Netkiller Linux Shell 手札

netkiller Neo Chan

2009-11-15

文档出处: http://netkiller.sourceforge.net/ | http://netkiller.github.com

文档最近一次更新于 Thu Dec 1 12:48:41 UTC 2011

下面是我多年积累下来的经验总结，整理成文档供大家参考:

Netkiller Architect 手札	Netkiller Linux 手札	Netkiller Developer 手札	Netkiller Database 手札
Netkiller Debian 手札	Netkiller CentOS 手札	Netkiller FreeBSD 手札	Netkiller Shell 手札
Netkiller Web 手札	Netkiller Monitoring 手札	Netkiller Storage 手札	Netkiller Mail System 手札
Netkiller MySQL 手札	Netkiller LDAP 手札	Netkiller Security 手札	Netkiller Version 手札
Netkiller Intranet 手札	Netkiller Cisco IOS 手札	Netkiller Writer 手札	Netkiller Studio Linux 手札

目录

范例清单

下一页
自述

自述

目录

1. 内容简介

当前文档档容比较杂，涉及内容广泛。

慢慢我会将其中章节拆成新文档.

文档内容简介:

1. Network
2. Security
3. Web Application
4. Database
5. Storage And Backup/Restore
6. Cluster
7. Developer

1.1. Audience(读者对象)

This book is intended primarily for Linux system administrators who are familiar with the following activities:

Audience

1. Linux system administration procedures, including kernel configuration
2. Installation and configuration of cluster, such as load balancing, High Availability,
3. Installation and configuration of shared storage networks, such as Fibre Channel SANs
4. Installation and configuration of web server, such as apache, nginx, lighttpd, tomcat/resin ...

本文档的读者对象:

文档面向有所有读者。您可以选读您所需要的章节,无需全篇阅读,因为有些章节不一定对你有用,用得着就翻来看看,暂时用不到的可以不看.

大体分来读者可以分为几类:

1. 架构工程师
2. 系统管理员
3. 系统支持,部署工程师

不管是谁,做什么的,我希望通过阅读这篇文档都能对你有所帮助。

1.2. 写给读者

欢迎提出宝贵的建议,如有问题请到 邮件列表 讨论

为什么写这篇文章

有很多想法,工作中也用不到所以未能实现，所以想写出来,和大家分享.有一点写一点,写得也不好,只要能看懂就行,就当学习笔记了.

开始零零碎碎写过一些文档，也向维基百科供过稿，但维基经常被ZF封锁，后来发现sf.net可以提供主机存放文档，便做了迁移。并开始了我的写作生涯。

这篇文档是作者8年来对工作的总结,是作者一点一滴的积累起来的，有些笔记已经丢失，所以并不完整。

因为工作太忙整理比较缓慢。目前的工作涉及面比较窄所以新文档比较少。

我现在花在技术上的时间越来越少，兴趣转向摄影，无线电。也想写写摄影方面的心得体会。

写作动力:

曾经在网上看到外国开源界对中国的评价，中国人对开源索取无度，但贡献却微乎其微.这句话一直记在我心中，发誓要为中国开源事业做我仅有的一点微薄贡献

另外写文档也是知识积累，还可以增加在圈内的影响力.

人跟动物的不同,就是人类可以把自己学习的经验教给下一代人.下一代在上一代的基础上再创新,不断积累才有今天.

所以我把自己的经验写出来,可以让经验传承

没有内容的章节:

目前我自己一人维护所有文档，写作时间有限，当我发现一个好主题就会加入到文档中，待我有时间再完善章节，所以你会发现很多章节是空无内容的.

文档目前几乎是流水帐试的写作，维护量很大，先将就着看吧.

我想到哪写到哪,你会发现文章没一个中心,今天这里写点,明天跳过本章写其它的.
文中例子绝对多,对喜欢复制然后粘贴朋友很有用,不用动手写,也省时间.
理论的东西,网上大把,我这里就不写了,需要可以去网上查.
我爱写错别字,还有一些是打错的,如果发现请指正.
文中大部分试验是在Debian/Ubuntu/Redhat AS上完成.

1.3. 获得文档

1.3.1. PDF

Download PDF Document 下载PDF文档1

Download PDF Document 下载PDF文档2

1.3.2. EPUB

http://netkiller.sourceforge.net/technology.html

1.3.3. 获得光盘介质

如有特别需要，请联系我

---

2. 作者简介

主页地址：http://netkiller.sourceforge.net, http://netkiller.github.com/

陈景峰 (ㄔㄣˊ ㄐㄧㄥˇ ㄈㄥ)

Nickname：netkiller | English name: Neo chen | Nippon name: ちんけいほう (音訳) | Korean name: | Thailand name:

IT民工， UNIX like Evangelist, 业余无线电爱好者（呼号：BG7NYT）,户外运动以及摄影爱好者。

《PostgreSQL实用实例参考》，《Postfix 完整解决方案》，《Netkiller Linux 手札》的作者

2001年来深圳进城打工,成为一名外来务工者.

2002年我发现不能埋头苦干,埋头搞技术是不对的,还要学会"做人".

2003年这年最惨,公司拖欠工资16000元,打过两次官司2005才付清.

2004年开始加入分布式计算团队,目前成绩

2004-10月开始玩户外和摄影

2005-6月成为中国无线电运动协会会员

2006年单身生活了这么多年,终于找到归宿.

2007物价上涨,金融危机，休息了4个月（其实是找不到工作）

2008终于找到英文学习方法，， 《Netkiller Developer 手札》，《Netkiller Document 手札》

2008-8-8 08:08:08 结婚,后全家迁居湖南省常德市

2009《Netkiller Database 手札》,年底拿到C1驾照

2010对电子打击乐产生兴趣，计划学习爵士鼓

2011 职业生涯路上继续打怪升级

2.1. 联系作者

Mobile: +86 13113668890

Tel: +86 755 2981-2080

Callsign: BG7NYT QTH: Shenzhen, China

注：请不要问我安装问题!

E-Mail: openunix@163.com
IRC irc.freenode.net #ubuntu / #ubuntu-cn

Yahoo: bg7nyt
ICQ: 101888222
AIM: bg7nyt

TM/QQ: 13721218
MSN: netkiller@msn.com
G Talk: 很少开
网易泡泡：很少开

写给火腿:

欢迎无线电爱好者和我QSO,我的QTH在深圳宝安区龙华镇溪山美地12B7CD,设备YAESU FT-50R,FT-60R,FT-7800 144-430双段机,拉杆天线/GP天线 Nagoya MAG-79EL-3W/Yagi

如果这篇文章对你有所帮助,请寄给我一张QSL卡片,qrz.cn or qrz.com or hamcall.net

Personal Amateur Radiostations of P.R.China

ZONE CQ24 ITU44 ShenZhen, China

Best Regards, VY 73! OP. BG7NYT

---

3. 支持这个项目(Support this project)

Donations

招商银行(China Merchants Bank) 陈景峰 9555500000007459

第 1 章 Introduction

目录

1. chsh - change login shell

```
# chsh --list
/bin/sh
/bin/bash
/sbin/nologin
/bin/tcsh
/bin/csh
/bin/ksh
```

```
neo@netkiller:~$ chsh -s /bin/zsh
```

show me current shell

```
neo@netkiller:~$ echo $SHELL
/bin/zsh

neo@netkiller:~$ cat /etc/passwd|grep neo
neo:x:1000:1000:Neo Chen,,,:/home/neo:/bin/zsh
```

2. I/O 重定向

```
cat <<End-of-message
   8 -----------------------------------
   9 This is line 1 of the message.
  10 This is line 2 of the message.
  11 This is line 3 of the message.
  12 This is line 4 of the message.
  13 This is the last line of the message.
  14 -----------------------------------
End-of-message
```

```
MYSQL=mysql
MYSQLOPTS="-h $zs_host -u $zs_user -p$zs_pass $zs_db"

$MYSQL $MYSQLOPTS <<SQL
SELECT
        category.cat_id AS  cat_id ,
        category.cat_name AS  cat_name ,
        category.cat_desc AS  cat_desc ,
        category.parent_id AS  parent_id ,
        category.sort_order AS  sort_order ,
        category.measure_unit AS  measure_unit ,
        category.style AS  style ,
        category.is_show AS is_show ,
        category.grade AS  grade
FROM  category
SQL
```

<<-LimitString可以抑制输出时前边的tab(不是空格). 这可以增加一个脚本的可读性.

```
cat <<-ENDOFMESSAGE
        This is line 1 of the message.
        This is line 2 of the message.
        This is line 3 of the message.
        This is line 4 of the message.
        This is the last line of the message.
ENDOFMESSAGE
```

关闭参数替换

```
NAME="John Doe"
RESPONDENT="the author of this fine script"

cat <<'Endofmessage'

Hello, there, $NAME.
Greetings to you, $NAME, from $RESPONDENT.

Endofmessage
```

```
NAME="John Doe"
```

```
RESPONDENT="the author of this fine script"

cat <<\Endofmessage

Hello, there, $NAME.
Greetings to you, $NAME, from $RESPONDENT.

Endofmessage
```

2.1. error 重定向

```
your_shell 2>&1
```

2.2. 使用块记录日志

```
{
        ...
        ...
} > $LOGFILE 2>&1
```

2.3. tee - read from standard input and write to standard output and files

```
echo 1 > /proc/sys/net/ipv4/ip_forward
echo 1 | sudo tee /proc/sys/net/ipv4/ip_forward;
```

