这条命令搜寻最大的目录：

du -kx | egrep -v "\./.+/" | sort -n

Find命令的一般形式为：

find pathname -options [-print -exec -ok]  
让我们来看看该命令的参数：  
    pathname   find命令所查找的目录路径。例如用.来表示当前目录，用/来表示系统根目录。  
   -options      为命令选项  
   -print           find命令将匹配的文件输出到标准输出。  
   -exec         find命令对匹配的文件执行该参数所给出的 shell命令。相应命令的形式为 ' comm -and' {} \;，注意{ }和\；之间的空格。  
   -ok   和-exec的作用相同，只不过以一种更为安全的模式来执行该参数所给出的 shell命令，在执行每一个命令之前，都会给出提示，让用户来确定是否执行。

**1.  find命令选项**

   -name   按照文件名查找文件。  
   -perm   按照文件权限来查找文件。  
   -prune   使用这一选项可以使find命令不在当前指定的目录中查找，                                                                                                                                  如果同时使用了-depth选项，那么-prune选项将被find命令忽略。  
   -user   按照文件属主来查找文件。  
   -group   按照文件所属的组来查找文件。  
   -mtime   -n +n   按照文件的更改时间来查找文件，  
                - n表示文件更改时间距现在n天以内，+ n表示文件更改时间距现在n天以前。   
                find命令还有- atime和-ctime选项，但它们都和-mtime选项相似，所以我们在这里只介绍- mtime选项。  
   -nogroup   查找无有效所属组的文件，即该文件所属的组在 /etc/groups中不存在。  
   -nouser   查找无有效属主的文件，即该文件的属主在 /etc/passwd中不存在。  
   -newer file1 ! file2   查找更改时间比文件file1新但比文件file2旧的文件。  
   -type   查找某一类型的文件，诸如：  
   b - 块设备文件。  
   d - 目录。  
   c - 字符设备文件。  
   p - 管道文件。  
   l - 符号链接文件。  
   f - 普通文件。  
  -size n[c]   查找文件长度为n块的文件，带有c时表示文件长度以字节计。  
  -depth   在查找文件时，首先查找当前目录中的文件，然后再在其子目录中查找。  
  -fstype   查找位于某一类型文件系统中的文件，这些文件系统类型通常可以在配置文件/etcfstab中找到，该配置文件中包含了本系统中有关文件系统的信息。  
   -mount   在查找文件时不跨越文件系统mount点。  
   -follow   如果find命令遇到符号链接文件，就跟踪至链接所指向的文件。  
   -cpio   对匹配的文件使用cpio命令，将这些文件备份到磁带设备中。

2. find的例子：

1)匹配$ H O M E目录下的所有文件，下面两种方法都可以使用：  
$ find $HOME -print  
$ find ~ -print  
2) 在当前目录中查找suid置位，文件属主具有读、写、执行权限，并且文件所属组的用户和其他用户具有读和执行的权限的文件，可以用：  
$ find . -type f -perm 4755 -print  
3 )查找系统中所有文件长度为0的普通文件，并列出它们的完整路径，可以用：

        $ find / -type f -size 0 -exec ls -l {} \;

可以按照文件长度来查找文件，这里所指的文件长度既可以用块（block）来计量，也可以用字节来计量。以字节计量文件长度的表达形式为N c；以块计量文件长度只用数字表示即可。

在按照文件长度查找文件时，一般使用这种以字节表示的文件长度，在查看文件系统的大小，因为这时使用块来计量更容易转换。

**实例1：在当前目录下查找文件长度大于1 M字节的文件**

**命令：**

find . -size +1000000c -print

**实例2：在/home/apache目录下查找文件长度恰好为100字节的文件:**

**命令：**

find /home/apache -size 100c -print

**实例3：在当前目录下查找长度超过10块的文件（一块等于512字节）**

**命令：**

find . -size +10 -print

4) 查找/var/logs目录中更改时间在7日以前的普通文件，并删除它们，可以用：  
$ find /var/logs -type f -mtime +7 -exec rm {} \;  
5) 查找系统中所有属于audit组的文件，可以用：  
$find /-name -group audit -print  
6) 我们的一个审计系统每天创建一个审计日志文件。日志文件名的最后含有数字，这样我们一眼就可以看出哪个文件是最新的，哪个是最旧的。 admin.log文件编上了序号：  
admin.log.001、admin.log.002等等。下面的find命令将删除/logs目录中访问时间在7日以前、含有数字后缀的admin.log文件。该命令只检查三位数字，所以相应日志文件的后缀不要超过999。  
$ find /logs -name 'admin.log[0-9][0-9][0-9] '-atime +7 -exec rm {} \;  
7) 查找当前文件系统中的所有目录并排序，可以用：  
$ find . -type d -print -local -mount |sort  
8) 查找系统中所有的rmt磁带设备，可以用：  
$ find /dev/rmt -print

**9）从根目录开始查找所有扩展名为.log的文本文件，并找出包含”ERROR”的行：**

**$ find / -type f -name "\*.log" | xargs grep "ERROR"**

**10、使用find在多个文件中替换掉相同的文本**

要替换当前目录以及下层目录里所有文件中的Windows为[**Linux**](http://lib.csdn.net/base/linux)，你可以这样运行：

find . -name '\*.txt' -print | xargs perl -pi -e's/Windows/[**linux**](http://lib.csdn.net/base/linux)/ig' \*.txt

或者如果你更需要让它只作用于普通文件上

find -type f -name '\*.txt' -print0 | xargs --null perl -pi -e 's/Windows/Linux/'

## 3. 后台执行命令

### 1. crontab

1). crontab的域

是crontab的格式：  
分  时  日  月  星期 要运行的命令  
第1列 分钟1～59  
第2列 小时1～23（0表示子夜）  
第3列 日1～31  
第4列 月1～12  
第5列 星期0～6（0表示星期天）  
第6列 要运行的命令  
crontab文件的一个条目是从左边读起的，第一列是分，最后一列是要运行的命令，它位于星期的后面。  
用横杠-来表示一个时间范围，例如你希望星期一至星期五运行某个作业，那么可以在星期域使用1-5来表示。  
使用逗号“,” ，例如你希望星期一和星期四运行某个作业，只需要使用 1,4来表示。  
用星号 \*来表示连续的时间段。如果你对某个表示时间的域没有特别的限定，也应该在该域填入 \*。  
该文件的每一个条目必须含有 5个时间域，而且每个域之间要用空格分隔。该文件中所有的注释行要在行首用#来表示。

