

人人都能学会的 SHELL编程

目 录

1.	Shell 入门简介	2
2.	Shell 编程之变量详解	5
3.	If 条件语句学习	8
4.	使用 if 条件语句编写 MySQL备份脚本	11
5.	IF 条件综合 Shell 实战脚本编写	12
6.	循环语句 for	20
7.	循环语句 while.....	22
8.	Until 循环语句	23
9.	Case选择语句	24
10.	select 选择语句	25
11.	Shell 编程函数讲解	25
12.	Shell 数组编程	27
13.	Shell 编程之 awk sed 命令案例分析	30
14.	全备和增量备份 Linux 系统脚本编写	31
15.	Shell 编程之 IP 匹配及磁盘邮件告警 .. 错误！未定义书签。	

1. Shell 入门简介

说到 Shell 编程，我想很多从事 Linux 运维工作的朋友都不陌生，都对 Shell 有基本的了解，可能大家刚开始接触 Shell 的时候，有各种想法？

能不能不学？

高大上？

到底是浮云？还是神马？



很难啊？

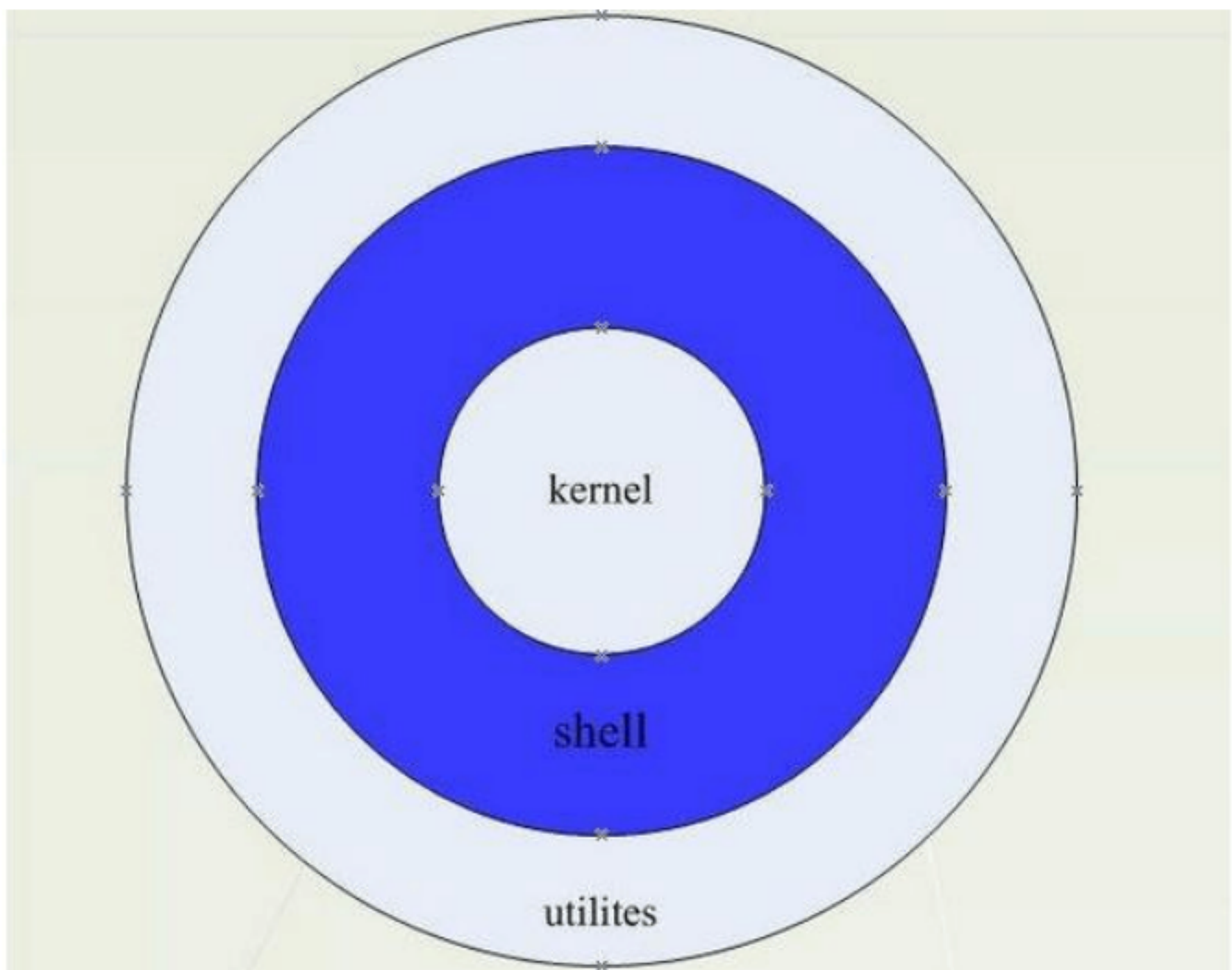
我想告诉大家的是，大家有这些想法一点都不觉得惊讶，为什么？这些都很正常，对于任何一件事情在未开始之前，肯定会有总总假设，那我今天想说的是，学完人人都会 Shell 编程后，我们会觉得一切的 Shell 都是浮云，我们每个人都能很好的使用它，驾驭它，更好的完成我们日常运维的工作。