2.3.1. nettee - a network "tee" program

```

```

3. pipes (FIFOs)

create a pipes

```
$ mkfifo /tmp/pipe
$ mkfifo -m 0644 /tmp/pipe

$ mknod /tmp/pipe p
```

let's see it

```
$ ls -l /tmp/piple
prw-r--r-- 1 neo neo 0 2009-03-13 14:40 /tmp/piple
```

remove a pipes

```
rm /tmp/pipe
```

using it

standing by pipe

```
$ cat /tmp/pipe
```

push string to pipe

```
$ echo hello world > /tmp/pipe
```

fetch string from /tmp/pipe

```
$ cat /tmp/piple
hello world
```

部分 I. Bash Shell

目录

第 2 章 prompt

.bashrc

```
# Prompt definitions
if [ -f ~/.bash_prompt ]; then
        . ~/.bash_prompt
fi
```

.bash_prompt

```
#!/bin/bash

function tonka2 {
local GRAY="\[\033[1;30m\]"
local LIGHT_GRAY="\[\033[0;37m\]"
local WHITE="\[\033[1;37m\]"

local LIGHT_BLUE="\[\033[1;34m\]"
local LIGHT_RED="\[\033[1;31m\]"
local YELLOW="\[\033[1;33m\]"

case $TERM in
    xterm*)
        TITLEBAR='\[\033]0;\u@\h:\w\007\]'
        ;;
    *)
        TITLEBAR=""
        ;;
esac

PS1="$TITLEBAR\
$YELLOW-$LIGHT_BLUE-(\
$YELLOW\u$LIGHT_BLUE@$YELLOW\h\
$LIGHT_BLUE)-(\
$YELLOW\$PWD\
$LIGHT_BLUE)-$YELLOW-\
$LIGHT_GRAY\n\
$YELLOW-$LIGHT_BLUE-(\
$YELLOW\$(date +%F)$LIGHT_BLUE:$YELLOW\$(date +%I:%M:%S)\
$LIGHT_BLUE:$WHITE\$$LIGHT_BLUE)-$YELLOW-$LIGHT_GRAY "

PS2="$LIGHT_BLUE-$YELLOW-$YELLOW-$LIGHT_GRAY "
}

function proml {
local BLUE="\[\033[0;34m\]"
local RED="\[\033[0;31m\]"
local LIGHT_RED="\[\033[1;31m\]"
local WHITE="\[\033[1;37m\]"
local NO_COLOUR="\[\033[0m\]"
case $TERM in
    xterm*|rxvt*)
        TITLEBAR='\[\033]0;\u@\h:\w\007\]'
        ;;
    *)
        TITLEBAR=""
        ;;
esac

PS1="${TITLEBAR}\
$BLUE[$RED\$(date +%H%M)$BLUE]\
$BLUE[$LIGHT_RED\u@\h:\w$BLUE]\
$WHITE\$$NO_COLOUR "
PS2='> '
PS4='+ '
}
```

```
function neo_prompt {
local GRAY="\[\033[1;30m\]"
local LIGHT_GRAY="\[\033[0;37m\]"
local WHITE="\[\033[1;37m\]"

local LIGHT_BLUE="\[\033[1;34m\]"
local LIGHT_RED="\[\033[1;31m\]"
local YELLOW="\[\033[1;33m\]"

case $TERM in
    xterm*)
        TITLEBAR='\[\033]0;\u@\h:\w\007\]'
        ;;
    *)
        TITLEBAR=""
        ;;
esac

PS1="$TITLEBAR\
$YELLOW-$LIGHT_BLUE-(\
$YELLOW\$(date +%F)$LIGHT_BLUE $YELLOW\$(date +%I:%M:%S)\
$LIGHT_BLUE)-(\
$YELLOW\$PWD\
$LIGHT_BLUE)-$YELLOW-\
$LIGHT_GRAY\n\
$YELLOW-$LIGHT_BLUE-(\
$YELLOW\u$LIGHT_BLUE@$YELLOW\h\
$LIGHT_BLUE:$WHITE\$$LIGHT_BLUE)-$YELLOW-$LIGHT_GRAY "

PS2="$LIGHT_BLUE-$YELLOW-$YELLOW-$LIGHT_GRAY "
}

# Created by KrON from windowmaker on IRC
# Changed by Spidey 08/06
function elite {
PS1="\[\033[31m\]\332\304\[\033[34m\](\[\033[31m\]\u\[\033[34m\]@\[\033[31m\]\h\
\[\033[34m\])\[\033[31m\]-\[\033[34m\](\[\033[31m\]\$(date +%I:%M%P)\
\[\033[34m\]-:-\[\033[31m\]\$(date +%m)\[\033[34m\033[31m\]/\$(date +%d)\
\[\033[34m\])\[\033[31m\]\304-\[\033[34m]\\371\[\033[31m\]-\371\371\
\[\033[34m\]\372\n\[\033[31m\]\300\304\[\033[34m\](\[\033[31m\]\W\[\033[34m\])\
\[\033[31m\]\304\371\[\033[34m\]\372\[\033[00m\]"
PS2="> "
}
```

例 2.1. A "Power User" Prompt

.bash_prompt

```
#!/bin/bash
#-------------------------------------------------------------------------
#       POWER USER PROMPT "pprom2"
#-------------------------------------------------------------------------
#
#   Created August 98, Last Modified 9 November 98 by Giles
#
#   Problem: when load is going down, it says "1.35down-.08", get rid
#   of the negative

function prompt_command
{
#   Create TotalMeg variable: sum of visible file sizes in current directory
local TotalBytes=0
for Bytes in $(ls -l | grep "^-" | awk '{print $5}')
do
    let TotalBytes=$TotalBytes+$Bytes
done
TotalMeg=$(echo -e "scale=3 \nx=$TotalBytes/1048576\n if (x<1) {print \"0\"} \n
print x \nquit" | bc)

#      This is used to calculate the differential in load values
#      provided by the "uptime" command.  "uptime" gives load
#      averages at 1, 5, and 15 minute marks.
#
local one=$(uptime | sed -e "s/.*load average: \(.*\...\), \(.*\...\),
\(.*\...\)/\1/" -e "s/ //g")
local five=$(uptime | sed -e "s/.*load average: \(.*\...\), \(.*\...\),
```

```
\(.*\...\).*/\2/" -e "s/ //g")
local diff1_5=$(echo -e "scale = scale ($one) \nx=$one - $five\n if (x>0) {print
\"up\"} else {print \"down\"}\n print x \nquit \n" | bc)
loaddiff="$(echo -n "${one}${diff1_5}")"

#   Count visible files:
let files=$(ls -l | grep "^-" | wc -l | tr -d " ")
let hiddenfiles=$(ls -l -d .* | grep "^-" | wc -l | tr -d " ")
let executables=$(ls -l | grep ^-..x | wc -l | tr -d " ")
let directories=$(ls -l | grep "^d" | wc -l | tr -d " ")
let hiddendirectories=$(ls -l -d .* | grep "^d" | wc -l | tr -d " ")-2
let linktemp=$(ls -l | grep "^l" | wc -l | tr -d " ")
if [ "$linktemp" -eq "0" ]
then
    links=""
else
    links=" ${linktemp}l"
fi
unset linktemp
let devicetemp=$(ls -l | grep "^[bc]" | wc -l | tr -d " ")
if [ "$devicetemp" -eq "0" ]
then
    devices=""
else
    devices=" ${devicetemp}bc"
fi
unset devicetemp

}

PROMPT_COMMAND=prompt_command

function pprom2 {

local        BLUE="\[\033[0;34m\]"
local  LIGHT_GRAY="\[\033[0;37m\]"
local LIGHT_GREEN="\[\033[1;32m\]"
local  LIGHT_BLUE="\[\033[1;34m\]"
local  LIGHT_CYAN="\[\033[1;36m\]"
local      YELLOW="\[\033[1;33m\]"
local       WHITE="\[\033[1;37m\]"
local         RED="\[\033[0;31m\]"
local   NO_COLOUR="\[\033[0m\]"

case $TERM in
    xterm*)
        TITLEBAR='\[\033]0;\u@\h:\w\007\]'
        ;;
    *)
        TITLEBAR=""
        ;;
esac

PS1="$TITLEBAR\
$BLUE[$RED\$(date +%H%M)$BLUE]\
$BLUE[$RED\u@\h$BLUE]\
$BLUE[\
$LIGHT_GRAY\${files}.\${hiddenfiles}-\
$LIGHT_GREEN\${executables}x \
$LIGHT_GRAY(\${TotalMeg}Mb) \
$LIGHT_BLUE\${directories}.\
\${hiddendirectories}d\
$LIGHT_CYAN\${links}\
$YELLOW\${devices}\
$BLUE]\
$BLUE[${WHITE}\${loaddiff}$BLUE]\
$BLUE[\
$WHITE\$(ps ax | wc -l | sed -e \"s: ::g\")proc\
$BLUE]\
\n\
$BLUE[$RED\$PWD$BLUE]\
$WHITE\$\
\
$NO_COLOUR "
PS2='> '
PS4='+ '
}
```