例子：

分 时 日 月 星期 要运行的命令  
30 21 \*   \*  \*   /apps/bin/cleanup.sh :表示每晚的21:30运行/apps/bin目录下的cleanup.sh。  
45 4  1,10,22 \* \* /apps/bin/backup.sh :表示每月1、10、22日的4:45运行/apps/b i n目录下的backup.sh。  
10 1 \* \* 6,0 /bin/find -name "core" -exec rm {} \; :表示每周六、周日的1 : 1 0运行一个f i n d命令。  
0,30 18-23 \* \* \* /apps/bin/dbcheck.sh:表示在每天18:00至23:00之间每隔30分钟运行/apps/bin目录下的dbcheck.sh。  
0 23 \* \* 6 /apps/bin/qtrend.sh :表示每星期六的11:00pm运行/apps/bin目录下的qtrend.sh。

每隔30分钟时运行:

\*/30 \* \*   \*  \*   /apps/bin/cleanup.sh :

**2) 创建和提交crontab文件：**  
创建一个名为davecron。的文件,加入如下的内容:  
#15 minutes between 6pm and 6am  
0,15,30,45 18-06 \* \* \*  /bin/echo `date` > /dev/console  
系统将每隔 15分钟向控制台输出一次当前时间。  
如果系统崩溃或挂起，从最后所显示的时间就可以一眼看出系统是什么时间停止工作的。  
提交crontab文件，cron命令的参数：  
$ crontab davecron  
现在该文件已经提交给cron进程，它将每隔1 5分钟运行一次。  
同时，新创建文件的一个副本已经被放在/var/spool/cron目录中，文件名就是用户名davecron。

**2) 常用参数:**

crontab   -l   //查看当前用户下的cron任务  
crontab -e  //编辑当前用户的定时任务  
crontab -u  linuxso  -e  //编辑用户linuxso的定时任务  
crontab -r   //删除crontab文件  
注意：编辑crontab文件，最好先选择好编辑器，选vim（3）：

update-alternatives --config editor

There are 3 choices for the alternative editor (providing /usr/bin/editor).

  Selection    Path               Priority   Status

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

  0            /bin/nano           40        auto mode

  1            /bin/ed            -100       manual mode

  2            /bin/nano           40        manual mode

  3            /usr/bin/vim.tiny   10        manual mode

**3) $HOME目录中对crontab文件做一备份：**  
$crontab -l > $HOME/mycron  
**4) 恢复丢失的crontab文件：**  
如果不小心误删了crontab文件，假设你在自己的$HOME目录下还有一个备份，那么可以将其拷贝到/var/spool/cron/<username>，其中<username>是用户名。如果由于权限问题无法完  
成拷贝，可以用：  
$ crontab <filename>  
其中，<filename>是你在$HOME目录中副本的文件名。  
我建议你在自己的$HOME目录中保存一个该文件的副本。我就有过类似的经历，有数次误删了crontab 文件（因为r键紧挨在e键的右边…） 。这就是为什么有些系统文档建议不要直接  
编辑crontab 文件，而是编辑该文件的一个副本，然后重新提交新的文件。  
有些crontab 的变体有些怪异，所以在使用crontab 命令时要格外小心。如果遗漏了任何选  
项，crontab 可能会打开一个空文件，或者看起来像是个空文件。这时敲crontab 键退出，不要按  
< Ctrl - D >，否则你将丢失crontab 文件。

查看crontab服务状态：service crond status

手动启动crontab服务：service crond start

### 2. at命令提交命令或者shell脚本

**1）at命令使用**：  
at命令允许用户向cron守护进程提交作业，使其在稍后的时间运行。这里稍后的时间可能是指10min以后，也可能是指几天以后。如果你希望在一个月或更长的时间以后运行，最好还是使用crontab文件。一旦一个作业被提交， at命令将会保留所有当前的环境变量，包括路径，不象crontab，只提供缺省的环境。该作业的所有输出都将以电子邮件的形式发送给用户，除非你对其输出进行了重定向，绝大多数情况下是重定向到某个文件中。和crontab一样，根用户可以通过/etc目录下的at.allow和at.deny文件来控制哪些用户可以使用at命令，哪些用户不行。不过一般来说，对at命令的使用不如对crontab的使用限制那么严格。  
**at命令的基本形式为：**  
at [-f script] [-m -l -r] [time] [date]  
其中，  
-f script 是所要提交的脚本或命令。  
-l 列出当前所有等待运行的作业。at q命令具有相同的作用。  
-r 清除作业。为了清除某个作业，还要提供相应的作业标识（ID）；有些UNIX变体只接受atrm作为清除命令。  
-m 作业完成后给用户发邮件。  
time at命令的时间格式非常灵活:  
1) 可以是H、HH.HH MM、HH : MM或H:M，其中H和M分别是小时和分钟。还可以使用a.m.或p.m.。  
2)date日期格式可以是月份数或日期数，而且at命令还能够识别诸如today、tomorrow这样的词。现在就让我们来看看如何提交作业。  
  
**2）使用at命令提交命令或脚本**  
使用at命令提交作业有几种不同的形式，可以通过命令行方式，也可以使用at命令提示符。  
一般来说在提交若干行的系统命令时，我使用at命令提示符方式，而在提交s h e l l脚本时，使用命令行方式。  
如果你想提交若干行的命令，可以在at命令后面跟上日期/时间并回车。然后就进入了at命令提示符，这时只需逐条输入相应的命令，然后按‘ <CTRL - D>’退出。下面给出一个例子：  
root@ubuntu:/home/hadoop/testsh# at 16:10  
warning: commands will be executed using /bin/sh  
at> find /etc -name "passwd" -print  
at> <EOT>  
at> job 1 at 2012-10-22 16:10       
其中，   
<EOT>就是<CTRL - D>。在16:10  系统将执行一个简单的find命令。  
你应当已经注意到，我所提交的作业被分配了一个唯一标识job 1。该命令在完成以后会将全部结果以邮件的形式发送给我。  
如果希望向at命令提交一个shell脚本，使用其命令行方式即可。在提交脚本时使用- f选项。  
$ at 3.00pm tomorrow -f /apps/bin/db\_table.sh  
warning: commands will be executed using /bin/sh  
job 8 at 2012-10-23 15:00  
在上面的例子中，一个叫做db\_table.sh的脚本将在明天下午3:00运行。  
还可以使用echo命令向at命令提交作业：