曾经有人说过，玩 Linux 不知道 Shell ，那就是不懂 Linux ；现在细细品味确实是这样。为了让更多的人能接触 Shell 、了解 Shell ，使用 Shell ，所以今天开设了这样一个网络课程，让我们的 Shell 飞起来。

好的，其他不多说了，我们正式进入主题，什么是 Shell 呢？

Shell 是操作系统的最外层，Shell 可以合并编程语言以控制进程和文件，以及启动和控制其它程序。shell 通过提示您输入，向操作系统解释该输入，然后处理来自操作系统的任何结果输出来管理您与操作系统之间的交互。简单来说 Shell 就是一个用户跟操作系统之间的一个命令解释器。

Shell 是用户与 Linux 操作系统之间沟通的桥梁。用户可以输入命令执行，又可以利用 Shell 脚本编程去运行。



Linux Shell 种类非常多，常见的有：Bourne Shell（`/usr/bin/sh` 或 `/bin/sh`）、Bourne Again Shell（`/bin/bash`）、C Shell（`/usr/bin/csh`）、K Shell（`/usr/bin/ksh`）、Shell for Root（`/sbin/sh`）等。不同的 Shell 语言的语法有所不同，所以不能交换使用。

最常用的 shell 是 Bash，也就是 Bourne Again Shell，由于易用和免费，Bash 在日常工作中被广泛使用，也是大多数 Linux 系统默认的 Shell。接下来我们来写一个简单的 shell 脚本。（shell 脚本一般文件名以 `.sh` 结尾，同时文件第一行定义该脚本为 shell 脚本）

```
vi first_shell.sh
```

```
#!/bin/bash
```

```
#This is my First shell
```

```
echo "Hello World!"
```

这就是我们的第一个脚本，是不是很简单呢，注解如下：

```
#!/bin/bash // 表示定义该脚本是一个 shell 脚本（固定格式）。
```

```
#This is myFirst shell // 这里的 #号属于注解，没有任何的意义，  
SHELL不会解析它。
```

```
echo "Hello World!" //shell 脚本主命令，我们执行这个脚  
本讲看到：Hello World! 信息。
```

脚本编写完毕，如何来执行呢，首先执行 shell 脚本需要执行权限，
赋予执行权限：

```
chmod o+x first_shell.sh 然后 ./first_shell.sh 执行即可；也可  
以直接使用命令执行：/bin/sh first_shell.sh，显示效果一样。
```

2. Shell 编程之变量详解

Shell 编程语言是非类型的解释型语言，不像 C++/JAVA语言编程
时需要事先声明变量，SHELL给一个变量赋值，实际上就是定义了变
量，在 Linux 支持的所有 shell 中，都可以用赋值符号(=)为变量赋值。

SHELL变量可分为两类：局部变量和环境变量。局部变量只在创
建它们的 shell 脚本中使用。而环境变量则可以在创建它们的 shell
及其派生出来的任意子进程中使用。有些变量是用户创建的，其他的
则是专用 shell 变量。