例 2.2. A Prompt the Width of Your Term

```
#!/bin/bash
#   termwide prompt with tty number
#      by Giles - created 2 November 98, last tweaked 31 July 2001
#
#     This is a variant on "termwide" that incorporates the tty number.
#

hostnam=$(hostname -s)
usernam=$(whoami)
temp="$(tty)"
#   Chop off the first five chars of tty (ie /dev/):
cur_tty="${temp:5}"
unset temp

function prompt_command {

#   Find the width of the prompt:
TERMWIDTH=${COLUMNS}

#   Add all the accessories below ...
local temp="--(${usernam}@${hostnam}:${cur_tty})---(${PWD})--"

let fillsize=${TERMWIDTH}-${#temp}
if [ "$fillsize" -gt "0" ]
then
        fill="--------------------------------------------------------------------------------
------------------------------------------------------------------------"
        #   It's theoretically possible someone could need more
        #   dashes than above, but very unlikely!  HOWTO users,
        #   the above should be ONE LINE, it may not cut and
        #   paste properly
        fill="${fill:0:${fillsize}}"
        newPWD="${PWD}"
fi

if [ "$fillsize" -lt "0" ]
then
        fill=""
        let cut=3-${fillsize}
        newPWD="...${PWD:${cut}}"
fi
}

PROMPT_COMMAND=prompt_command

function twtty {

local WHITE="\[\033[1;37m\]"
local NO_COLOUR="\[\033[0m\]"

local LIGHT_BLUE="\[\033[1;34m\]"
local YELLOW="\[\033[1;33m\]"

case $TERM in
    xterm*|rxvt*)
        TITLEBAR='\[\033]0;\u@\h:\w\007\]'
        ;;
    *)
        TITLEBAR=""
        ;;
esac

PS1="$TITLEBAR\
$YELLOW-$LIGHT_BLUE-(\
$YELLOW\$usernam$LIGHT_BLUE@$YELLOW\$hostnam$LIGHT_BLUE:$WHITE\$cur_tty\
${LIGHT_BLUE})-${YELLOW}-\${fill}${LIGHT_BLUE}-(\
$YELLOW\${newPWD}\
$LIGHT_BLUE)-$YELLOW-\
\n\
$YELLOW-$LIGHT_BLUE-(\
$YELLOW\$(date +%H%M)$LIGHT_BLUE:$YELLOW\$(date \"+%a,%d %b %y\")\
$LIGHT_BLUE:$WHITE\$$LIGHT_BLUE)-\
$YELLOW-\
$NO_COLOUR "

PS2="$LIGHT_BLUE-$YELLOW-$YELLOW-$NO_COLOUR "

}
```

例 2.3. The Elegant Useless Clock Prompt

```
#!/bin/bash

#   This prompt requires a VGA font.  The prompt is anchored at the bottom
#   of the terminal, fills the width of the terminal, and draws a line up
#   the right side of the terminal to attach itself to a clock in the upper
#   right corner of the terminal.

function prompt_command {
#   Calculate the width of the prompt:
hostnam=$(echo -n $HOSTNAME | sed -e "s/[\.].*//")
#   "whoami" and "pwd" include a trailing newline
usernam=$(whoami)
newPWD="${PWD}"
#   Add all the accessories below ...
let promptsize=$(echo -n "--(${usernam}@${hostnam})---(${PWD})-----" \
                 | wc -c | tr -d " ")
#   Figure out how much to add between user@host and PWD (or how much to
#   remove from PWD)
let fillsize=${COLUMNS}-${promptsize}
fill=""
#   Make the filler if prompt isn't as wide as the terminal:
while [ "$fillsize" -gt "0" ]
do
   fill="${fill}Ä"
   # The A with the umlaut over it (it will appear as a long dash if
   # you're using a VGA font) is \304, but I cut and pasted it in
   # because Bash will only do one substitution - which in this case is
   # putting $fill in the prompt.
   let fillsize=${fillsize}-1
done
#   Right-truncate PWD if the prompt is going to be wider than the terminal:
if [ "$fillsize" -lt "0" ]
then
   let cutt=3-${fillsize}
   newPWD="...$(echo -n $PWD | sed -e "s/\(^.\{$cutt\}\)\(.*\)/\2/")"
fi
#
#   Create the clock and the bar that runs up the right side of the term
#
local LIGHT_BLUE="\033[1;34m"
local     YELLOW="\033[1;33m"
#   Position the cursor to print the clock:
echo -en "\033[2;$((${COLUMNS}-9))H"
echo -en "$LIGHT_BLUE($YELLOW$(date +%H%M)$LIGHT_BLUE)\304$YELLOW\304\304\277"
local i=${LINES}
echo -en "\033[2;${COLUMNS}H"
#   Print vertical dashes down the side of the terminal:
while [ $i -ge 4 ]
do
   echo -en "\033[$(($i-1));${COLUMNS}H\263"
   let i=$i-1
done

let prompt_line=${LINES}-1
#   This is needed because doing \${LINES} inside a Bash mathematical
#   expression (ie. $(())) doesn't seem to work.
}

PROMPT_COMMAND=prompt_command

function clock3 {
local LIGHT_BLUE="\[\033[1;34m\]"
local     YELLOW="\[\033[1;33m\]"
local      WHITE="\[\033[1;37m\]"
local LIGHT_GRAY="\[\033[0;37m\]"
local  NO_COLOUR="\[\033[0m\]"

case $TERM in
    xterm*)
        TITLEBAR='\[\033]0;\u@\h:\w\007\]'
        ;;
    *)
        TITLEBAR=""
        ;;
esac

PS1="$TITLEBAR\
\[\033[\${prompt line};0H\]
```

```
$YELLOW\332$LIGHT_BLUE\304(\
$YELLOW\${usernam}$LIGHT_BLUE@$YELLOW\${hostnam}\
${LIGHT_BLUE})\304${YELLOW}\304\${fill}${LIGHT_BLUE}\304(\
$YELLOW\${newPWD}\
$LIGHT_BLUE)\304$YELLOW\304\304\304\331\
\n\
$YELLOW\300$LIGHT_BLUE\304(\
$YELLOW\$(date \"+%a,%d %b %y\")\
$LIGHT_BLUE:$WHITE\$$LIGHT_BLUE)\304\
$YELLOW\304\
$LIGHT_GRAY "

PS2="$LIGHT_BLUE\304$YELLOW\304$YELLOW\304$NO_COLOUR "

}
```

第 3 章 variable

目录

1. set

```
$ set -- `echo aa bb cc`
$ echo $1
aa
$ echo $2
bb
$ echo $3
cc

$ set -- aa bb cc
```

2. export / unset

```
export CATALINA_OUT=/www/logs/tomcat/catalina.out
unset CATALINA_OUT
```

3. declare

功能说明：声明 shell 变量。

语　法：declare [+/-][rxi][变量名称=设置值] 或 declare -f

补充说明：declare为shell指令，在第一种语法中可用来声明变量并设置变量的属性([rix]即为变量的属性），在第二种语法中可用来显示shell函数。若不加上任何参数，则会显示全部的shell变量与函数(与执行set指令的效果相同)。

参　数：
+/-　"-"可用来指定变量的属性，"+"则是取消变量所设的属性。
-f　仅显示函数。
r　将变量设置为只读。
x　指定的变量会成为环境变量，可供shell以外的程序来使用。
i　[设置值]可以是数值，字符串或运算式。

4. 系统变量

系统变量

```
Shell常用的系统变量并不多，但却十分有用，特别是在做一些参数检测的时候。下面是Shell常用的系统变量
表示方法     描述
$n        $1 表示第一个参数，$2 表示第二个参数 ...
$#        命令行参数的个数
$0        当前程序的名称
$?        前一个命令或函数的返回码
$*        以"参数1 参数2 ... " 形式保存所有参数
$@        以"参数1" "参数2" ... 形式保存所有参数
$$        本程序的(进程ID号)PID
$!        上一个命令的PID
```

```
[root@cc tmp]# cat test.sh
echo $#
echo $@

[root@cc tmp]# ./test.sh  helloworld
1
helloworld
```

```
其中使用得比较多得是 $n $# $0 $? ,看看下面的例子:
#!/bin/sh
if [ $# -ne 2 ] ; then
echo "Usage: $0 string file";
exit 1;
fi
grep $1 $2 ;
if [ $? -ne 0 ] ; then
echo "Not Found \"$1\" in $2";
exit 1;
fi
echo "Found \"$1\" in $2";
上面的例子中使用了$0 $1 $2 $# $? 等变量

下面运行的例子:

./chapter2.2.sh usage chapter2.2.sh
Not Found "usage" in chapter2.2.sh
-bash-2.05b$ ./chapter2.2.sh Usage chapter2.2.sh
echo "Usage: $0 string file";
Found "Usage" in chapter2.2.sh
```

5. Strings

5.1. ##/#

```
$ MYVAR=foodforthought.jpg
$ echo ${MYVAR##*fo}
rthought.jpg
$ echo ${MYVAR#*fo}
odforthought.jpg
```

一个简单的脚本例子

```
mytar.sh

#!/bin/bash

if [ "${1##*.}" = "tar" ]
then
    echo This appears to be a tarball.
else
    echo At first glance, this does not appear to be a tarball.
fi

$ ./mytar.sh thisfile.tar
This appears to be a tarball.
$ ./mytar.sh thatfile.gz
At first glance, this does not appear to be a tarball.
```

5.2. %%/%

```
$ MYFOO="chickensoup.tar.gz"
$ echo ${MYFOO%%.*}
chickensoup
$ echo ${MYFOO%.*}
chickensoup.tar

MYFOOD="chickensoup"
$ echo ${MYFOOD%%soup}
chicken
```

5.3. :n1:n2

：${varible:n1:n2}:截取变量varible从n1到n2之间的字符串。

```
$ EXCLAIM=cowabunga
$ echo ${EXCLAIM:0:3}
cow
$ echo ${EXCLAIM:3:7}
abunga
```

5.4.