$ echo find /etc -name "passwd" -print | at now +1 minute

### 3. shell输入与输出

**1 . echo**  
字符串被重定向到一个名为myfile文件中：  
$ echo "The log files have all been done"> myfile  
**2 . read**  
可以使用read 语句从键盘或文件的某一行文本中读入信息，并将其赋给一个变量。如果只  
指定了一个变量，那么read 将会把所有的输入赋给该变量，直至遇到第一个文件结束符或回  
车。  
它的一般形式为：  
read varible1 varible2 ...  
只指定了一个变量，它将被赋予直至回车之前的所有内容:  
root@ubuntu:/home/testsh# read name  
test1  
root@ubuntu:/home/testsh# echo $name  
test1  
root@ubuntu:/home/testsh#   
**3 . cat**  
显示myfile1、myfile2、myfile3这三个文件，可以用：  
$ cat myfile1 myfile2 myfile3  
**4 . 管道**  
格式：命令1 |命令2

其中|是管道符号。  
sed、awk和grep都很适合用管道，特别是在简单的一行命令中。在下面的例子中， who命令的输出通过管道传递给awk命令，以便只显示用户名和所在的终端。  
root@ubuntu:/home/testsh# who |awk '{print $1 "\t" $2}'  
hgs ptty7  
hgs pts/0  
root@ubuntu:/home/testsh#   
**5. tee**  
它的一般形式为：  
tee -a files  
其中，- a表示追加到文件末尾。  
当执行某些命令或脚本时，如果希望把输出保存下来， tee命令非常方便。  
我们使用 who命令，结果输出到屏幕上，同时保存在 who.out文件中：  
root@ubuntu:/home/testsh# who | tee who.out  
hgs   tty7         2011-10-21 23:26 (:0)  
hgs   pts/0        2011-10-21 23:26 (:0.0)  
root@ubuntu:/home/testsh# ls  
devcont  devcront  who.out  
root@ubuntu:/home/testsh# cat who.out   
hgs   tty7         2011-10-21 23:26 (:0)  
hgs   pts/0     2011-10-21 23:26 (:0.0)  
**6.重定向**  
command > filename 把把标准输出重定向到一个新文件中  
command >> filename 把把标准输出重定向到一个文件中(追加)  
command 1 > fielname 把把标准输出重定向到一个文件中  
command > filename 2>&1 把把标准输出和标准错误一起重定向到一个文件中  
command 2 > filename 把把标准错误重定向到一个文件中  
command 2 >> filename 把把标准输出重定向到一个文件中(追加)  
command >> filename 2>&1 把把标准输出和标准错误一起重定向到一个文件中 (追加)  
command < filename >filename2 把command命令以filename文件作为标准输入，以 filename 2文件作为标准输出  
command < filename 把command命令以filename文件作为标准输入  
command << delimiter 把从标准输入中读入，直至遇到delimiter分界符  
command <&m 把把文件描述符m作为标准输入  
command >&m 把把标准输出重定向到文件描述符m中  
command <&- 把关闭标准输入

### 4. grep功能

grep选项  
常用的grep选项有：  
-c   只输出匹配行的计数。  
-i   不区分大小写（只适用于单字符） 。  
-h   查询多文件时不显示文件名。  
-l   查询多文件时只输出包含匹配字符的文件名。  
-n   显示匹配行及行号。  
-s   不显示不存在或无匹配文本的错误信息。  
-v   显示不包含匹配文本的所有行。

1）如果要在当前目录下所有.doc文件中查找字符串“sort” ，方法：$ grep "sort"\*.doc

2）从文件内容查找匹配指定字符串的行：$ grep "被查找的字符串" 文件名   
3）从文件内容查找与正则表达式匹配的行：$ grep –e “正则表达式” 文件名   
4）查找时不区分大小写：$ grep –i "被查找的字符串" 文件名   
5）查找匹配的行数：$ grep -c "被查找的字符串" 文件名   
6）从文件内容查找不匹配指定字符串的行：$ grep –v "被查找的字符串" 文件名

**7）递归查找目录/etc/apache2下的文件内容“"\/var\/www”**：grep "\/var\/www" -i /etc/apache2 -r

### 5. awk功能

**1. awk简单使用：**  
awk '{print $0}' grade.txt > wow  
**2. awk过滤access.log日志**

#!/bin/sh

#if test -n "$1"; then

#  logfile="/var/log/apache2/access.log"

#else

#  logfile=$1

#fi

#读取log文件：文件内容格式：

#10.64.56.51 - - [22/Oct/2012:11:51:51 +0800] "GET /appdata/www/img/empty.gif HTTP/1.1" 304 260 "http://10.64.49.18/appdata/www/index.php?act=left" "codx=10100.12206.12208.12209; csdx=210910; project=card; left\_action=Action\_MultipleAction; PHPSESSID=p6pp5aumcgvit7ohc25fj5niv5" "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/535.11 (KHTML, like Gecko) Chrome/17.0.963.79 Safari/535.11"

#10.64.56.51 - - [22/Oct/2012:11:51:50 +0800] "GET /appdata/www/index.php?act=mid HTTP/1.1" 200 927 "http://10.64.49.18/appdata/www/index.php" "codx=10100.12206.12208.12209; csdx=210910; project=card; left\_action=Action\_MultipleAction; PHPSESSID=p6pp5aumcgvit7ohc25fj5niv5" "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/535.11 (KHTML, like Gecko) Chrome/17.0.963.79 Safari/535.11"

logfile="/var/log/apache2/access.log"

#项目名称

project=$1

if test -z "{$project}"; then

    echo "no project name"

    exit;

fi

#项目路径

projectDir="/www/$project/crontab"