例如在脚本里面定义 A=123，定义这样一个变量，前面变量名，
后面是变量的值。

引用变量可以使用 \$A ,把变量放在脚本里面会出现什么样的效果呢？

如下：

```
#!/bin/bash
```

```
#Author wugk 2014-06-10
```

```
A=123
```

```
echo " Printf variables equal is $A "
```

执行脚本： sh test.sh ，结果将会显示：

```
Printf variables equal is 123
```

简单的理解变量，相当于定义一个别名 - 名称，引用的时候加上 \$符号就可以了。

例如定义变量 name=wuguangke

执行 echo \$name 将会显示 wuguangke

SHELL常见的系统变量解析：

\$0 当前程序的名称

\$n 当前程序的第 n 个参数 ,n=1,2, , 9

\$* 当前程序的所有参数（不包括程序本身）

\$# 当前程序的参数个数（不包括程序本身）

\$? 命令或程序执行完后的状态，一般返回 0 表示执行成功。

\$UID 当前用户的 ID

\$PWD当前所在的目录

我们来测试一个常用变量的脚本： vi auto_var.sh

```
#!/bin/bash
```

```
#auto print variables
```

```
#by wugk 2014-09
```

```
echo -e '\033[32m-----\033[0m'
```

```
echo "This is $0 param !"
```

```
echo "This \"$1\" is \"$1\" param !"
```

```
echo "This \"$2\" is \"$2\" param !"
```

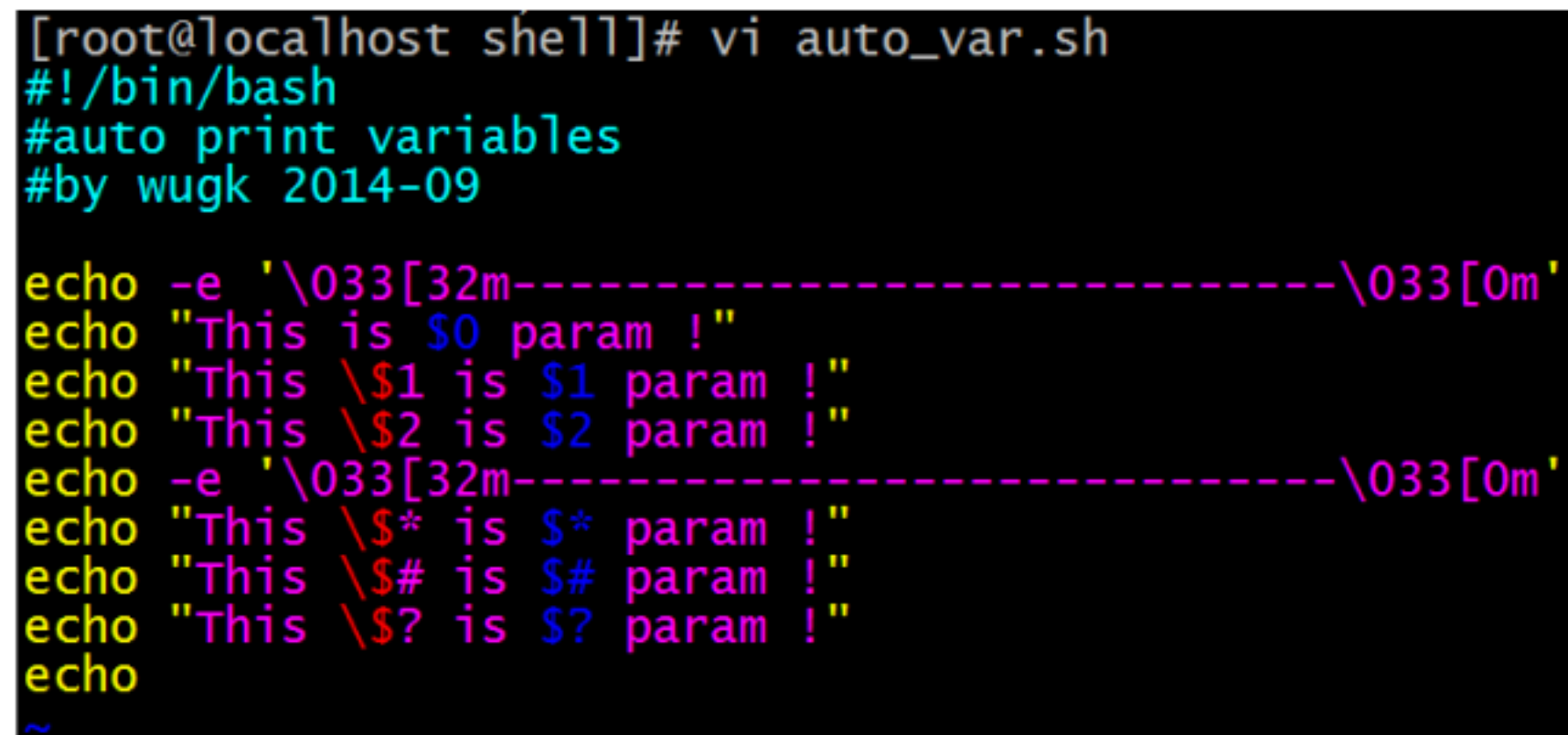
```
echo -e '\033[32m-----\033[0m'
```

```
echo "This \"$*\" is \"$*\" param !"
```

```
echo "This \"$#\" is \"$#\" param !"
```

```
echo "This \"$?\" is \"$?\" param !"
```

```
echo
```



```
[root@localhost shell]# vi auto_var.sh
#!/bin/bash
#auto print variables
#by wugk 2014-09

echo -e '\033[32m-----\033[0m'
echo "This is $0 param !"
echo "This \"$1\" is \"$1\" param !"
echo "This \"$2\" is \"$2\" param !"
echo -e '\033[32m-----\033[0m'
echo "This \"$*\" is \"$*\" param !"
echo "This \"$#\" is \"$#\" param !"
echo "This \"$?\" is \"$?\" param !"
echo
~
```