： ${varible:n1:n2}:截取变量varible从n1到n2之间的字符串。

上一页 上一级 下一页
4. 系统变量 起始页 6. Array

6. Array

定义数组

```
arr=(Hello World)

arr[0]=Hello
arr[1]=World
```

访问数组

```
echo ${arr[0]} ${arr[1]}

${arr[*]}         # All of the items in the array
${!arr[*]}        # All of the indexes in the array
${#arr[*]}        # Number of items in the array
${#arr[0]}        # Length of item zero
```

```
#!/bin/bash

array=(one two three four [5]=five)

echo "Array size: ${#array[*]}"

echo "Array items:"
for item in ${array[*]}
do
    printf "   %s\n" $item
done

echo "Array indexes:"
for index in ${!array[*]}
do
    printf "   %d\n" $index
done

echo "Array items and indexes:"
for index in ${!array[*]}
do
    printf "%4d: %s\n" $index ${array[$index]}
done
```

```
#!/bin/bash

array=("first item" "second item" "third" "item")

echo "Number of items in original array: ${#array[*]}"
for ix in ${!array[*]}
do
    printf "   %s\n" "${array[$ix]}"
done
echo

arr=(${array[*]})
echo "After unquoted expansion: ${#arr[*]}"
for ix in ${!arr[*]}
do
    printf "   %s\n" "${arr[$ix]}"
done
echo

arr=("${array[*]}")
echo "After * quoted expansion: ${#arr[*]}"
for ix in ${!arr[*]}
```

```
do
    printf "   %s\n" "${arr[$ix]}"
done
echo

arr=("${array[@]}")
echo "After @ quoted expansion: ${#arr[*]}"
for ix in ${!arr[*]}
do
    printf "   %s\n" "${arr[$ix]}"
done
```

```
array=({23..32} {49,50} {81..92})

echo "Array size: ${#array[*]}"

echo "Array items:"
for item in ${array[*]}
do
    printf "   %s\n" $item
done
```

7. eval

```
$ i=5
$ a_5=250
$ eval echo $"a_$i"
```

第 4 章 conditions if and case

目录

表 4.1. 文件目录表达式

Primary	意义
[-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
[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。

表 4.2. 字符串表达式

Primary	意义
[-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” 为 -eq, -ne, -lt, -le, -gt or -ge.

Arithmetic relational operators

1. -lt (<)
2. -gt (>)
3. -le (<=)
4. -ge (>=)
5. -eq (==)
6. -ne (!=)

表 4.3. 组合表达式

操作	效果
[! EXPR]	如果 EXPR 是false则为真。
[(EXPR)]	返回 EXPR的值。这样可以用来忽略正常的操作符优先级。
[EXPR1 -a EXPR2]	如果 EXPR1 and EXPR2 全真则为真。
[EXPR1 -o EXPR2]	如果 EXPR1 或者 EXPR2 为真则为真。

1. if

例 4.1. Basic conditional example if .. then

```
#!/bin/bash
if [ "foo" = "foo" ]; then
   echo expression evaluated as true
fi
```

例 4.2. Conditionals with variables

```
#!/bin/bash
T1="foo"
T2="bar"
if [ "$T1" = "$T2" ]; then
    echo expression evaluated as true
else
    echo expression evaluated as false
fi
```

```
(( $# != 1 )) && bool=0 || bool=${1}
```

```
[[ -f /tmp/test ]] && echo "True" || echo "False"
```

2. case

例 4.3. case

```
case "$1" in
        start)
            start
            ;;
        stop)
            stop
            ;;
        status)
            status
            ;;
        restart)
            stop
            start
            ;;
        condrestart)
            condrestart
            ;;

        *)
            echo $"Usage: $0 {start|stop|restart|condrestart|status}"
            exit 1
esac
```

第 5 章 Loops for, while and until

目录

1. for

```
#!/bin/bash
for i in 1 2 3 4 5
do
   echo "Welcome $i times"
done

for i in $( ls ); do
    echo item: $i
done

for i in `seq 1 10`;
do
    echo $i
done

for i in {1..5}
do
   echo "Welcome $i times"
done

for (( c=1; c<=5; c++ ))
do
        echo "Welcome $c times..."
done

for ((i=1; $i<=9;i++)); do echo $i; done
```

```
for i in {0..10..2}
do
    echo "Welcome $i times"
done

for i in $(seq 1 2 20)
do
   echo "Welcome $i times"
done
```

单行实例

```
for ip in {1..10};do echo $ip; done

for i in `seq 1 10`;do echo $i;done

for ip in {81..92}; do ssh root@172.16.3.$ip date; done

for n in  {23..32} {49,50} {81..92}; do mkdir /tmp/$n; echo $n; done
```

infinite loops

```
```

```
#!/bin/bash
for (( ; ; ))
do
   echo "infinite loops [ hit CTRL+C to stop]"
done
```

find file

```
#!/bin/bash
for file in /etc/*
do
        if [ "${file}" == "/etc/resolv.conf" ]
        then
                countNameservers=$(grep -c nameserver /etc/resolv.conf)
                echo "Total  ${countNameservers} nameservers defined in ${file}"
                break
        fi
done
```

backup file

```
#!/bin/bash
FILES="$@"
for f in $FILES
do
        # if .bak backup file exists, read next file
        if [ -f ${f}.bak ]
        then
                echo "Skiping $f file..."
                continue  # read next file and skip cp command
        fi
        # we are hear means no backup file exists, just use cp command to copy
file
        /bin/cp $f $f.bak
done
```

```
for n in  {23..32} {49,50} {81..92}; do mkdir /tmp/$n; echo $n; done
```

2. while

```
#!/bin/bash
COUNTER=0
while [  $COUNTER -lt 10 ]; do
    echo The counter is $COUNTER
    let COUNTER=COUNTER+1
done
```

```
while read name age
do
        echo $name $age
done << EOF
neo 30
jam 31
john 29
EOF


while read name age
do
        [[ $age -gt 30 ]] && echo $name
done << EOF
neo 30
jam 31
john 29
EOF
```

```
$ cat mount.sh
#!/bin/sh
while read LINE
do

        s=`echo $LINE|awk '{print $1}'`
        d=`echo $LINE|awk '{print $2}'`

        umount -f $d
        mount -t nfs4 $s $d

done < mount.conf

$ cat mount.conf
172.16.0.1:/    /www/logs/1
172.16.0.2:/    /www/logs/2
172.16.0.3:/    /www/logs/3
172.16.0.4:/    /www/logs/4
172.16.0.5:/    /www/logs/5
```

3. until

```
#!/bin/bash
COUNTER=20
until [  $COUNTER -lt 10 ]; do
    echo COUNTER $COUNTER
    let COUNTER-=1
done
```

第 6 章 Functions

目录

例 6.1. Functions with parameters sample

```
#!/bin/bash
function quit {
   exit
}
function e {
    echo $1
}
e Hello
e World
quit
echo foo
```

1. Local variables

```
 #!/bin/bash
        HELLO=Hello
        function hello {
                local HELLO=World
                echo $HELLO
        }
        echo $HELLO
        hello
        echo $HELLO
```

第 7 章 User interfaces

目录

例 7.1. Using select to make simple menus

```
#!/bin/bash
OPTIONS="Hello Quit"
select opt in $OPTIONS; do
    if [ "$opt" = "Quit" ]; then
     echo done
     exit
    elif [ "$opt" = "Hello" ]; then
     echo Hello World
    else
     clear
     echo bad option
    fi
done
```

例 7.2. Using the command line

```
#!/bin/bash
if [ -z "$1" ]; then
    echo usage: $0 directory
    exit
fi
SRCD=$1
TGTD="/var/backups/"
OF=home-$(date +%Y%m%d).tgz
tar -cZf $TGTD$OF $SRCD
```

例 7.3. Reading user input with read

In many ocations you may want to prompt the user for some input, and there are several ways to achive this. This is one of those ways:

```
#!/bin/bash
echo Please, enter your name
read NAME
echo "Hi $NAME!"
```

As a variant, you can get multiple values with read, this example may clarify this.

```
#!/bin/bash
echo Please, enter your firstname and lastname
read FN LN
echo "Hi! $LN, $FN !"
```

1. input

例 7.4. read

限时30秒内，输入你的名字

```
$ read -p "Please input your name: " -t 30 named
Please input your name: neo

$ echo $named

```

部分 II. Z Shell

目录

第 8 章 installing Z shell

$ sudo apt-get install zsh

第 9 章 Starting file

目录

1. ~/.zshrc

```
neo@netkiller:~$ cat .zshrc
# Created by newuser for 4.3.9
PROMPT='%n@%M:%~$ '

# enable color support of ls and also add handy aliases
if [ -x /usr/bin/dircolors ]; then
    eval "`dircolors -b`"
    alias ls='ls --color=auto'
    alias dir='dir --color=auto'
    alias vdir='vdir --color=auto'

    alias grep='grep --color=auto'
    alias fgrep='fgrep --color=auto'
    alias egrep='egrep --color=auto'
fi

# some more ls aliases
alias ll='ls -l'
alias la='ls -A'
alias l='ls -CF'

# Home/End/Del key
bindkey '\e[1~' beginning-of-line
bindkey '\e[4~' end-of-line
bindkey "\e[3~" delete-char
```