#datetime=`date -R|awk '{$5=substr($5,0,2)-1 ; print $2"/"$3"/"$4":"$5}'`

#当前时间格式化成和access.log时间格式一样

datetime=`date -R -d last-hour|awk '{$5=substr($5,0,2); print $2"/"$3"/"$4":"$5}'`

datafile=`pwd`"/log/"`date -d last-hour +%Y-%m-%d\_%H`"\_$project.log"

flagFile=`pwd`"/log/"`date -d last-hour +%Y-%m-%d\_%H`"\_$project"

if [ -f $flagFile ];then

    echo "flag exist:" $flagFile

    exit

fi

projectPath="$project/www/index.php"

#功能：把appache的access.log日志根据项目名和日期，按一定格式，输出到指定文件中

#使用awk解析date=$datetime接受外部变量,substr($5,0,14)==date中的变量date不要带$

#-F 分隔符：[][" ]，即使用[][" ]分割每行，没有是使用空格

#project=$projectPath date=$datetime接受外部的变量

#

awk  -F '[][" ]' '{hh=substr($5, index($5,":")+1);ydate=substr(date,0, index(date, ":")-1);if(substr($5,0,14)==date&&$10~project)print $1"\t"ydate"\t"hh"\t"$10 }' project=$projectPath date=$datetime $logfile > $datafile

size=`ls -ls  $datafile| awk '{print $6}'`

if [ $size == 0 ];then

    echo "Empty log"

    rm $datafile

    exit;

else

    echo `pwd`"/log/"`date -d last-hour +%Y-%m-%d\_%H` > $flagFile

fi

phpdir="/www/$project/crontab/data/accessLog.php"

if [ ! -f $phpdir ]; then

     echo "no php file:"$phpdir

     rm $datafile

     exit

fi

if [ ! -f $datafile ]; then

        echo "no datafile:" $datafile

    exit

fi

#php

php $phpdir $datafile

#使用：./cataccess.sh farm

### 6. 查看文件内容：head，tail，sed

相信最基本的cat、more和less你已经很熟悉了，如果有特殊的要求呢：  
1. 如果你只想看文件的前5行，可以使用head命令，如：  
head -5 /etc/passwd  
2. 如果你想查看文件的后10行，可以使用tail命令，如：  
tail -2 /etc/passwd 或 tail -n 2 /etc/passwd  
tail -f /var/log/messages  
参数-f使tail不停地去读最新的内容，这样有实时监视的效果，用Ctrl＋c来终止！  
3. 查看文件中间一段，你可以使用sed命令，如：  
sed -n '5,10p' /etc/passwd  
这样你就可以只查看文件的第5行到第10行。  
  
tail语法格式：  
    tail [ -f ] [ -c Number | -n Number | -m Number | -b Number | -k Number ] [ File ]  
或者  
tail [ -r ] [ -n Number ] [ File ]  
使用说明：  
tail 命令从指定点开始将 File 参数指定的文件写到标准输出。如果没有指定文件，则会使用标准输入。 Number 变量<br />指定将多少单元写入标准输出。 Number 变量的值可以是正的或负的整数。如果值的前面有 +（加号），从文件开头<br />指定的单元数开始将文件写到标准输出。如果值的前面有 -（减号），则从文件末尾指定的单元数开始将文件写到<br />标准输出。如果值前面没有 +（加号）或 -（减号），那么从文件末尾指定的单元号开始读取文件。  
主要参数：  
-b Number 从 Number 变量表示的 512 字节块位置开始读取指定文件。   
-c Number 从 Number 变量表示的字节位置开始读取指定文件。   
-f 如果输入文件是常规文件或如果 File 参数指定 FIFO（先进先出），那么 tail   
命令不会在复制了输入文件的最后的指定单元后终止，而是继续从输入文件读取和复  
制额外的单元（当这些单元可用时）。如果没有指定 File 参数，并且标准输入是管道  
则会忽略 -f 标志。tail -f 命令可用于监视另一个进程正在写入的文件的增长。   
-k Number 从 Number 变量表示的 1KB 块位置开始读取指定文件。   
-m Number 从 Number 变量表示的多字节字符位置开始读取指定文件。使用该标志提供  
在单字节和双字节字符代码集环境中的一致结果。   
-n Number 从首行或末行位置来读取指定文件，位置由 Number 变量的符号（+ 或 - 或无）  
表示，并通过行号 Number 进行位移。   
-r 从文件末尾以逆序方式显示输出。  
-r 标志的缺省值是以逆序方式显示整个文件。  
如果文件大于 20,480 字节，那么-r标志只显示最后的 20,480 字节。   
-r 标志只有与 -n 标志一起时才有效。否则，就会将其忽略

### 7. sed和awk的常用实例

一、文本间隔

1、在每一行后面增加一空行

sed G guo.sh

awk '{printf("%s\n\n",$0 ) }'

2、将文件中原来的空行删掉，并在在每一行后边增加一空行

sed '/^$/d;G '

awk '!/^$/ {printf("%s\n\n",$0 ) }'

3在匹配式样的行前插入一空行

sed '/good/i\\'

sed '/hello/{x;p;x；}'

awk '{ if(/hello/) printf("\n\%s\n",$0);else print $0}'

4、在匹配式样的行后插入一空行

sed '/good/a\\'

sed '/hello/G'

awk '{if(/hello/) printf("%s\n\n",$0) ;else print $0}'

5、在匹配式样的行前、行后各插入一空行

sed '/hello/{x;p;x；G；}'

awk '{ if(/hello/) printf("\n\%s\n\n",$0);else print $0}'

二、文本的替换

1、在每一行查找到good，然后把good替换为bad

sed 's/good/bad/'只把每行的第一个good替换为bad

sed 's/good/bad/2'只把每行的第二个good替换为bad

sed 's/good/bad/g'把每一行的所有good替换为bad

sed 's/.∗good/\1bad/'只把每一行的的最后一个good替换位bad

sed 's/.∗good.∗good/\1bad\2/'只把每一行的的倒数第二个good替换位bad

awk   '{sub(/good/,"bad"); print $0}'只把每行的第一个good替换为bad

awk   '{gsub(/good/,"bad"); print $0}'把每一行的所有good替换为bad

cat  test |tr 'root' 'good'

cat    test |tr 'root' 'god'

sed   's/root/good/p' test

awk   '{gsub(/root/,"good");print $0}' test

awk   '{gsub(/root/,"god");print $0}' test

通过对比可知tr的替换两个字串必须等长度，而sed不能把说有的root替换

2、只在出现字符串fell字符串的前提下，将找到的行中的good替换为bad

sed  '/fell/ s/good/bad/g'

awk   '{if(/fell/) gsub(/good/,"bad"); print $0 }'