紧接着我们来编写一个简单的 echo 打印菜单：

```
#!/bin/bash
```

```
#auto install httpd
```

```
#by wugk 2014-09
```

```
echo -e '\033[32m-----\033[0m'
```

```
FILES=httpd-2.2.17.tar.bz2

URL=http://mirrors.cnnic.cn/apache/httpd/

PREFIX=/usr/local/apache2/

echo -e "\033[36mPlease Select Install Menu:\033[0m"

echo

echo "1) 官方下载 Httpd 文件包 ."

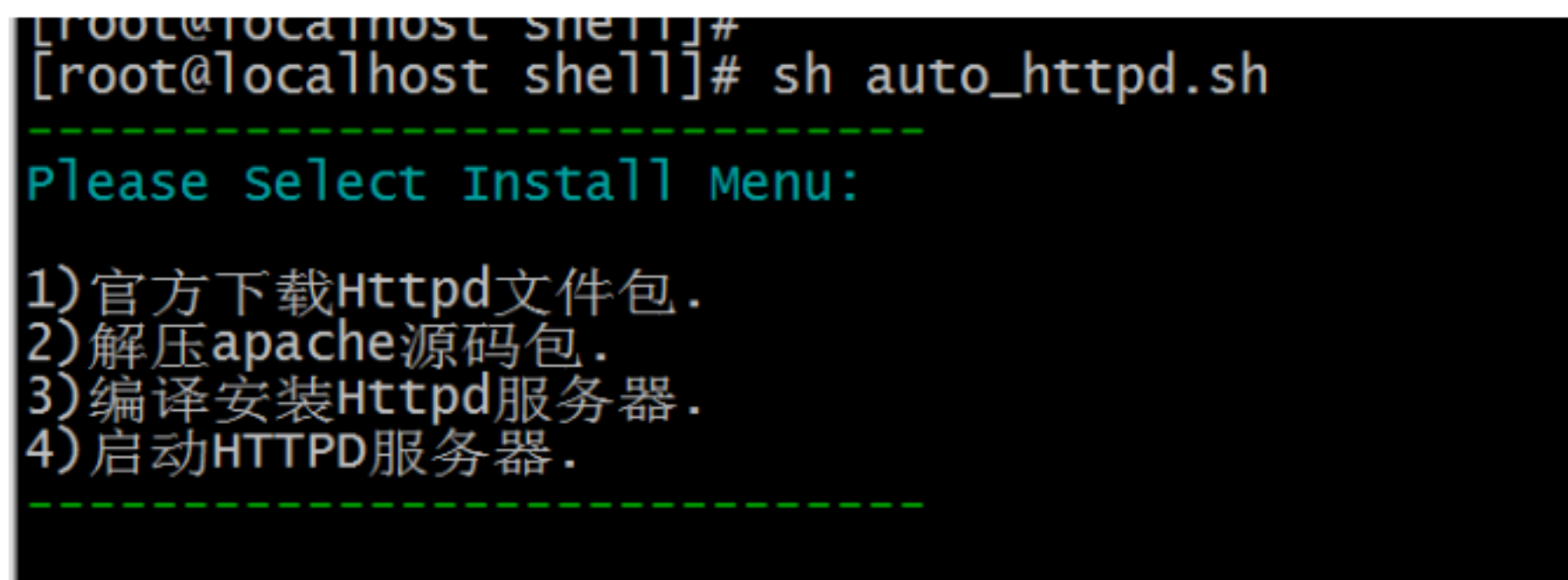
echo "2) 解压 apache 源码包 ."

echo "3) 编译安装 Httpd 服务器 ."

echo "4) 启动 HTTPD服务器 ."

echo -e '\033[32m-----\033[0m'

sleep 20
```



```
[root@localhost shell]#
[root@localhost shell]# sh auto_httpd.sh
-----
Please select Install Menu:

1)官方下载Httpd文件包.
2)解压apache源码包.
3)编译安装Httpd服务器.
4)启动HTTPD服务器.
-----
```