第 10 章 Prompting

```
$ PROMPT='%n@%M:%~$ '
neo@netkiller:~$
```

```
autoload colors; colors
export
PS1="%B[%{$fg[red]%}%n%{$reset_color%}%b@%B%{$fg[cyan]%}%m%b%{$reset_color%}:%~%B]%b
"
```

第 11 章 Aliases

```
# enable color support of ls and also add handy aliases
if [ -x /usr/bin/dircolors ]; then
    eval "`dircolors -b`"
    alias ls='ls --color=auto'
    alias dir='dir --color=auto'
    alias vdir='vdir --color=auto'

    alias grep='grep --color=auto'
    alias fgrep='fgrep --color=auto'
    alias egrep='egrep --color=auto'
fi

# some more ls aliases
alias ll='ls -l'
alias la='ls -A'
alias l='ls -CF'
```

第 12 章 History

```
$ !$
```

```
$ history
   18  cd workspace/Document
   19  ls
   20  ls

$ !20
ls
Docbook  makedoc  Tex
```

第 13 章 FAQ

目录

1. Home/End key

```
bindkey '\e[1~' beginning-of-line
bindkey '\e[4~' end-of-line
```

部分 III. Shell Commands

目录

第 14 章 Logging

目录

1. logger - a shell command interface to the syslog(3) system log module

```
# logger -p local0.notice -t HOSTIDM -f /dev/idmc
# tail /var/log/messages

# logger -p local0.notice -t passwd -f /etc/passwd
# tail /var/log/syslog

# logger -p user.notice -t neo -f /etc/passwd
# tail /var/log/syslog
# tail /var/log/messages

# logger -i -s -p local3.info -t passwd -f /etc/passwd
# tail /var/log/messages
```

第 15 章 screen - screen manager with VT100/ANSI terminal emulation

screen 类似 jobs, 前者是对terminal, 后者针对进程。你可以随时再次链接screen会话，而不用担心中途因网络不稳定造成的中断。

```
sudo apt-get install screen
```

进入

```
screen
```

查看任务

```
screen -ls
```

重新连接会话

```
screen -r 16582
```

退出screen 使用组合键 C-a K 或者

```
screen -wipe
```

第 16 章 Password

目录

1. Shadow password suite configuration.

```
# cat /etc/login.defs
# *REQUIRED*
#   Directory where mailboxes reside, _or_ name of file, relative to the
#   home directory.  If you _do_ define both, MAIL_DIR takes precedence.
#   QMAIL_DIR is for Qmail
#
#QMAIL_DIR      Maildir
MAIL_DIR        /var/spool/mail
#MAIL_FILE      .mail

# Password aging controls:
#
#       PASS_MAX_DAYS   Maximum number of days a password may be used.
#       PASS_MIN_DAYS   Minimum number of days allowed between password changes.
#       PASS_MIN_LEN    Minimum acceptable password length.
#       PASS_WARN_AGE   Number of days warning given before a password expires.
#
PASS_MAX_DAYS   99999
PASS_MIN_DAYS   0
PASS_MIN_LEN    5
PASS_WARN_AGE   7

#
# Min/max values for automatic uid selection in useradd
#
UID_MIN                  500
UID_MAX                60000

#
# Min/max values for automatic gid selection in groupadd
#
GID_MIN                  500
GID_MAX                60000

#
# If defined, this command is run when removing a user.
# It should remove any at/cron/print jobs etc. owned by
# the user to be removed (passed as the first argument).
#
#USERDEL_CMD    /usr/sbin/userdel_local

#
# If useradd should create home directories for users by default
# On RH systems, we do. This option is overridden with the -m flag on
# useradd command line.
#
CREATE_HOME     yes

# The permission mask is initialized to this value. If not specified,
# the permission mask will be initialized to 022.
UMASK           077

# This enables userdel to remove user groups if no members exist.
#
USERGROUPS_ENAB yes

# Use MD5 or DES to encrypt password? Red Hat use MD5 by default.
MD5_CRYPT_ENAB yes
```

```
ENCRYPT_METHOD MD5
```

2. newusers - update and create new users in batch

```
# cat userfile.txt
www00:x:520:520::/home/www00:/sbin/nologin
www01:x:521:521::/home/www01:/sbin/nologin
www02:x:522:522::/home/www02:/sbin/nologin
www03:x:523:523::/home/www03:/sbin/nologin
www04:x:524:524::/home/www04:/sbin/nologin
www05:x:525:525::/home/www05:/sbin/nologin
www06:x:526:526::/home/www06:/sbin/nologin
www07:x:527:527::/home/www07:/sbin/nologin
www08:x:528:528::/home/www08:/sbin/nologin
www09:x:529:529::/home/www09:/sbin/nologin

# newusers userfile.txt
```

3. chpasswd - update passwords in batch mode

echo "user:password" | chpasswd

```
[root@dev1 ~]# adduser test
[root@dev1 ~]# echo "test:123456" | chpasswd
```

```
# cat passwd.txt
neo:neopass
jam:jampass

# cat passwd.txt | chpasswd
```

```
# chpasswd -c < passwd.txt
```

passwd 命令实现相同功能

```
echo "mypasword" | passwd –stdin neo
```

第 17 章 date and time

目录

2010/06/18 17:57:38

```
$ date '+%Y/%m/%d %H:%M:%S'
```

2010-06-18 17:57:58

```
$ date '+%Y-%m-%d %H:%M:%S'
```

```
$ date +'%Y-%m-01 00:00:01'
2010-10-01 00:00:01
```

1. -d --date=

```
# date -d next-day +%Y%m%d
20060328
# date -d last-day +%Y%m%d
20060326
# date -d yesterday +%Y%m%d
20060326
# date -d tomorrow +%Y%m%d
20060328
# date -d last-month +%Y%m
200602
# date -d next-month +%Y%m
200604
# date -d next-year +%Y
2007
```

2. weekday name

```
date +%a
```

3. UTC

UTC time

```
$ datetime=$(date -u '+%Y%m%d %H:%M:%S')
$ echo $datetime
20091203 06:22:03
```

4. 日期偏移量

4.1. day

```
$ date -d '-1 day' +'%Y-%m-%d 00:00:01'
2010-10-14 00:00:01

$ date -d '+5 day' +'%Y-%m-%d 00:00:01'
2010-10-20 00:00:01
```

4.2. month

```
$ date -d '+2 month' +'%Y-%m-%d 00:00:01'
2010-12-15 00:00:01

$ date -d '-1 month' +'%Y-%m-%d 00:00:01'
2010-09-15 00:00:01
```

4.3. year

```
$ date -d '-5 year' +'%Y-%m-%d'
2005-10-15
$ date -d '+1 year' +'%Y-%m-%d'
2011-10-15
```

第 18 章 Numeric

目录

1. 数值运算

```
echo $((3+5))
expr 6 + 3
awk 'BEGIN{a=(3+2)*2;print a}'
```

2. seq - print a sequence of numbers

```
[neo@test ~]$ seq 10
1
2
3
4
5
6
7
8
9
10
[neo@test ~]$ seq 5 10
5
6
7
8
9
10
```

等差列, 步长设置

```
$ seq 1 1 10
1
2
3
4
5
6
7
8
9
10

$ seq 1 2 10
1
3
5
7
9

# seq 0 2 10
0
2
4
6
8
10
```

分隔符

```
# seq -s : -w 1 10
01:02:03:04:05:06:07:08:09:10

# seq -s '|' -w 1 10
01|02|03|04|05|06|07|08|09|10
```

等宽，前导字符用0填充

```
# seq -w 1 10
01
```

```
02
03
04
05
06
07
08
09
10
```

3. bc - An arbitrary precision calculator language

```
$ echo "4*5" | bc
```

```
# more calc.txt
3+2
4+5
8*2
10/4
# bc calc.txt
5
9
16
2
```

第 19 章 Text Processing

目录

1. iconv - Convert encoding of given files from one encoding to another

1.1. cconv - A iconv based simplified-traditional chinese conversion tool

cconv是建立在iconv之上，可以UTF8编码直接转换，并增加了词转换。

```
sudo apt-get install cconv
```

使用cconv进行简繁转换的方法为：

```
cconv -f UTF8-CN -t UTF8-HK zh-cn.txt -o zh-hk.txt
```

2. 字符串处理命令expr

字符串处理命令expr用法简介:
名称: expr
用途:求表达式变量的值。
语法: expr Expression
实例如下:
例子1:字串长度

```
shell>> expr length "this is a test content";
22
```