3、只在不出现字符串fell字符串的前提下，将找到的行中的good替换为bad

sed '/fell/ !s/good/bad/g'

awk '{if(!/fell/) gsub(/good/,"bad"); print $0 }'

4、不管是good、fell还是sun，一路替换为bad

sed 's/good/bad/g;s/fell/bad/g;s/sun/bad/g'

sed 's/good\|fell\|sun/bad/g'

awk '{gsub(/good|fell|sun/,"bad") ; print $0}'

5、倒置所有行，第一行变为最后一行（模拟tac）

sed '1!G;h;$!d'

sed -n '1!G;h;$p'

awk '{A[i++]=$0} END{for (j=i-1;j>=0;j--) print A[j]}'

6、将每两行连接为一行

sed '$!N;s/\n/ /'

awk '{f=!f;if(!f) printf("%s",$0);else printf("%s\n",$0)}'

7、在文件中每隔5行显示一空行

sed '0~5G'

sed 'n;n;n;n;G'

awk '{print $0 ;i++;if(i==5) {printf( "\n") ;i=0}}'

三、选择性的显示特定行

1、显示文件的前10行

sed10q

awk '{print ;if(NR==10) exit }'

2、显示文件的第一行

sed q

awk '{ print;exit}'

3、显示文件的倒数第二行

sed -e '$!{h;d;}' -e x当文件只有一行时，输出空行

sed -e '1{$q;}' -e '$!{h;d;}' -e x当文件只有一行时，显示该行

sed -e '1{$d;}' -e '$!{h;d;}' -e x当文件只有一行时，不输出

awk '{ B=A;A=$0} END{ print B}'当文件只有一行时，输出空行

4、只显示匹配正则表达式的行

sed -n '/good/p'

sed  '/good/!d'

awk '/good/ {print $0}'

awk '/good/'

grep 'good'

5、只显示不匹配正则表达式的行

sed -n '/good/!p'

sed '/good/d'

awk '!/good/ {print $0}'

grep -v 'good'

6、显示匹配行的上一行，而不显示匹配行

sed -n '/good/{g;1!p;};h'

awk '/good/ {print A} {A=$1}

7、显示匹配行的下一行，而不显示匹配行

sed -n '/good/{n;p;}'

awk '{if(A) print;A=0}/23/{A=1}'

8、显示文本的制定行

sed -n '3,5p'

sed '3,5!d'

awk '{if(NR>=3 &&NR<=5) print } '

目录： 1、不显示文件中的空行 2、删除文件的1到5行3、删除文件注释行4、打印匹配行5、显示从字符1到字符2的中间行6、匹配特别表达式7、替代文本

1、不显示文件中的空行

[guo@guo~]$ grep -v '^$' rc.local

[guo@guo~]$ sed -e '/^$/d' rc.local

[guo@guo~]$ awk '!/^$/{print $0 }' rc.local

2、删除文件的1到5行

[guo@guo~]$ cat test

[guo@guo~]$ sed -e '1,5d' test

[guo@guo~]$ awk '{if(NR>5 ) print $0} ' test

3、删除文件注释行

[guo@guo~]$ sed -e "/^#/d" rc.local

[guo@guo~]$ awk '!/^#/{print $0}' rc.local

[guo@guo~]$ grep -v '^#' rc.local

5、显示从字符1到字符2的中间行

[guo@guo~]$ sed -n -e '/hello/,/sorry/p' test1

[guo@guo~]$ awk '/hello/,/sorry/ {print $0 }' test1

6、匹配特别表达式

[guo@guo~]$ sed -n -e '/main[[:space:]]\*(/,/^}/p' for.c

1、通过awk访问系统环境变量

[guo@guoawk]$ awk 'BEGIN { print ENVIRON["HOME"]; printENVIRON["PATH"]}'

/home/guo

/usr/lib/qt-3.3/bin:/usr/local/bin:/usr/bin:/bin:/usr/local/sbin:/usr/sbin

2、awk中的算术运算符

[guo@guoawk]$ awk 'BEGIN {print "3+2=" 3+2 }'

3+2=5

[guo@guoawk]$ awk 'BEGIN {print "2^10=" 2^10 }'

2^10=1024

[guo@guoawk]$ awk 'BEGIN {print "(3+1)\*4/2=" (3+1)\*4/2}'

