# 第27章　几个脚本例子

本章包含了我最常用的几个脚本。你会发现它们都相当短小而简单。这就是脚本的一个优点；它不是很长、很复杂，只需很短的代码就能够完成相当多的功能，可以节约大量的时间。

本章中包含以下内容：

• 各种脚本的例子。

我本来打算在本章中提供一个通用的数据验证数据库脚本，但是由于它超过了 500行，我觉得编辑肯定不会同意把它收入书中。那个脚本几年前只有几行，后来由于不断增加功能，变成了现在这么长。最后，我选择了如下六个脚本作为例子：

pingall：一个按照/etc/hosts文件中的条目逐一ping所有主机的脚本。

backup_gen：一个通用的备份脚本，能够加载缺省设置。

del.lines：一个引用sed命令的脚本，能从文件中删除若干行。

access_deny：一个能够阻止某些特定用户登录的工具。

logroll：一个能够清除超过某一长度的日志的工具。

nfsdown：一个快速unmount所有nfs文件系统的工具。

## 27.1　pingall

几年前我写了一个名为 pingall的脚本在夜间运行，把它作为常规报告脚本的一部分。它能够按照/etc/hosts文件中的条目逐一ping所有的主机。

该脚本列出/etc/hosts文件并查找其中的非注释行（不以 #开头的行）。然后使用一个 while循环读入所有的行，接下来使用 awk分析出每行的第一个域，并把它赋给变量 ADDR。最后使用for循环逐一ping相应的地址。

下面就是该脚本。

```
$ pg pingall
#!/bin/sh
# pingall

# grab /etc/hosts and ping each address
cat /etc/hosts| grep -v '^#' | while read LINE
do
  ADDR=`awk '{print $1}'`
  for MACHINE in $ADDR
  do
    ping -s -c1 $MACHINE
  done
done
```

上述脚本可以很容易地进行扩展，加进其他网络报告工具。

## 27.2　backup_gen

在本章中我选择了这个脚本并不是因为它展示了如何备份目录，而是因为它是一个同其

他脚本共享设置的很好例子。

backup_gen是一个用于备份的脚本，它从一个缺省的配置文件中读入设置，然后根据这些参数对系统进行备份。用户可以根据自己的需要改变这些缺省设置。这是一个不同脚本如何使用相同设置或仅在自己运行期间改变相应设置的极好例子。当该脚本执行时，它首先确认源文件backup.defaults是否存在，如果不存在，则退出。

该脚本在运行时，会显示出一个题头和缺省设置，并询问用户是否需要改变任何缺省设置。如果用户回答“是”，在他们修改设置之前，该脚本就会提示他们输入一个代码，用户可以有三次机会；如果输入正确的代码后仍无法改变设置，这就意味着用户必须要使用缺省设置。一般来说，在输入正确代码后，用户可以改变下列设置（[ ]中的为缺省设置）：

- 磁带设备[rmt0] 可以选择rmt1和rmt3
- 备份完成后是否向系统管理员发邮件[是] 可以选择否
- 备份的类型[全备份] 可以选择普通备份或sybase备份

脚本中使用了一些临时变量来保存被修改的设置。用户可以按回车键选择缺省设置。下列设置不能被改变：

备份日志文件名。

用户代码。

接着所有的改变会生效。在这些改变生效之后，相应的临时变量又会被重新赋予缺省值。在备份进行之前，首先要测试磁带设备。备份过程使用 find和cpio命令，它们从设置文件中读入相应变量的缺省值，或使用用户设定的值。