例子2:求余数

```
shell>> expr 20 % 9
2
```

例子3:从指定位置处截取字符串

```
shell>> expr substr "this is a test content" 3 5
is is
```

例子4:指定字符串第一次出现的位置

```
shell>> expr index "testforthegame" s
3
```

例子5:字符串真实重现

```
shell>> expr quote thisisatestformela
thisisatestformela
```

3. cat - concatenate files and print on the standard output

```
-b      不对空白行编号。
-e      使用 $ 字符显示行尾。
-n      从 1 开始对所有输出行编号。
-q      使用静默操作（禁止错误消息）。
-r      将所有多个空行替换为单行（“压缩”空白）。
-S      将多个空白行压缩到单行中（与 -r 相同）。
-s      禁止错误消息（静默操作）。
-t      将制表符显示为 ^I。
-u      不对输出进行缓冲。
-v      可视地显示非打印控制字符。
```

4. nl - number lines of files

```
$ nl /etc/issue
     1  CentOS release 5.4 (Final)
     2  Kernel \r on an \m
```

5. od - dump files in octal and other formats

5.1. 16进制

```
$ echo "helloworld" | od -x
```

6. tr - translate or delete characters

":"替换为"\n"

```
$ cat /etc/passwd |tr ":" "\n"
```

7. cut - remove sections from each line of files

列操作

```
$ last | grep  'neo' | cut -d ' ' -f1
```

```
$ cat /etc/passwd | cut -d ':' -f1
root
daemon
bin
sys
sync
games
man
lp
mail
news
uucp
proxy

$ cat /etc/passwd | cut -d ':' -f1,3,4
```

行操作

```
$ cat /etc/passwd | cut -c 1-4
root
daem
bin:
sys:
sync
game
man:

$ echo "No such file or directory"| cut -c4-7
such

$ echo "No such file or directory"| cut -c -8
No such

$ echo "No such file or directory"| cut -c-8
No such
```

8. printf - format and print data

```
printf "%d\n" 1234
```

```
$ printf "\033[1;33m TEST COLOR \n\033[m"
```

9. Free ‘recode’ converts files between various character sets and surfaces.

Following will convert text files between DOS, Mac, and Unix line ending styles:

```
$ recode /cl../cr <dos.txt >mac.txt
$ recode /cr.. <mac.txt >unix.txt
$ recode ../cl <unix.txt >dos.txt
```

10. 随机字符串

```
[neo@test .deploy]$ echo `< /dev/urandom tr -dc A-Z-a-z-0-9 | head -c 8`
GidAuuNN
[neo@test .deploy]$ echo `< /dev/urandom tr -dc A-Z-a-z-0-9 | head -c 8`
UyGaWSKr
```

我常常使用这样的随机字符初始化密码

```
[neo@test .deploy]$ echo `< /dev/urandom tr -dc [:alnum:] | head -c 8`
xig8Meym
[neo@test .deploy]$ echo `< /dev/urandom tr -dc [:alnum:] | head -c 8`
23Ac1vZg
[neo@test .deploy]$ echo `< /dev/urandom tr -dc [:digit:] | head -c 8`
73652314
[neo@test .deploy]$ echo `< /dev/urandom tr -dc [:graph:] | head -c 8`
GO_o>OnJ
[neo@test .deploy]$ echo `< /dev/urandom tr -dc [:graph:] | head -c 10`
iGy0FS/aO5
[neo@test .deploy]$ echo `< /dev/urandom tr -dc [:graph:] | head -c 50`
;`E^{5(T4v~5$YovW.?%_?9la<`+qPcRh@7mD\!Whx;MJZVQ\K
[neo@test .deploy]$ echo `< /dev/urandom tr -dc [:print:] | head -c 50`
fy$[#:'(')jt'gp1/g-)d~p]8 :r9i;MO2d!8M<?Qs3t:QgK$O
[neo@test .deploy]$ echo `< /dev/urandom tr -dc [:graph:] | head -c 50`
6SivJ5y$/FTi8mf}rrqE&s0"WkA}r;uK-=MT!Wp0UlL_lF0|bL
```

11. col

清除 ^M 字符

```
$ cat oldfile | col -b > newfile
```

12. apg - generates several random passwords

```
sudo apt-get install apg

$ apg

Please enter some random data (only first 16 are significant)
(eg. your old password):>
imlogNukcel5 (im-log-Nuk-cel-FIVE)
Drocdaf1 (Droc-daf-ONE)
fagJook0 (fag-Jook-ZERO)
heabugJer4 (heab-ug-Jer-FOUR)
5OsEsudy (FIVE-Os-Es-ud-y)
IrjOgneagOc9 (Irj-Og-neag-Oc-NINE)


$ apg -M SNCL -m 16
WoidWemFut6dryn,
byRowpEus-Flutt0
|QuogCagFaycsic0
ojHoadCyct4Freg_
Vir9blir`orhohoo
bapOip?Ibreawov2
```

13. head/tail

```
head -c 17 | tail -c 1
```

14. grep, egrep, fgrep, rgrep - print lines matching a pattern

14.1. -v, --invert-match

grep -v "grep"

```
[root@development ~]# ps ax | grep httpd
 6284 ?        Ss     0:10 /usr/local/httpd-2.2.14/bin/httpd -k start
 8372 ?        S      0:00 perl ./wrapper.pl -chdir -name httpd -class
com.caucho.server.resin.Resin restart
19136 ?        S      0:00 /usr/local/httpd-2.2.14/bin/httpd -k start
19749 pts/1    R+     0:00 grep httpd
31530 ?        Sl     0:57 /usr/local/httpd-2.2.14/bin/httpd -k start
31560 ?        Sl     1:12 /usr/local/httpd-2.2.14/bin/httpd -k start
31623 ?        Sl     1:06 /usr/local/httpd-2.2.14/bin/httpd -k start
[root@development ~]# ps ax | grep httpd | grep -v grep
 6284 ?        Ss     0:10 /usr/local/httpd-2.2.14/bin/httpd -k start
 8372 ?        S      0:00 perl ./wrapper.pl -chdir -name httpd -class
com.caucho.server.resin.Resin restart
19136 ?        S      0:00 /usr/local/httpd-2.2.14/bin/httpd -k start
31530 ?        Sl     0:57 /usr/local/httpd-2.2.14/bin/httpd -k start
31560 ?        Sl     1:12 /usr/local/httpd-2.2.14/bin/httpd -k start
31623 ?        Sl     1:06 /usr/local/httpd-2.2.14/bin/httpd -k start
```

14.2. 递归操作

递归查询

```
$ sudo grep -r 'neo' /etc/*
```

递归替换

```
for file in $( grep -rl '8800.org' *  | grep -v .svn ); do
    echo item: $file
        [ -f $file ] && sed -e 's/8800\.org/sf\.net/g' -e 's/netkiller/neo/g'
$file >$file.bak; cp $file.bak $file;
done
```

14.3. -c

```
$ cat /etc/resolv.conf
nameserver localhost
nameserver 208.67.222.222
nameserver 208.67.220.220
nameserver 202.96.128.166
nameserver 202.96.134.133
$ grep -c nameserver /etc/resolv.conf
5
```

14.4. regular

n 开头

```
$ grep '^n' /etc/passwd
news:x:9:9:news:/var/spool/news:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
```

```
neo:x:1000:1000:neo chan,,,:/home/neo:/bin/bash
nagios:x:116:127::/var/run/nagios2:/bin/false
```

bash 结尾

```
$ grep 'bash$' /etc/passwd
root:x:0:0:root:/root:/bin/bash
neo:x:1000:1000:neo chan,,,:/home/neo:/bin/bash
postgres:x:114:124:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash
cvsroot:x:1001:1001:cvsroot,,,,:/home/cvsroot:/bin/bash
svnroot:x:1002:1002:subversion,,,,:/home/svnroot:/bin/bash
```

中间包含 root

```
$ grep '.*root' /etc/passwd
root:x:0:0:root:/root:/bin/bash
cvsroot:x:1001:1001:cvsroot,,,,:/home/cvsroot:/bin/bash
svnroot:x:1002:1002:subversion,,,,:/home/svnroot:/bin/bash
```

```
ps ax |grep -E "mysqld|httpd|resin"
```

14.4.1. 2010:(13|14|15|16)

regular 匹配一组

```
egrep "2010:(13|14|15|16)"  access.2010-11-18.log > apache.log
```

14.5. ^M

```
fgrep -rl `echo -ne '\r'` .

find . -type f -exec grep $'\r' {} +
```

14.6. egrep

```
$ snmpwalk -v2c -c public 172.16.1.254 | egrep -i 'if(in|out)'

for pid in $(ps -axf |grep 'php-cgi' | egrep egrep "0:00.(6|7|8|9)"'{print $1}');
do kill -9 $pid; done

for pid in $(ps -axf |grep 'php-cgi' | egrep "0:(0|1|2|3|4|5)0.(6|7|8|9)" |awk
'{print $1}'); do kill -9 $pid; done
```

15. sort - sort lines of text files

```
$ du -s * | sort -k1,1rn
```

```
$ rpm -q -a --qf '%10{SIZE}\t%{NAME}\n' | sort -k1,1n
$ dpkg-query -W -f='${Installed-Size;10}\t${Package}\n' | sort -k1,1n
```

15.1. 对列排序

```
# sort -t ':' -k 1 /etc/passwd
```

```
ort -n -t ‘ ‘ -k 2 file.txt
```