3. If 条件语句学习

在 Linux Shell 编程中，if 、 for 、 while 、 case 等条件流程控制语句用的非常多，把这些学好，对提升脚本的功力有非常大的帮助。

下面将逐个来讲解具体的用法：

If 条件判断语句

if (表达式) #if (Variable in Array)

语句 1

else

语句 2

fi

案例一，测试数字大小

```
#!/bin/sh
```

```
NUM=100
```

```
if (( $NUM > 4 )) ;then
```

```
    echo " this num is $NUM greater 4 !"
```

```
fi
```

案例二，测试目录是否存在，不存在则新建（注意，中括号之间必须要空格）

```
#!/bin/sh
```

```
#judge dir exist
```

```
if [ ! -d /data/20140515 ];then
```

```
    mkdir -p /data/20140515
```

```
else
```

```
    echo " This DIR is exist,Please exit , .."
```

```
fi
```

逻辑运算符解析：

-f 判断文件是否存在 eg: if [-f filename]

-d 判断目录是否存在 eg: if [-d dir]

-eq 等于 应用于：整型比较

-ne 不等于 应用于：整型比较

-lt 小于 应用于：整型比较

-gt 大于 应用于：整型比较

-le 小于或等于 应用于：整型比较

-ge 大于或等于 应用于：整型比较

-a 双方都成立（ and） 逻辑表达式 - a 逻辑表达式

-o 单方成立（ or ） 逻辑表达式 - o 逻辑表达式

-z 空字符串

案例三，多个条件测试判断

```
#!/bin/sh

scores=80

if [[ $scores -gt 85 ]]; then
    echo "very good!";
elif [[ $scores -gt 75 ]]; then
    echo "good!";
elif [[ $scores -gt 60 ]]; then
    echo "pass!";
else
    echo "no pass!"
fi
```

4. 使用 if 条件语句编写 MySQL备份脚本

a) 自动备份 Mysql 数据库脚本

```
#!/bin/sh

#auto backup mysql

#wugk 2012-12-12

#Define PATH 定义变量

BAKDIR=/data/backup/mysql/`date +%Y-%m-%d`

MYSQLDB=webapp

MYSQLPW=backup

MYSQLUSR=backup

#must use root user run scripts 必须使用 root 用户运行，$UID
为系统变量

if

[ $UID -ne 0 ];then

echo This script must use the root user !!!

sleep 2

exit 0

fi

#Define DIR and mkdir DIR 判断目录是否存在，不存在则新建

if

[ ! -d $BAKDIR ];then

mkdir -p $BAKDIR
```

```
else

    echo This is $BAKDIR exists....

fi

#Use mysqldump backup mysql 使用 mysqldump备份数据库

/usr/bin/mysqldump -u$MYSQLUSR -p$MYSQLPW -d

$MYSQLDB >$BAKDIR/webapp_db.sql

echo "The mysql backup successfully "
```