下面就是该脚本。

```
$ pg backup_run
#!/bin/sh
# backup_run

# script to run the backups
# loads in a setting file for the user to change

SOURCE=/appdva/bin/backup.defaults
check_source()
{
# check_source
# can we load the file
# backup.defaults is the source file containing config/functions
# make sure your path includes this directory you are running from
if [ -r $SOURCE ]; then
  . /$SOURCE
else
  echo "`basename $0`: cannot locate defaults file"
  exit 1
fi
}

header()
{
# header
USER=`whoami`
MYDATE=`date +%A" "%e" of "%B-%Y`
```

```
clear
cat << MAYDAY
User : $USER                                                    $MYDATE
                          NETWORK SYSTEM BACKUP
                          =====================
MAYDAY
}

change_settings()
{
# change_settings
# let the user see the default settings..
header
echo "Valid Entries Are..."
echo "Tape Device: rmt0, rmt1, rmt3"
echo "Mail Admin: yes, no"
echo "Backup Type: full, normal, sybase "
while :
do
  echo -n -c "\n\n Tape Device To Be Used For This Backup  [$_DEVICE] :"
  read T_DEVICE
  : ${T_DEVICE:=$_DEVICE}
  case $T_DEVICE in
  rmt0|rmt1|rmt3) break;;
  *) echo "The devices are either ... rmt0, rmt1, rmt3"
    ;;
  esac
done

# if the user hits return on any of the fields, the default value will be
used
while :
do
  echo -n " Mail Admin When Done                  [$_INFORM] :"
  read T_INFORM
  : ${T_INFORM:=$_INFORM}
  case $T_INFORM in
  yes|Yes) break;;
  no|No) break;;
  *) echo "The choices are yes, no"
    ;;
  esac
done

while :
do
  echo -n " Backup Type                             [$_TYPE] :"
  read T_TYPE
  : ${T_TYPE:=$_TYPE}
  case $T_TYPE in
  Full|full) break;;
  Normal|normal)break;;
  Sybase|sybase)break;;
  *) echo "The choices are either ... full, normal, sybase"
```

```
  esac
done
# re-assign the temp variables back to original variables that
# were loaded in
_DEVICE=$T_DEVICE; _INFORM=$T_INFORM; _INFORM=$T_INFORM
}

show_settings()
# display current settings
{
cat << MAYDAY
                        Default Settings Are...
                    Tape Device To Be Used       : $_DEVICE
                    Mail Admin When Done         : $_INFORM
                    Type Of Backup               : $_TYPE
                    Log file of backup           : $_LOGFILE
MAYDAY
}

get_code()
{
# users get 3 attempts at entering the correct code
# _CODE is loaded in from the source file
clear
header
_COUNTER=0
echo " YOU MUST ENTER THE CORRECT CODE TO BE ABLE TO CHANGE
     DEFAULT SETTINGS"
while :
do
  _COUNTER=`expr $_COUNTER + 1`
  echo -n "Enter the code to change the settings  :"
  read T_CODE
  # echo $_COUNTER
  if [ "$T_CODE" = "$_CODE" ]; then
    return 0
  else
    if [ "$_COUNTER" -gt 3 ]; then
      echo "Sorry incorrect code entered, you cannot change the settings.."
      return 1
    fi
  fi
done
}

check_drive()
{
# make sure we can rewind the tape
mt -f /dev/$_DEVICE rewind > /dev/null 2>&1
if [ $? -ne 0 ]; then
  return 1
else
  return 0
fi
```

```
}
#========== main==============

# can we source the file
check_source
header
# display the loaded in variables
show_settings
# ask user if he/she wants to change any settings
if continue_prompt "Do you wish To Change Some Of The System Defaults" "Y";
then
  # yes then enter code name
  if get_code; then
    # change some settings
    change_settings
  fi
fi

#------- got settings.. now do backup
if check_drive; then
  echo "tape OK...."
else
  echo "Cannot rewind the tape..Is it in the tape drive ???"
  echo "Check it out"
  exit 1
fi

# file system paths to backup
case $_TYPE in
Full|full)
  BACKUP_PATH="sybase syb/support etc var bin apps use/local"
  ;;
Normal|normal)
  BACKUP_PATH="etc var bin apps usr/local"
  ;;
Sybase|sybase)
  BACKUP_PATH="sybase syb/support"
  ;;
esac
# now for backup
cd /
echo "Now starting backup......"
find $BACKUP_PATH -print | cpio -ovB -O /dev/$_DEVICE >> $_LOGFILE 2>&1

# if the above cpio does not work on your system try cpio below, instead
# find $BACKUP_PATH -print | cpio -ovB > /dev/$_DEVICE >> $_LOGFILE 2>&1

# to get more information on the tape change -ovB to -ovcC66536

if [ "$_INFORM" = "yes" ]; then
  echo "Backup finished check the log file" | mail admin
fi
```