多列排序

```
$ sort -n -t ‘ ‘ -k 2 -k 3 file.txt
```

16. ed, red - text editor

行寻址

```
.       此选项对当前行寻址（缺省地址）。
number  此选项对第 number 行寻址。可以按逗号分隔的范围 (first,last) 对行寻址。0 代表缓冲区的开头（第一行之前）。
-number 此选项对当前行之前的第 number 行寻址。如果没有 number，则减号对紧跟在当前行之前的行寻址。
+number 此选项对当前行之后的第 number 行寻址。如果没有 number，则加号对紧跟在当前行之后的行寻址。
$       此选项对最后一行寻址。
,       此选项对第一至最后一行寻址，包括第一行和最后一行（与 1,$ 相同）。
;       此选项对当前行至最后一行寻址。
/pattern/       此选项对下一个包含与 pattern 匹配的文本的行寻址。
?pattern?       此选项对上一个包含与 pattern 匹配的文本的行寻址。
```

命令描述

```
a       此命令在指定的地址之后追加文本。
c       此命令将指定的地址更改为给定的文本。
d       此命令删除指定地址处的行。
i       此命令在指定的地址之前插入文本。
q       此命令在将缓冲区保存到磁盘后终止程序并退出。
r file  此命令读取 filespec 的内容并将其插入指定的地址之后。
s/pattern/replacement/  此命令将匹配 pattern 的文本替换为指定地址中的 replacement 文本。
w file  此命令将指定的地址写到 file。如果没有 address，则此命令缺省使用整个缓冲区。
```

实例，删除passwd中的neo用户

```
ed -s passwd <<EOF
/neo/
d
wq
EOF
```

```
ed -s mfsmetalogger.cfg <<EOF
,s/^# //
wq
EOF
```

删除尾随空格

```
$ cat -vet input.txt

This line has trailing blanks.    $
This line does not.$

$ (echo ',s/ *$//'; echo 'wq') | ed -s input.txt

$ cat -vet input.txt

This line has trailing blanks.$
This line does not.$
```

17. vim

test script

```
vim test.txt <<end > /dev/null 2>&1
:%s/neo/neo chen/g
:%s/hello/hello world/g
:wq
end
```

test.txt

```
begin
neo
test
hello
world
end
```

test result

```
$ ./test
$ cat test.txt
begin
neo chen
test
hello world
world
end
neo@netkiller:/tmp$
```

17.1. vi 批处理

```
for i in file_list
do
vi $i <<-!
:g/xxxx/s//XXXX/g
:wq
!
done
```

17.2. line()

加入行号

```
:g/^/ s//\=line('.').' '/
```

18. awk

18.1. 处理列

```
# cat /etc/fstab | awk '{print $1}'
```

18.2. 查找文件并删除

```
#!/bin/sh
LOCATE=/home/samba
find $LOCATE -name '*.eml'>log
find $LOCATE -name '*.nws'>>log
gawk '{print "rm -rf "$1}' log > rmfile
chmod 755 rmfile
./rmfile
```

18.3. TCP/IP Status

```
netstat -n | awk '/^tcp/ {++S[$NF]} END {for(a in S) print a, S[a]}'

TIME_WAIT 88
CLOSE_WAIT 6
FIN_WAIT1 9
FIN_WAIT2 9
ESTABLISHED 303
SYN_RECV 126
LAST_ACK 5
```

18.4. 字符匹配

```
输出包含（不包含）特定字符的行（sed也可以完成该功能）：
neo@monitor:~$ awk '/[a-c]/ { print }' file.txt
daemon x 1 1 daemon /usr/sbin /bin/sh
bin x 2 2 bin /bin /bin/sh
sys x 3 3 sys /dev /bin/sh
sync x 4 65534 sync /bin /bin/sync
games x 5 60 games /usr/games /bin/sh
man x 6 12 man /var/cache/man /bin/sh
lp x 7 7 lp /var/spool/lpd /bin/sh
mail x 8 8 mail /var/mail /bin/sh
news x 9 9 news /var/spool/news /bin/sh
uucp x 10 10 uucp /var/spool/uucp /bin/sh
proxy x 13 13 proxy /bin /bin/sh
www-data x 33 33 www-data /var/www /bin/sh
backup x 34 34 backup /var/backups /bin/sh
list x 38 38 Mailing List Manager /var/list /bin/sh
irc x 39 39 ircd /var/run/ircd /bin/sh
gnats x 41 41 Gnats Bug-Reporting System (admin) /var/lib/gnats /bin/sh
nobody x 65534 65534 nobody /nonexistent /bin/sh
libuuid x 100 101  /var/lib/libuuid /bin/sh
syslog x 101 103  /home/syslog /bin/false
sshd x 102 65534  /var/run/sshd /usr/sbin/nologin
landscape x 103 108  /var/lib/landscape /bin/false
mysql x 104 112 MySQL Server,,, /var/lib/mysql /bin/false
ntpd x 105 114  /var/run/openntpd /bin/false
postfix x 106 115  /var/spool/postfix /bin/false
nagios x 107 117  /var/lib/nagios /bin/false
chun x 1003 1003 Li Fu Chun,,, /home/chun
munin x 108 118  /var/lib/munin /bin/false
```

```
$ awk '!/[a-c]/ { print }' file.txt
root x 0 0 root /root
neo x 1000 1000 neo,,, /home/neo


采用判断来输出特定的列数据:
neo@monitor:~$ sed -e 's/:/ /g' /etc/passwd | awk '$1 == "neo" { print $1 }'
neo

部分包含，不包含指定的字符:
$ awk '$1 ~ /[a-d]/ { print }' file.txt
$ awk '$1 !~ /[a-d]/ { print }' file.txt
```

18.5. NF

```
uptime | awk '{print $(NF-2)}'
```

18.6. 过滤相同的行

```
grep 'Baiduspider' access.2011-02-22.log | awk '{print $1}' | awk '! a[$0]++'
```

```
awk '! a[$0]++' 1.txt >2.txt
这个是删除文件中所有列都重复的记录

awk '! a[$1]++' 1.txt >2.txt
删除文件中第一列重复的记录

awk '! a[$1,$2]++' 1.txt >2.txt
删除文件中第一，二列都重复的记录
```

19. sed

http://sed.sourceforge.net/

```
sed -i -e 's/aaa/bbb/g' *
perl -p -i -e 's/aaa/bbb/g' *
```

19.1. find and replace

```
ls -1 *.html| awk '{printf "sed \047s/ADDRESS/address/g\047 %s >%s.sed;mv %s.sed
%s\n", $1, $1, $1, $1;}'|bash

for f in `ls -1 *.html`; do [ -f $f ] && sed 's/<\/BODY>/<script
src="http:\/\/www.google-analytics.com\/urchin.js"
type="text\/javascript"><\/script>\n<script type="text\/javascript">\n_uacct =
"UA-2033740-1";\nurchinTracker();\n<\/script>\n<\/BODY>/g' $f >$f.sed;mv $f.sed $f
; done;
```

```
my=/root/dir
str="/root/dir/file1 /root/dir/file2 /root/dir/file3 /root/dir/file/file1"
echo $str | sed "s:$my::g"
```

19.1.1. 正则

```
sed s/[[:space:]]//g  filename          删除空格
```

19.1.2. delete

```
sed /^$/d         filename        删除空行
```

19.1.3. aaa="bbb" 提取bbb

```
$ echo "aaa=\"bbb\"" | sed 's/.*=\"\(.*\)\"/\1/g'
```

第 20 章 CURL

目录

1. 基本用法

```
curl -o /dev/null -s -w %{time_connect}:%{time_starttransfer}:%{time_total}
http://www.example.net
```

测试页面所花费的时间

```
date ; curl  -s  -w 'Connect: %{time_connect} TTFB: %{time_starttransfer} Total
time: %{time_total} \n'  -H "Host: www.example.com"
http://172.16.0.1/webapp/test.jsp ; date ;
```

2. connect-timeout

```
curl -o /dev/null --connect-timeout 30 -m 30 -s -w %{http_code}
http://www.google.com/
```

3. vhosts

有时候你需要设觉察/etc/hosts文件才能访问vhost,下面例子可以不设置/etc/hosts

```
curl -x 127.0.0.1:80 your.exmaple.com/index.php
```

4. http status

```
curl -s -I http://netkiller.sourceforge.net/ | grep HTTP | awk '{print $2" "$3}'
curl -o /dev/null -s -w %{http_code} http://netkiller.sourceforge.net/
```

5. referer

```
curl -v -o /dev/null -e "http://www.example.com" http://www.your.com/
* About to connect() to www.your.com port 80
*   Trying 172.16.1.10... connected
* Connected to www.your.com (172.16.1.10) port 80
> GET / HTTP/1.1
> User-Agent: curl/7.15.5 (x86_64-redhat-linux-gnu) libcurl/7.15.5 OpenSSL/0.9.8b
zlib/1.2.3 libidn/0.6.5
> Host: www.your.com
> Accept: */*
> Referer: http://www.example.com
>
< HTTP/1.1 200 OK
< Date: Thu, 30 Sep 2010 07:59:47 GMT
< Server: Apache/2.2.16 (Unix) mod_ssl/2.2.16 OpenSSL/0.9.8e-fips-rhel5 PHP/5.2.14
< Accept-Ranges: bytes
< Transfer-Encoding: chunked
< Content-Type: text/html; charset=UTF-8
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  172k    0  172k    0     0  10.2M      0 --:--:-- --:--:-- --:--:-- 11.9M*
Connection #0 to host www.your.com left intact

* Closing connection #0
```

6. -v

7. -H/--header <line> Custom header to pass to server (H)