5. IF 条件综合 Shell 实战脚本编写

通过 if 语句和之前变量的学习，今天我们将把我们所学的综合在一起来讲解一个实战运维脚本，一键安装 LAMP环境的脚本：

一键源码安装 LAMP脚本，先分解脚本的各个功能：

打印菜单：

- 1) 安装 apache WEB服务器
- 2) 安装 Mysql DB 服务器
- 3) 安装 PHP 服务器
- 4) 整合 LAMP架构并启动服务

1、Apache服务器安装部署。

下载 httpd-2.2.27.tar.gz 版本，下载 URL, 解压，进入安装目录，
configure;make ;make install

2、Mysql 服务器的安装。

下载 mysql-5.5.20.tar.bz2 版本，下载 URL, 解压，进入安装目录，
configure;make ;make install

3、PHP服务器安装。

下载 php-5.3.8.tar.bz2 版本，下载 URL, 解压，进入安装目录，

configure;make ;make install

4、LAMP架构的整合和服务启动。

/usr/local/apache2/bin/apachectl start

vi htdocs/index.php

```
<?php
```

```
phpinfo();
```

```
?>
```

vi auto_lamp.sh 内容如下：

```
#!/bin/bash
```

```
#auto install LAMP
```

```
#by wugk 2014-11
```

```
#Httpd define path variable
```

```
H_FILES=httpd-2.2.27.tar.bz2
```

```
H_FILES_DIR=httpd-2.2.27
```

```
H_URL=http://mirrors.cnnic.cn/apache/httpd/
```

```
H_PREFIX=/usr/local/apache2/
```

```
#MySQL define path variable
```

```
M_FILES=mysql-5.5.20.tar.gz
```

```
M_FILES_DIR=mysql-5.5.20
```

M_URL=http://down1.chinaunix.net/distfiles/

M_PREFIX=/usr/local/mysql/

#PHP define path variable

P_FILES=php-5.3.28.tar.bz2

P_FILES_DIR=php-5.3.28

P_URL=http://mirrors.sohu.com/php/

P_PREFIX=/usr/local/php5/

echo -e "\033[32m-----\033[0m'

echo

if [-z "\$1"];then

echo -e "\033[36mPlease Select Install Menufollow:\033[0m"

echo -e "\033[32m1) 编译安装 Apache服务器 \033[1m"

echo "2) 编译安装 MySQL服务器 "

echo "3) 编译安装 PHP服务器 "

echo "4) 配置 index.php 并启动 LAMP服务 "

echo -e "\033[31mUsage: { /bin/sh \$0 1|2|3|4|help}\033[0m"

exit

```
fi
```

```
if [[ "$1" -eq "help" ]];then
```

```
    echo -e "\033[36mPlease Select Install Menu
```

```
follow:\033[0m"
```

```
    echo -e "\033[32m1)          编译安装 Apache 服务器 \033[1m"
```

```
    echo "2)          编译安装 MySQL服务器 "
```

```
    echo "3)          编译安装 PHP服务器 "
```

```
    echo "4)          配置 index.php 并启动 LAMP服务 "
```

```
    echo -e "\033[31mUsage: { /bin/sh $0
```

```
1|2|3|4|help}\033[0m"
```

```
    exit
```

```
fi
```

```
#####
```

```
#Install httpd web server
```

```
if [[ "$1" -eq "1" ]];then
```

```
    wget -c $H_URL/$H_FILES && tar -jxvf $H_FILES && cd
```

```
$H_FILES_DIR &&./configure --prefix=$H_PREFIX
```

```
if [ $? -eq 0 ];then
```

```
    make && make install
```

```
echo -e
```

```
033[0m"
```

```
echo -e "\033[32mThe $H_FILES_DIR Server Install
```

```
Success !\033[0m"
```

```
else
```

```
echo -e "\033[32mThe $H_FILES_DIR Make or Make install
```

```
ERROR,Please Check....."
```

```
exit 0
```

```
fi
```

```
fi
```

```
#Install Mysql DB server
```

```
if [[ "$1" -eq "2" ]];then
```

```
wget -c $M_URL/$M_FILES && tar -xzvf $M_FILES && cd
```

```
$M_FILES_DIR &&yum install cmake -y ;cmake .
```

```
-DCMAKE_INSTALL_PREFIX=$M_PREFIX \
```

```
-DMYSQL_UNIX_ADDR=/tmp/mysql.sock \
```

```
-DMYSQL_DATADIR=/data/mysql \
```

```
-DSYSCONFDIR=/etc \
```

```
-DMYSQL_USER=mysql \
```