源文件backup.defaults中包含函数continue_prompt，还有所有缺省设置。下面就是该源文件。

```
$ pg backup.defaults
#!/bin/sh
# backup.defaults
# configuration default file for network backups
# edit this file at your own risk !!
# name backup.defaults
#--------------------------------
# not necessary for the environments to be in quotes..but hey!
_CODE="comet"
_LOGFILE="/appdva/backup/log.`date +%y%m%d`"
_DEVICE="rmt0"
_INFORM="yes"
_TYPE="Full"
#--------------------------------
continue_prompt()
# continue_prompt
# to call: continue_prompt "string to display" default_answer
{
_STR=$1
_DEFAULT=$2
# check we have the right params
if [ $# -lt 1 ]; then
  echo "continue_prompt: I need a string to display"
  return 1
fi
while :
do
  echo -n "$_STR [Y..N] [$_DEFAULT]:"
  read _ANS
  : ${_ANS:=$_DEFAULT}
    if [ "$_ANS" = "" ]; then
      case $_ANS in
      Y) return 0 ;;
      N) return 1 ;;
      esac
    fi
  # user has selected something
  case $_ANS in
  y|Y|Yes|YES)
    return 0
    ;;
  n|N|No|NO)
    return 1
    ;;
  *) echo "Answer either Y or N, default is $_DEFAULT"
    ;;
  esac
  echo $_ANS
done
}
```

下面是该脚本运行时的输出，缺省设置被显示在屏幕上，用户被询问是否要改变这些设置：

```
User : dave                                    Tuesday 15 of June-1999
                        NETWORK SYSTEM BACKUP
```

```
                    ========================
                    Default Settings Are...
          Tape Device To Be Used     : rmt0
          Mail Admin When Done       : yes
          Type Of Backup             : Full
          Log file of backup         : /appdva/backup/log.990615
Do you wish To Change Some Of The System Defaults [Y..N] [Y]:
```

下面是用户改变缺省值的过程。在下面的例子中，备份类型被用户改变，但是该脚本在检查了相应的磁带设备之后，发现它有点问题。在使用了最后一个状态命令之后，该脚本将会退出。

```
User : dave                                          Tuesday 15 of June-1999
                        NETWORK SYSTEM BACKUP
                        =====================
Valid Entries Are...
Tape Device: rmt0, rmt1, rmt3
Mail Admin: yes, no
Backup Type: full, normal, sybase


 Tape Device To Be Used For This Backup  [rmt0] :
 Mail Admin When Done                     [yes] :
 Backup Type                              [Full : Normal
Cannot rewind the tape..Is it in the tape drive ???
Check it out
```

## 27.3 del.lines

之所以要编写这个脚本，是因为应用程序开发者总是问我“用sed的哪个命令删除空行？”。我决定写一个小脚本给他们使用，以免他们老是打电话问我这个命令。

这个脚本只是包装了一下sed命令，但它能够使用户很方便地使用，他们非常喜欢用。

脚本一般都不长。如果你认为写一个脚本能够使某些任务自动化，能够节约时间，那么你就可以编写一个脚本。

这个脚本可以处理一个或多个文件。每个文件在用 sed删除空行之前要先核实是否存在。sed的输出被导入一个文件名中含有 $$的临时文件，最后这个临时文件又被移回到原来的文件中。

该脚本使用shift命令取得所有的文件名，用 while循环逐个处理所有的文件，直至处理完为止。

可以使用del.lines -help获得一个简短的帮助。你也可以创建一个更好的帮助。

下面是该脚本。

```
$ pg del.lines
#!/bin/sh


# del.lines
# script takes filename(s) and deletes all blank lines


TEMP_F=/tmp/del.lines.$$


usage()
{
```

```
# usage
echo "Usage :`basename $0` file [file..]"
echo "try `basename $0` -help for more info"
exit 1
}

if [ $# -eq 0 ]; then
  usage
fi

FILES=$1
while [ $# -gt 0 ]
do
  echo "..$1"
  case $1 in
  -help)  cat << MAYDAY
    Use this script to delete all blank lines from a text file(s)
    MAYDAY
    exit 0
    ;;
  *) FILE_NAME=$1

    if [ -f $1 ]; then
      sed '/^$/d' $FILE_NAME >$TEMP_F
      mv $TEMP_F $FILE_NAME

    else
      echo "`basename $0` cannot find this file : $1"
    fi
    shift
    ;;
  esac
done
```

## 27.4 access.deny

在对系统进行某些更新时，你可能不希望用户登录，这时可以使用 /etc/nologin文件，大多数系统都提供这个文件。一旦在 /etc目录中使用touch命令创建了一个名为 nologin的文件，除root以外的任何用户都将无法登录。

如果系统不支持这种方法，你一样还可以做到这点——可以自己创建这个文件，下面就是具体的做法。

可以在/etc/profile文件中加入下面的代码：

```
if [ -f /etc/nologin ]; then
  if [ $LOGNAME != "root" ]; then
    echo "Sorry $LOGNAME the system is unavailable at the moment"
    exit 1
  fi
fi
```