7.1. Last-Modified / If-Modified-Since

If-Modified-Since

```
neo@neo-OptiPlex-780:/tmp$ curl -I http://images.example.com/test/test.html
HTTP/1.0 200 OK
Cache-Control: s-maxage=7200, max-age=900
Expires: Mon, 16 May 2011 08:10:37 GMT
Content-Type: text/html
Accept-Ranges: bytes
ETag: "1205579110"
Last-Modified: Mon, 16 May 2011 06:57:39 GMT
Content-Length: 11
Date: Mon, 16 May 2011 07:55:37 GMT
Server: lighttpd/1.4.26
Age: 604
X-Via: 1.0 ls71:80 (Cdn Cache Server V2.0), 1.0 lydx136:8105 (Cdn Cache Server
V2.0)
Connection: keep-alive

neo@neo-OptiPlex-780:/tmp$ curl -H "If-Modified-Since: Fri, 12 May 2011 18:53:33
GMT"  -I http://images.example.com/test/test.html
HTTP/1.0 304 Not Modified
Date: Mon, 16 May 2011 07:56:19 GMT
Content-Type: text/html
Expires: Mon, 16 May 2011 08:11:19 GMT
Last-Modified: Mon, 16 May 2011 06:57:39 GMT
ETag: "1205579110"
Cache-Control: s-maxage=7200, max-age=900
Age: 790
X-Via: 1.0 wzdx168:8080 (Cdn Cache Server V2.0)
Connection: keep-alive
```

7.2. ETag / If-None-Match

```
neo@neo-OptiPlex-780:/tmp$ curl -I http://images.example.com/test/test.html
HTTP/1.1 200 OK
Cache-Control: s-maxage=7200, max-age=900
Expires: Mon, 16 May 2011 09:48:45 GMT
Content-Type: text/html
Accept-Ranges: bytes
ETag: "1984705864"
Last-Modified: Mon, 16 May 2011 09:01:07 GMT
Content-Length: 22
Date: Mon, 16 May 2011 09:33:45 GMT
Server: lighttpd/1.4.26
```

```
neo@neo-OptiPlex-780:/tmp$ curl -H 'If-None-Match: "1984705864"' -I
http://images.example.com/test/test.html
HTTP/1.1 304 Not Modified
Cache-Control: s-maxage=7200, max-age=900
Expires: Mon, 16 May 2011 09:48:32 GMT
Content-Type: text/html
Accept-Ranges: bytes
ETag: "1984705864"
Last-Modified: Mon, 16 May 2011 09:01:07 GMT
Date: Mon, 16 May 2011 09:33:32 GMT
Server: lighttpd/1.4.26
```

7.3. Accept-Encoding:gzip,defalte

```
$ curl -H Accept-Encoding:gzip,defalte -I
http://www.example.com/product/374218.html
HTTP/1.1 200 OK
Date: Mon, 16 May 2011 09:13:18 GMT
Server: Apache
Accept-Ranges: bytes
Content-Encoding: gzip
Content-Length: 16660
Content-Type: text/html; charset=UTF-8
X-Pad: avoid browser bug
Age: 97
X-Via: 1.1 dg44:8888 (Cdn Cache Server V2.0)
Connection: keep-alive
```

```
$ curl -H Accept-Encoding:gzip,defalte http://www.example.com/product/374218.html
| gunzip
```

7.4. HOST

```
curl -H HOST:www.example.com -I http://172.16.1.10/product/374218.html
```

第 21 章 expect

目录


```
$ sudo apt-get install expect
```

1. 模拟登录 ssh

例 21.1. example for expect

```
#!/usr/bin/expect
set timeout 30
spawn ssh root@192.168.1.1
expect "password:"
send "mypassword\r"
interact
```

例 21.2. example 1

```
#!/usr/bin/expect
  set password 1234  #密码
  #download
  spawn scp /www/* root@172.16.1.2:/www/
  set timeout 300
  expect "172.16.1.2's password:"
  set timeout 3000
  #exec sleep 1
  send "$password\r"
  set timeout 300
  send "exit\r"
  #expect eof
  interact
  spawn scp /www/* root@172.16.1.3:/www/
  set timeout 300
  expect "root@172.16.1.3's password:"
  set timeout 3000
  #exec sleep 1
  send "$password\r"
  set timeout 300
  send "exit\r"
  interact
```

例 21.3. *.exp

```
$ expect autossh.exp neo@192.168.3.10 chen "ls /"
```

autossh.exp

```
#!/usr/bin/expect -fset ipaddress [lindex $argv 0]
set ipaddress [lindex $argv 0]
```

```
set passwd [lindex $argv 1]
set command [lindex $argv 2]
set timeout 30
spawn ssh $ipaddress
expect {
    "yes/no" { send "yes\r";exp_continue }
    "password:" { send "$passwd\r" }
}
expect ""

send "$command \r"

send "exit\r"

expect eof
exit
```

批量执行

```
password.lst
192.168.0.1 passwd
192.168.0.2 passwd
192.168.0.3 passwd
```

```
#!/bin/bash

cat password.lst | while read line
do
        host=$(echo $line|awk '{print $1}')
        passwd=$(echo $line|awk '{print $2}')
        expect autossh.exp $host $passwd
        sleep 2
done
```

2. SCP

```
#! /usr/bin/expect -f
spawn scp 1 neo@192.168.0.1:
expect "*password:"
send "your password\r"

expect eof
```

```
#!/bin/expect

spawn scp x.x.x.x

for {} {1} {} {
        expect {
                "password:" {
                send "YourPassWord"
            }
        }
}
```

```
spawn scp 1 neo@172.16.0.1:
for { set i 1 } {$i<5} {incr i} {
        expect {
                "*password:" {send "koven\r"}
                "*(yes/no)*" {send "yes\r"}
        }
}
```

第 22 章 TUI

目录

1. dialog

```
neo@netkiller:~$ sudo apt-get install dialog
```

2. tput

第 23 章 Example

目录

1. 有趣的Shell

运行后会不停的fork新的进程，直到你的资源消耗尽。

```
:() { :|:& }; :

.() { .|.& }; .
```

2. backup

```
#!/bin/sh
umount /mnt/backup
mount /dev/sdb1 /mnt/backup

if [ `date +%d` = '01' ] #每月1号进行完全备份
then
        bakdir="/mnt/bak/daybak/month/"`date +%m%d`
        zl="" #进行完全备份
else
        backup_dir="/mnt/backup/"`date +%d`
        zl="-N "`date +'%Y-%m-01 00:00:01'`; #差异备份
        #zl="-N "`date -d '-1 day' +'%Y-%m-%d 00:00:01'` #日增量备份
fi

tar "${zl}" -czf ${backup_dir}/www.tgz /var/www
umount /mnt/backup
```

3. CPU 核心数

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

```
Password
```

例 23.1. random password

```
cat /dev/urandom | head -1 | md5sum | head -c 8

od -N 4 -t x4 /dev/random | head -1 | awk '{print $2}'
```

5. processes

5.1. pid

```
neo@debian:~/html/temp$ pidof lighttpd
2775

neo@debian:~/html/temp$ pgrep lighttpd
2775

neo@debian:~/html/temp$ pid=`pidof lighttpd`
neo@debian:~/html/temp$ echo $pid
2775
```

```
# user=`whoami`
# pgrep -u $user -f cassandra | xargs kill -9
```

5.2. kill

kill 占用7800端口的进程

```
kill -9 `netstat -nlp | grep '192.168.0.5:7800' | awk -F ' ' '{print $7}' | awk -
F '/' '{print $1}'`
```

5.3. pgrep

```
#!/bin/bash
ntpdate 172.16.10.10

pid=$(pgrep rsync)

if [ -z "$pid" ]; then

rsync -auzP --delete -e ssh  --exclude=zoshow/images --exclude=project/product --
exclude=project/templates/caches  root@172.16.10.10:/www/project /www

fi
```

6. Shell 技巧

6.1. 行转列，再批评

```
echo "abc def gfh ijk"| sed "s:\ :\n:g" |grep -w gfh
```

6.2. for vs while

```
echo "aaa bbb ccc" > test.txt
echo "ddd eee fff" >> test.txt
```

```
for line in $(cat test.txt)
do
        echo $line
done
```

```
cat test.txt| while read line
do
        echo $line
done
```

6.3. 遍历字符串

```
# find . -name "*.html" -o -name "*.php" -o -name '*.dwt' -printf "[%p] " -exec
grep -c 'head' {} \; | grep -v "0$" |more
```

附录 A. 附录

目录

1. Linux 下载排名

http://distrowatch.com/

2. to convert utf-8 from gb2312 code

```
perl   -   MEncode   -   pi   -   e   '   $_=encode_utf8(decode(gb2312=>$_))   '
filename
for f in `find .`; do [ -f $f ] && perl -MEncode -pi -e
'$_=encode_utf8(decode(gb2312=>$_))' $f; done;
```

3. 使用内存的百分比

```
$ free | sed -n 2p | awk '{print "used="$3/$2*100"%","free="$4/$2*100"%"}'
used=53.9682% free=46.0318%
```

4. 合并apache被cronlog分割的log文件

```
$ find 2009 -type f -name access.log -exec cat {} >> access.log \;
```

5. 参考文献

《高级Bash脚本编程指南》http://www.linuxsir.org/main/doc/abs/abs3.7cnhtm/index.html