-DMYSQL_TCP_PORT=3306 \
-DWITH_XTRADB_STORAGE_ENGINE=1 \
-DWITH_INNOBASE_STORAGE_ENGINE=1 \
-DWITH_PARTITION_STORAGE_ENGINE=1 \
-DWITH_BLACKHOLE_STORAGE_ENGINE=1 \
-DWITH_MYISAM_STORAGE_ENGINE=1 \
-DWITH_READLINE=1 \
-DENABLED_LOCAL_INFILE=1 \
-DWITH_EXTRA_CHARSETS=1 \
-DDEFAULT_CHARSET=utf8 \
-DDEFAULT_COLLATION=utf8_general_ci \
-DEXTRA_CHARSETS=all \
-DWITH_BIG_TABLES=1 \
-DWITH_DEBUG=0

make && make install

/bin/cp support-files/my-small.cnf /etc/my.cnf

/bin/cp support-files/mysql.server /etc/init.d/mysqld

chmod +x /etc/init.d/mysqld

chkconfig --add mysqld

chkconfig mysqld on

```
if [ $? -eq 0 ];then

    make && make install

    echo -e

033[0m"

    echo -e "\033[32mThe $M_FILES_DIR Server

Install Success !\033[0m"

else

    echo -e "\033[32mThe $M_FILES_DIRMakeor Make

install ERROR,Please Check....."

    exit 0

fi

fi
```

#Install PHP server

```
if [[ "$1" -eq "3" ]];then
```

```
    wget -c $P_URL/$P_FILES && tar -jxvf $P_FILES && cd

    $P_FILES_DIR &&./configure --prefix=$P_PREFIX

    --with-config-file-path=$P_PREFIX/etc

    --with-mysql=$M_PREFIX --with-apxs2=$H_PREFIX/bin/apxs
```

```
if [ $? -eq 0 ];then
```

```

        make ZEND_EXTRA_LIBS='-liconv' && make

install

        echo -e

033[0m"

        echo -e "\033[32mThe $P_FILES_DIR Server

Install Success !\033[0m"

    else

        echo          -e "\033[32mThe $P_FILES_DIRMakeor Make

install ERROR,Please Check....."

        exit 0

    fi

fi

#####

if [[ "$1" -eq "4" ]];then

    sed -i '/DirectoryIndex/s/index.html/index.php

index.html/g' $H_PREFIX/conf/httpd.conf

    $H_PREFIX/bin/apachectl restart

    echo "AddType

application/x-httpd-php .php" >>$H_PREFIX/conf/httpd.conf

```

```
IP=`ifconfig eth1|grep "Bcast"|awk '{print $2}'|cut -d:
-f2`
```

```
echo "You can access http://$IP/"
```

```
cat >$H_PREFIX/htdocs/index.php <<EOF
```

```
<?php
```

```
phpinfo();
```

```
?>
```

```
EOF
```

```
Fi
```

6. 循环语句 for

For 变量 in 字符串

```
do
```

```
语句 1
```

```
done
```

案例一，打印 seq 数字循环

```
#!/bin/sh
```

```
for i in `seq 15`
```

```
do
```

```
    echo " NUM is $i "
```

```
done
```