(3+1)\*4/2=8

awk支持的运算符有+ - \* / % ^。

4、awk的循环

if语句

[guo@guoawk]$ cat num

12243535

890232

232 9809

890 -8990

[guo@guoawk]$ awk '{if($1>$2) print $0}' num

890232

890 -8990

[guo@guoawk]$ awk '{if($1>$2) {print $0 ; print $1 "is larger than"$2}}' num

890232

890islarger than232

890 -8990

890islarger than-8990

if/else

[guo@guoawk]$ awk '{if($1>$2) print $1 ;else print $2}' num

3535

890

9809

890

[guo@guoawk]$ awk '{if($1>$2) {count++ ;print count} else print $2}' num

3535

1

9809

2

# 在每一行后面增加一空行  
sed G  
  
# 将原来的所有空行删除并在每一行后面增加一空行。  
# 这样在输出的文本中每一行后面将有且只有一空行。  
sed '/^$/d;G'

 # 在每一行后面增加两行空行  
sed 'G;G'  
  
# 将第一个脚本所产生的所有空行删除（即删除所有偶数行）  
sed 'n;d'

# 在匹配式样“regex”的行之前插入一空行  
sed '/regex/{x;p;x;}'  
  
# 在匹配式样“regex”的行之后插入一空行  
sed '/regex/G'  
  
# 在匹配式样“regex”的行之前和之后各插入一空行  
sed '/regex/{x;p;x;G;}'

编号：  
--------  
  
# 为文件中的每一行进行编号（简单的左对齐方式）。这里使用了“制表符”  
# （tab，见本文末尾关于'\t'的用法的描述）而不是空格来对齐边缘。  
sed = filename | sed 'N;s/\n/\t/'  
  
# 对文件中的所有行编号（行号在左，文字右端对齐）。  
sed = filename | sed 'N; s/^/     /; s/ \*.{6,}\n/\1  /'  
  
# 对文件中的所有行编号，但只显示非空白行的行号。  
sed '/./=' filename | sed '/./N; s/\n/ /'  
  
# 计算行数 （模拟 "wc -l"）  
sed -n '$='

sed 's/.$//'      将文件每行字母最后一个删除

# 下面的脚本只对UnxUtils sed 4.0.7 及更高版本有效。要识别UnxUtils版本的  
#  sed可以通过其特有的“--text”选项。你可以使用帮助选项（“--help”）看  
# 其中有无一个“--text”项以此来判断所使用的是否是UnxUtils版本。其它DOS  
# 版本的的sed则无法进行这一转换。但可以用“tr”来实现这一转换。  
sed "s/\r//" infile >outfile     # UnxUtils sed v4.0.7 或更高版本  
tr -d \r <infile >outfile        # GNU tr 1.22 或更高版本  
# 将每一行前导的“空白字符”（空格，制表符）删除  
# 使之左对齐  
sed 's/^[ \t]\*//'                # 见本文末尾关于'\t'用法的描述  
# 将每一行拖尾的“空白字符”（空格，制表符）删除  
sed 's/[ \t]\*$//'                # 见本文末尾关于'\t'用法的描述  
# 将每一行中的前导和拖尾的空白字符删除  
sed 's/^[ \t]\*//;s/[ \t]\*$//'  
# 在每一行开头处插入5个空格（使全文向右移动5个字符的位置）  
sed 's/^/     /'  
# 以79个字符为宽度，将所有文本右对齐  
sed -e :a -e 's/^.\{1,78\}$/ &/;ta'  # 78个字符外加最后的一个空格  
# 以79个字符为宽度，使所有文本居中。在方法1中，为了让文本居中每一行的前  
# 头和后头都填充了空格。 在方法2中，在居中文本的过程中只在文本的前面填充  
# 空格，并且最终这些空格将有一半会被删除。此外每一行的后头并未填充空格。  
sed  -e :a -e 's/^.\{1,77\}$/ & /;ta'                     # 方法1  
sed  -e :a -e 's/^.\{1,77\}$/ &/;ta' -e 's/∗\1/\1/'  # 方法2  
# 在每一行中查找字串“foo”，并将找到的“foo”替换为“bar”  
sed 's/foo/bar/'                 # 只替换每一行中的第一个“foo”字串  
sed 's/foo/bar/4'                # 只替换每一行中的第四个“foo”字串  
sed 's/foo/bar/g'                # 将每一行中的所有“foo”都换成“bar”  
sed 's/.∗foo.∗foo/\1bar\2/' # 替换倒数第二个“foo”  
sed 's/.∗foo/\1bar/'            # 替换最后一个“foo”  
# 只在行中出现字串“baz”的情况下将“foo”替换成“bar”  
sed '/baz/s/foo/bar/g'  
# 将“foo”替换成“bar”，并且只在行中未出现字串“baz”的情况下替换  
sed '/baz/!s/foo/bar/g'  
# 不管是“scarlet”“ruby”还是“puce”，一律换成“red”  
sed 's/scarlet/red/g;s/ruby/red/g;s/puce/red/g'  #对多数的sed都有效  
gsed 's/scarlet\|ruby\|puce/red/g'               # 只对GNU sed有效  
# 倒置所有行，第一行成为最后一行，依次类推（模拟“tac”）。  
# 由于某些原因，使用下面命令时HHsed v1.5会将文件中的空行删除  
sed '1!G;h;$!d'               # 方法1  
sed -n '1!G;h;$p'             # 方法2  
# 将行中的字符逆序排列，第一个字成为最后一字，……（模拟“rev”）  
sed '/\n/!G;s/..∗\n/&\2\1/;//D;s/.//'  
# 将每两行连接成一行（类似“paste”）  
sed '$!N;s/\n/ /'  
# 如果当前行以反斜杠“\”结束，则将下一行并到当前行末尾  
# 并去掉原来行尾的反斜杠  
sed -e :a -e '/\\$/N; s/\\\n//; ta'  
# 如果当前行以等号开头，将当前行并到上一行末尾  
# 并以单个空格代替原来行头的“=”  
sed -e :a -e '$!N;s/\n=/ /;ta' -e 'P;D'  
# 为数字字串增加逗号分隔符号，将“1234567”改为“1,234,567”  
gsed ':a;s/\B[0-9]\{3\}\>/,&/;ta'                     # GNU sed  
sed -e :a -e 's/.∗[0−9][0−9]{3}/\1,\2/;ta'  # 其他sed  
# 为带有小数点和负号的数值增加逗号分隔符（GNU sed）  
gsed -r ':a;s/(^|[^0-9.])([0-9]+)([0-9]{3})/\1\2,\3/g;ta'  
# 在每5行后增加一空白行 （在第5，10，15，20，等行后增加一空白行）  
gsed '0~5G'                      # 只对GNU sed有效  
sed 'n;n;n;n;G;'                 # 其他sed  
选择性地显示特定行：  
--------  
# 显示文件中的前10行 （模拟“head”的行为）  
sed 10q  
# 显示文件中的第一行 （模拟“head -1”命令）  
sed q  
# 显示文件中的最后10行 （模拟“tail”）  
sed -e :a -e '$q;N;11,$D;ba'  
# 显示文件中的最后2行（模拟“tail -2”命令）  
sed '$!N;$!D'  
# 显示文件中的最后一行（模拟“tail -1”）  
sed '$!d'                        # 方法1  
sed -n '$p'                      # 方法2  
# 显示文件中的倒数第二行  
sed -e '$!{h;d;}' -e x              # 当文件中只有一行时，输入空行  
sed -e '1{$q;}' -e '$!{h;d;}' -e x  # 当文件中只有一行时，显示该行  
sed -e '1{$d;}' -e '$!{h;d;}' -e x  # 当文件中只有一行时，不输出  
# 只显示匹配正则表达式的行（模拟“grep”）  
sed -n '/regexp/p'               # 方法1  
sed '/regexp/!d'                 # 方法2  
# 只显示“不”匹配正则表达式的行（模拟“grep -v”）  
sed -n '/regexp/!p'              # 方法1，与前面的命令相对应  
sed '/regexp/d'                  # 方法2，类似的语法  
# 查找“regexp”并将匹配行的上一行显示出来，但并不显示匹配行  
sed -n '/regexp/{g;1!p;};h'  