现在，可以通过在/etc目录下创建nologin文件来阻止除根用户以外的其他用户登录。记住，该文件要对所有用户可读。

```
touch /etc/nologin
chmod 644 /etc/nologin
```

当决定恢复用户登录时，只要删除该文件即可。

```
rm /etc/nologin
```

上述办法可以很方便地组织除根用户外的所有用户登录。如果希望临时禁止某个用户登录，可以修改/etc/passwd文件，把该用户的口令域的第一个字符变成 *。不过，这个问题比较复杂，在操作之前一定要搞清楚，否则会带来系统性的问题。

LINUX提供了一个工具，可以通过它在 login.access文件中写入用户名和用户组。该文件可以用来允许或禁止用户对系统的访问。

这里有一个上述工具的简化版本 deny.access。该脚本从/etc/profile文件中运行，它读入一个名为lockout.users的文件。该文件包含有禁止登录的用户名。如果该文件中出现了 all这个单词，那么除root以外的所有用户都将被禁止登录。

下面是lockout.users文件的一个例子，该文件可以包含注释行。

```
$ pg lockout.users
# lockout.users
# put the user names in this file, that you want
# locked out of the system.
# Remove the user names from this file, to let the users back in.
# peter is on long holiday back next month
peter
# lulu is off for two weeks, back at the end of the month
lulu
#dave
#pauline
```

下面解释该脚本的工作过程。首先，通过设置trap忽略所有的信号，这样用户就无法中断它的执行。如果文件lockout.users存在，那么脚本将会继续运行。它首先检查该文件中是否存在单词all，如果存在，就不再检查该文件中的其他用户名，并禁止除根用户以外的所有其他用户登录。不要使用注释来屏蔽单词all，因为这样它仍然有可能起作用。不过你可以注释用户名。

如果单词all被找到，那么除root外的所有用户都将无法登录。为了准确起见，在该脚本中使用了grep的精确匹配模式all\>。这时用户将会在屏幕上看到系统不可用的消息。

该脚本中的主要函数是get_users 。它读入文件lockout.users，忽略所有以#开头的注释行。它通过比较用户名来确保用户名 root没有出现在该文件中，即使出现也不会禁止 root登录，否则后果将难以想象。

当前正在登录的用户名可以从变量 LOGNAME中得到，并与变量NAMES做比较，而变量NAMES的内容来自于lockout.users文件。如果匹配，相应用户的登录进程将被终止。

我在几个拥有近 40个用户的系统上运行该脚本，它并没有影响用户登录的速度。当用户外出超过一周或者用户午餐，而我需要对系统进行更新时，我就使用该脚本临时锁住相应的帐户。

需要在/etc/profile文件中加入这样一行。我把它加在该文件的末尾，这样即使用户无法登录，也可以在此之前看见当前发给他的新消息。

```
. /apps/bin/deny.access
```

/apps/bin目录是我存放全局性脚本的地方 —— 你可能把这些脚本放在另外的目录中，不过

一定要确保所有用户都对该脚本及存放它的目录具有执行权限。

如果得到“权限不足”的错误提示，那说明该脚本或目录的权限不足。

我的lockout.users文件放在/apps/etc目录下。如果你的系统的目录结构有所不同的话，应该作出相应的调整。由于该文件在登录时被引用，可以使用 set命令看到相应的函数（不过无法看到lockout.users文件）。如果你觉得这不妥，只要在这些函数执行后使用 unset命令去掉它们即可。可以把unset命令直接放在/etc/profile文件中该命令行之后，就像这样：

```
unset getusers
```

下面就是该脚本。

```
$ pg deny.access
#!/bin/sh
# deny.access

trap "" 2 3
# CHANGE BELOW IF YOU CHANGE THE LOCATION OF LOCKOUT.USERS
LOCKOUT=/apps/etc/lockout.users
MSG="Sorry $LOGNAME, your account has been disabled, ring the
administrator"
MSG2="Sorry $LOGNAME, the system is unavailable at the moment"

check_lockout()
# check_logout
# make sure we have a file containing names to lockout
{
if [ -r $LOCKOUT ] ; then
  return 0
else
  return 1
fi
}

get_users()
# get_users
# read the file, if their LOGNAME matches a name in lockout.users'
# then kick them out!
{
while read NAMES
do
  case $NAMES in
  \#*);;   #do ignore comments
  *)
  # just in case somebody tries to lockout root..not in this script they
  # won't
    if [ "$NAMES" = "root" ]; then
      break
    fi
    if [ "$NAMES" = "$LOGNAME" ]; then
    # let use see message before kicking them out
      echo $MSG
      sleep 2
      exit 1
    else
      # no match next iteration of reading the file
```

```
      continue
    fi
    ;;
  esac
done < $LOCKOUT
}
if check_lockout; then
  if grep 'all\>' $LOCKOUT >/dev/null 2>&1
then
  # first check that 'all' is not present. If it is then
  # keep everybody out apart from root
  if [ "$LOGNAME" != "root" ]; then
    echo $MSG2
    sleep 2
    exit 2
    fi
  fi
  # do normal users, if any
  get_users
fi
```

