Netkiller Linux Shell 手札

netkiller Neo Chan

2009-11-15

文档最近一次更新于 Sat May 8 05:59:05 UTC 2010

系列文档

下面是我多年积累下来的经验整理文档供大家参考:

Netkiller Linux Basics 手札 | Netkiller Linux Advanced 手札 | Netkiller CentOS 手札 | Netkiller FreeBSD 手札 | Netkiller Shell 手札

Netkiller Version 手札 | Netkiller Developer 手札 | Netkiller Security 手札 | Netkiller Database 手札 | Netkiller LDAP 手札

Netkiller Architect 手札 | Netkiller Intranet 手札 | Netkiller Cisco IOS 手札 | Netkiller Mail System 手札 | Netkiller Document 手札

目录

表格清单

范例清单

下一页

About author

About author

上一页 下一页

About author

目录

作者简介

主页地址：http://netkiller.sourceforge.net, http://netkiller.hikz.com, http://netkiller.8800.org

陈景峰 (Neo chen) IT民工，昵称：netkiller, UNIX like爱好者,Senior PHP Software Engineer, 业余无线电爱好者（呼号：BG7NYT）,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,买不起房，买不起车，辛辛苦苦几十年，一下回到解放前

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

上一页 下一页

Netkiller Linux Shell 手札 起始页 联系作者

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: 问我
MSN: 问我
G Talk: 问我
网易泡泡：openunix

写给火腿:

也同样欢迎无线电爱好者和我QSO,我的QTH在深圳龙华苹果园10F,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

部分 I. Shell

目录

第 **1** 章 **Strings**

目录

/

```
$ 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.
```

上一页 上一级 下一页
部分 I. Shell 起始页 %%/%

% % / %

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

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

: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
```

#

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

第 2 章 I/O 重定向

目录


```
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
```

error 重定向

```
your_shell 2>&1
```

上一页 上一级 下一页
起始页 使用块记录日志

使用块记录日志

使用块记录日志

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

第 **3** 章 **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 **II. Bash Shell**

上一页 下一页

部分 II. Bash Shell

目录

上一页 下一页

第 3 章 pipes (FIFOs) 起始页 第 4 章 prompt

第 **4** 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 **4.1. 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 **4.2. 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 **4.3. 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 "

}
```

上一页 上一级 下一页
部分 II. Bash Shell 起始页 第 5 章 variable

第 **5** 章 **variable**

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
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
```

第 6 章 conditions if and case

目录

表 6.1. 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 **6.2.** 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 **6.3.** 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。

if

例 **6.1. Basic conditional example if .. then**

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

例 **6.2. 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
```

第 5 章 variable 起始页 case

case

例 **6.3. 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 **7** 章 **Loops for, while and until**

目录

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
```

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
```

while

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

until

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

第 8 章 Functions

目录

例 **8.1. Functions with parameters sample**

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

Local variables

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

上一页 上一级 下一页
until 起始页 第 9 章 User interfaces

第 9 章 User interfaces

例 9.1. 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 9.2. 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 9.3. 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 !"
```

部分 **III. Z Shell**

上一页 下一页

部分 III. Z Shell

目录

http://www.zsh.org/

上一页 下一页
第 9 章 User interfaces 起始页 第 10 章 installing Z shell

第 **10** 章 **installing Z shell**

$ sudo apt-get install zsh

第 **11** 章 **Starting file**

目录

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

上一页 上一级 下一页
第 10 章 installing Z shell 起始页 第 12 章 Prompting

第 **12** 章 **Prompting**

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

第 **13** 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 **14** 章 **History**

```
$ !$
```

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

$ !20
ls
Docbook  makedoc  Tex
```

第 **15** 章 **FAQ**

目录

Home/End key

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

部分 **IV. Commands**

上一页 下一页

部分 IV. Commands

目录

上一页 第 15 章 FAQ

起始页

下一页 第 16 章 chsh - change login shell

第 16 章 chsh - change login shell

```
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
```

第 **17** 章 **Directory and File System Related**

目录

test - check file types and compare values

```
test -x /usr/bin/munin-cron && /usr/bin/munin-cron
```

xargs

```
find /etc -type f|xargs md5sum
```

```
find ./ -name "*html" | xargs -n 1 sed -i -e 's/aaa/bbb/g'
```

find

替换文本

```
# find ./ -exec grep str1 '{}' \; -exec sed -i.bak s/str1/str2/g '{}' \;
```

分别设置文件与目录的权限

```
find /usr/www/phpmyadmin -type d -exec chmod 755 {} \;
find /usr/www/phpmyadmin -type f -exec chmod 644 {} \;
```

第 **18** 章 **Text Processing**

目录

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。

上一页 find
上一级 起始页
下一页 nl - number lines of files

nl - number lines of files

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

tr - translate or delete characters

":"替换为"\n"

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

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

递归查询

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

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
```

-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
```

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
```

regular

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

上一页 tr - translate or delete characters
上一级 起始页
下一页 ed, red - text editor

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
```

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.$
```

上一页 grep, egrep, fgrep, rgrep - print lines matching a pattern

上一级

起始页

下一页 sort - sort lines of text files

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
```

printf - format and print data

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

Free `recode' converts files between various character sets and surfaces.

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
```

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
```

Free `recode' converts files between various character sets and surfaces.

起始页

awk

awk

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
```

sed

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

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;


```

第 **19** 章 **Numeric**

目录

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

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

上一页 上一级 下一页
sed 起始页 第 20 章 date and time

第 **20** 章 **date and time**

目录


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

UTC

UTC time

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

第 **21** 章 **processes**

目录

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
```

ps

```
[root@development ~]# ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
```

第 22 章 Logging

目录

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 23 章 TUI

上一页 下一页

第 23 章 TUI

目录

dialog

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

上一页 下一页
第 22 章 Logging 起始页 tput

tput

附录 **A.** 附录

上一页 下一页

附录 **A.** 附录

目录

Linux 下载排名

http://distrowatch.com/

上一页 下一页

tput 起始页 to convert utf-8 from gb2312 code

to convert utf-8 from gb2312 code

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;
```

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%
```

合并**apache**被**cronlog**分割的**log**文件

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

vi 批处理

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

参考文献

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