# 查找“regexp”并将匹配行的下一行显示出来，但并不显示匹配行  
sed -n '/regexp/{n;p;}'  
# 显示包含“regexp”的行及其前后行，并在第一行之前加上“regexp”所  
# 在行的行号 （类似“grep -A1 -B1”）  
sed -n -e '/regexp/{=;x;1!p;g;$!N;p;D;}' -e h  
# 显示包含“AAA”、“BBB”或“CCC”的行（任意次序）  
sed '/AAA/!d; /BBB/!d; /CCC/!d'  # 字串的次序不影响结果  
# 显示包含“AAA”、“BBB”和“CCC”的行（固定次序）  
sed '/AAA.\*BBB.\*CCC/!d'  
# 显示包含“AAA”“BBB”或“CCC”的行 （模拟“egrep”）  
sed -e '/AAA/b' -e '/BBB/b' -e '/CCC/b' -e d    # 多数sed  
gsed '/AAA\|BBB\|CCC/!d'                        # 对GNU sed有效  
# 显示包含“AAA”的段落 （段落间以空行分隔）  
# HHsed v1.5 必须在“x;”后加入“G;”，接下来的3个脚本都是这样  
sed -e '/./{H;$!d;}' -e 'x;/AAA/!d;'  
# 显示包含“AAA”“BBB”和“CCC”三个字串的段落 （任意次序）  
sed -e '/./{H;$!d;}' -e 'x;/AAA/!d;/BBB/!d;/CCC/!d'  
# 显示包含“AAA”、“BBB”、“CCC”三者中任一字串的段落 （任意次序）  
sed -e '/./{H;$!d;}' -e 'x;/AAA/b' -e '/BBB/b' -e '/CCC/b' -e d  
gsed '/./{H;$!d;};x;/AAA\|BBB\|CCC/b;d'         # 只对GNU sed有效  
# 显示包含65个或以上字符的行  
sed -n '/^.\{65\}/p'  
# 显示包含65个以下字符的行  
sed -n '/^.\{65\}/!p'            # 方法1，与上面的脚本相对应  
sed '/^.\{65\}/d'                # 方法2，更简便一点的方法  
# 显示部分文本??从包含正则表达式的行开始到最后一行结束  
sed -n '/regexp/,$p'  
# 显示部分文本??指定行号范围（从第8至第12行，含8和12行）  
sed -n '8,12p'                   # 方法1  
sed '8,12!d'                     # 方法2  
# 显示第52行  
sed -n '52p'                     # 方法1  
sed '52!d'                       # 方法2  
sed '52q;d'                      # 方法3, 处理大文件时更有效率  
# 从第3行开始，每7行显示一次      
gsed -n '3~7p'                   # 只对GNU sed有效  
sed -n '3,${p;n;n;n;n;n;n;}'     # 其他sed  
# 显示两个正则表达式之间的文本（包含）  
sed -n '/Iowa/,/Montana/p'       # 区分大小写方式  
选择性地删除特定行：  
--------  
# 显示通篇文档，除了两个正则表达式之间的内容  
sed '/Iowa/,/Montana/d'  
# 删除文件中相邻的重复行（模拟“uniq”）  
# 只保留重复行中的第一行，其他行删除  
sed '$!N; /^.∗\n\1$/!P; D'  
# 删除文件中的重复行，不管有无相邻。注意hold space所能支持的缓存  
# 大小，或者使用GNU sed。  
sed -n 'G; s/\n/&&/; /^[− ]∗\n.\*\n\1/d; s/\n//; h; P'  
# 删除除重复行外的所有行（模拟“uniq -d”）  
sed '$!N; s/^.∗\n\1$/\1/; t; D'  
# 删除文件中开头的10行  
sed '1,10d'  
# 删除文件中的最后一行  
sed '$d'  
# 删除文件中的最后两行  
sed 'N;$!P;$!D;$d'  
# 删除文件中的最后10行  
sed -e :a -e '$d;N;2,10ba' -e 'P;D'   # 方法1  
sed -n -e :a -e '1,10!{P;N;D;};N;ba'  # 方法2  
# 删除8的倍数行  
gsed '0~8d'                           # 只对GNU sed有效  
sed 'n;n;n;n;n;n;n;d;'                # 其他sed  
# 删除匹配式样的行  
sed '/pattern/d'                      # 删除含pattern的行。当然pattern  
                                       # 可以换成任何有效的正则表达式  
# 删除文件中的所有空行（与“grep '.' ”效果相同）  
sed '/^$/d'                           # 方法1  
sed '/./!d'                           # 方法2  
# 只保留多个相邻空行的第一行。并且删除文件顶部和尾部的空行。  
# （模拟“cat -s”）  
sed '/./,/^$/!d'        #方法1，删除文件顶部的空行，允许尾部保留一空行  
sed '/^$/N;/\n$/D'      #方法2，允许顶部保留一空行，尾部不留空行  
# 只保留多个相邻空行的前两行。  
sed '/^$/N;/\n$/N;//D'  
# 删除文件顶部的所有空行  
sed '/./,$!d'  
# 删除文件尾部的所有空行  
sed -e :a -e '/^\n\*$/{$d;N;ba' -e '}'  # 对所有sed有效  
sed -e :a -e '/^\n\*$/N;/\n$/ba'        # 同上，但只对 gsed 3.02.\*有效  
# 删除每个段落的最后一行  
sed -n '/^$/{p;h;};/./{x;/./p;}'  
特殊应用：  
--------  
# 移除手册页（man page）中的nroff标记。在Unix System V或bash shell下使  
# 用'echo'命令时可能需要加上 -e 选项。  
sed "s/.`echo \\\b`//g"    # 外层的双括号是必须的（Unix环境）  
sed 's/.^H//g'             # 在bash或tcsh中, 按 Ctrl-V 再按 Ctrl-H  
sed 's/.\x08//g'           # sed 1.5，GNU sed，ssed所使用的十六进制的表示方法  
# 提取新闻组或 e-mail 的邮件头  
sed '/^$/q'                # 删除第一行空行后的所有内容  
# 提取新闻组或 e-mail 的正文部分  
sed '1,/^$/d'              # 删除第一行空行之前的所有内容  
# 从邮件头提取“Subject”（标题栏字段），并移除开头的“Subject:”字样  
sed '/^Subject: \*/!d; s///;q'  
# 从邮件头获得回复地址  
sed '/^Reply-To:/q; /^From:/h; /./d;g;q'  
# 获取邮件地址。在上一个脚本所产生的那一行邮件头的基础上进一步的将非电邮  
# 地址的部分剃除。（见上一脚本）  
sed 's/ \*(.\*)//; s/>.\*//; s/.\*[:<] \*//'  
# 在每一行开头加上一个尖括号和空格（引用信息）  
sed 's/^/> /'  
# 将每一行开头处的尖括号和空格删除（解除引用）  
sed 's/^> //'  
# 移除大部分的HTML标签（包括跨行标签）  
sed -e :a -e 's/<[^>]\*>//g;/</N;//ba'  
# 将分成多卷的uuencode文件解码。移除文件头信息，只保留uuencode编码部分。  
# 文件必须以特定顺序传给sed。下面第一种版本的脚本可以直接在命令行下输入；  
# 第二种版本则可以放入一个带执行权限的shell脚本中。（由Rahul Dhesi的一  
# 个脚本修改而来。）  
sed '/^end/,/^begin/d' file1 file2 ... fileX | uudecode   # vers. 1  
sed '/^end/,/^begin/d' "$@" | uudecode                    # vers. 2  
# 将文件中的段落以字母顺序排序。段落间以（一行或多行）空行分隔。GNU sed使用  
# 字元“\v”来表示垂直制表符，这里用它来作为换行符的占位符??当然你也可以  
# 用其他未在文件中使用的字符来代替它。  
sed '/./{H;d;};x;s/\n/={NL}=/g' file | sort | sed '1s/={NL}=//;s/={NL}=/\n/g'  
gsed '/./{H;d};x;y/\n/\v/' file | sort | sed '1s/\v//;y/\v/\n/'  
# 分别压缩每个.TXT文件，压缩后删除原来的文件并将压缩后的.ZIP文件  
# 命名为与原来相同的名字（只是扩展名不同）。（DOS环境：“dir /b”  
# 显示不带路径的文件名）。  
echo @echo off >zipup.bat  
dir /b \*.txt | sed "s/^.∗\.TXT/pkzip -mo \1 \1.TXT/" >>zipup.bat