## 27.5 logroll

我的系统中的有些日志文件增长十分迅速，每天手工检查这些日志文件的长度并倒换这些日志文件（通常是给文件名加个时间戳）是非常乏味的。于是我决定编写一个脚本来自动完成这项工作。该脚本将提交给 cron进程来运行，如果某个日志文件超过了特定的长度，那么它的内容将被倒换到另一个文件中，并清除原有文件中的内容。

你可以很容易地改编这个脚本用于清除其他的日志文件。我使用另外一个脚本来清除我的系统日志文件，它每周运行一次，截断相应的日志文件。如果我需要再回头看这些日志文件，只需在备份中寻找即可，这些日志文件的备份周期为 16周，这个周期长度应该说是足够了。

该脚本中日志文件的长度限制是由变量 BLOCK_LIMIT设定的。这一数字代表了块数目，在本例中是8块（每块大小为4K字节）。可以按照自己的需求把这一数字设得更高。所有我要检查的日志文件名都保存在变量LOGS中。

这里使用了一个 for循环来依次检查每一个日志文件，使用 du命令来获取日志文件长度。如果相应的文件长度大于 BLOCK_LIMIT变量所规定的值，那么该文件将被拷贝到一个文件名含有时间戳的文件中，并改变这个文件所属的组，原先的文件长度将被截断为 0。

该脚本由cron每周运行几次，生成了一些文件名中含有时间戳的日志文件备份，这样如果系统出现了任何问题，我还可以回到这些备份中查找。

```
$ pg logroll
#!/bin/sh
# logroll
# roll over the log files if sizes have reached the MARK
# could also be used for mail boxes ?
# limit size of log
# 4096 k
BLOCK_LIMIT=8

MYDATE=`date +%d%m`
# list of logs to check...yours will be different!
```

```
LOGS="/var/spool/audlog /var/spool/networks/netlog /etc/dns/named_log"
for LOG_FILE in $LOGS
do
  if [ -f $LOG_FILE ] ; then
    # get block size
    F_SIZE=`du -a $LOG_FILE | cut -f1`
  else
    echo "`basename $0` cannot find $LOG_FILE" >&2
    # could exit here, but I want to make sure we hit all
    # logs
    continue
  fi

  if [ "$F_SIZE" -gt "$BLOCK_LIMIT" ]; then
    # copy the log across and append a ddmm  date on it
    cp $LOG_FILE $LOG_FILE$MYDATE
    # create / zero the new log
    >$LOG_FILE
    chgrp admin $LOG_FILE$MYDATE
  fi
done
```

## 27.6 nfsdown

如果系统中包含nfs文件系统，你将发现下面的脚本非常实用。我管理着几台主机，不时地需要在工作时间重启动其中的某台机器。这种重启动过程当然是越快越好。

由于我在好几个机器上都挂接了远程目录，我不想依靠系统的重启动过程来卸载这些 nfs文件系统，宁愿自己来完成这个工作。这样还可以更快一些。

只要运行这个脚本就可以迅速卸载所有的nfs文件系统，这样就能更快的重新启动机器。

该脚本的LIST变量中含有提供nfs目录的主机名。使用 for循环逐一卸载相应的目录，用grep命令在df命令的结果中查找nfs文件系统。nfs目录的mount形式为：

```
machine: remote_directory
```

这一字符串被保存在变量NFS_MACHINE中。在umount命令中使用了该变量。

下面就是该脚本：

```
$ pg nfsdown
#!/bin/sh
# nfsdown
LIST="methalpha accounts warehouse dwaggs"
for LOOP in $LIST
do
  NFS_MACHINE=`df -k | grep $LOOP | awk '{print $1}'`
  if [ "$NFS_MACHINE" != "" ]; then
    umount $LOOP
  fi
done
```

## 27.7 小结

本章中所提供的脚本都是我最常用的。正如前面所提到的，脚本不一定很长、很复杂，但是它却不失为一种高效的方法。