# (十一)linux内核参数的优化

Linux系统的内核参数配置是在/etc/sysctl.conf上面,针对服务器的功能不同,可以对配置进行修改.进而压榨服务器的性能,进而使得服务器的在该性能上达到最佳.

fs.file-max = 999999

net.ipv4.tcp\_tw\_reuse = 1

net.ipv4.tcp\_keepalive\_time = 600

net.ipv4.tcp\_fin\_timeout = 30

net.ipv4.tcp\_max\_tw\_buckets = 5000

net.ipv4.ip\_local\_port\_range = 1024 61000

net.ipv4.tcp\_rmem = 4096 32768 262142

net.ipv4.tcp\_wmem = 4096 32768 262142

net.core.netdev\_max\_backlog = 8096

net.core.rmem\_default = 262144

net.core.wmem\_default = 262144

net.core.rmem\_max = 2097152

net.core.wmem\_max = 2097152

net.ipv4.tcp\_syncookies = 1

net.ipv4.tcp\_max\_syn.backlog=1024

然后执行sysctl-p命令，使上述修改生效。

上面的参数意义解释如下：

·file-max：这个参数表示进程（比如一个worker进程）可以同时打开的最大句柄数，这

个参数直接限制最大并发连接数，需根据实际情况配置。

·tcp\_tw\_reuse：这个参数设置为1，表示允许将TIME-WAIT状态的socket重新用于新的

TCP连接，这对于服务器来说很有意义，因为服务器上总会有大量TIME-WAIT状态的连接。

·tcp\_keepalive\_time：这个参数表示当keepalive启用时，TCP发送keepalive消息的频度。

默认是2小时，若将其设置得小一些，可以更快地清理无效的连接。

·tcp\_fin\_timeout：这个参数表示当服务器主动关闭连接时，socket保持在FIN-WAIT-2状

态的最大时间。

·tcp\_max\_tw\_buckets：这个参数表示操作系统允许TIME\_WAIT套接字数量的最大值，

如果超过这个数字，TIME\_WAIT套接字将立刻被清除并打印警告信息。该参数默认为

180000，过多的TIME\_WAIT套接字会使Web服务器变慢。

·tcp\_max\_syn\_backlog：这个参数表示TCP三次握手建立阶段接收SYN请求队列的最大

长度，默认为1024，将其设置得大一些可以使出现Nginx繁忙来不及accept新连接的情况时，

Linux不至于丢失客户端发起的连接请求。

·ip\_local\_port\_range：这个参数定义了在UDP和TCP连接中本地（不包括连接的远端）

端口的取值范围。

·net.ipv4.tcp\_rmem：这个参数定义了TCP接收缓存（用于TCP接收滑动窗口）的最小

值、默认值、最大值。

·net.ipv4.tcp\_wmem：这个参数定义了TCP发送缓存（用于TCP发送滑动窗口）的最小

值、默认值、最大值。

·netdev\_max\_backlog：当网卡接收数据包的速度大于内核处理的速度时，会有一个队列

保存这些数据包。这个参数表示该队列的最大值。

·rmem\_default：这个参数表示内核套接字接收缓存区默认的大小。

·wmem\_default：这个参数表示内核套接字发送缓存区默认的大小。

·rmem\_max：这个参数表示内核套接字接收缓存区的最大大小。

·wmem\_max：这个参数表示内核套接字发送缓存区的最大大小。