案例二，求和 1-100 的值

```
#!/bin/bash
```

```
#auto sum 1 100
```

```
j=0
```

```
for ((i=1;i<=100;i++))
```

```
do
```

```
    j=`expr $i + $j`
```

```
done
```

```
echo $j
```

案例三，找到相关 log，然后批量打包

```
#!/bin/sh
```

```
for i in `find /var/log -name " *.log "`
```

```
do
```

```
    tar -czf 2014log.tgz $i
```

```
done
```

案例四，远程主机批量传输文件：

```
#!/bin/bash
```

```
#auto scp files for client
```

```
#by authors wugk 2014
```

```
for i in `seq 100 200`
```

```
do
```

```
    scp -r /tmp/test.txt
```

```
root@192.168.1.$i:/data/webapps/www
```

Done

案例五，远程主机批量执行命令：

```
#!/bin/bash
```

```
#auto scp files for client
```

```
#by authors wugk 2014
```

```
for i in `seq 100 200`
```

```
do
```

```
    ssh -l root 192.168.1.$i          'ls /tmp'
```

```
done
```

7. 循环语句 while

```
while 条件语句
```

```
do
```

```
    语句 1
```

```
done
```

案例一， while 条件判断数字

```
#!/bin/sh
```

```
i=1;
```

```
while [[ $i -lt 10 ]];do
```

```
    echo $i;

    ((i++));

done;
```

案例二，扩展讲解 linux read 命令

```
read -p ' Please Input number: '
```

案例三， while 逐行读取某个文件

```
#!/bin/sh

while read line
do

    echo $line;

done < /etc/hosts
```

综合脚本编写：

8. Until 循环语句

```
until 条件

do

    action

done
```

直到满足条件，才退出。否则执行 action 。

案例一，条件判断数字

```
#!/bin/sh

a=10;
```

```
until [[ $a -lt 0 ]];do
```

```
echo $a;
```

```
((a--));
```

```
done;
```

9. Case选择语句

```
case $arg in
```

```
    pattern1)
```

```
        语句 1
```

```
;;
```

```
    pattern2)
```

```
        语句 2
```

```
;;
```

```
    *)
```

```
        语句 3
```

```
;;
```

```
esac
```

案例一，创建选择参数脚本

```
#!/bin/sh
```

```
case $1 in
```

```
    monitor_log)
```

```
        monitor_log
```



```

;;

archive_log)

archive_log

;;

*      )

echo "Usage:{$0 monitor_log | archive_log |help }"

;;

esac

```

10. select 选择语句

Select 一般用于选择菜单的创建，可以配合 PS3来做菜单的打印输出信息。

```
#!/bin/sh
```

```
PS3="What you like most of the open source system?"
```

```
select i in CentOS RedHat Ubuntu
```

```
do
```

```
    echo "Your Select System: "$i
```

```
Done
```

11. Shell 编程函数讲解

shell 允许将一组命令集或语句形成一个可用块，这些块称为

shell 函数，定义函数的格式：

```
function name (){  
  
    command1  
  
    .....  
  
}
```

name

编写一个 Apache安装函数：

```
#!/bin/bash  
  
#auto install LAMP  
  
#by wugk 2014-11  
  
#Httpd define path variable  
  
H_FILES=httpd-2.2.27.tar.bz2  
  
H_FILES_DIR=httpd-2.2.27  
  
H_URL=http://mirrors.cnnic.cn/apache/httpd/  
  
H_PREFIX=/usr/local/apache2/  
  
function Apache_install    ( )  
  
{  
  
    #Install httpd web server  
  
    if [[ "$1" -eq "1" ]];then  
  
        wget -c $H_URL/$H_FILES && tar -jxvf $H_FILES && cd  
$H_FILES_DIR &&./configure --prefix=$H_PREFIX  
  
        if [ $? -eq 0 ];then
```

```

        make && make install

        echo -e

033[0m"

        echo -e "\033[32mThe $H_FILES_DIR Server Install

Success !\033[0m"

    else

        echo -e "\033[32mThe $H_FILES_DIR Make or Make

install ERROR,Please Check....."

        exit 0

    fi

fi

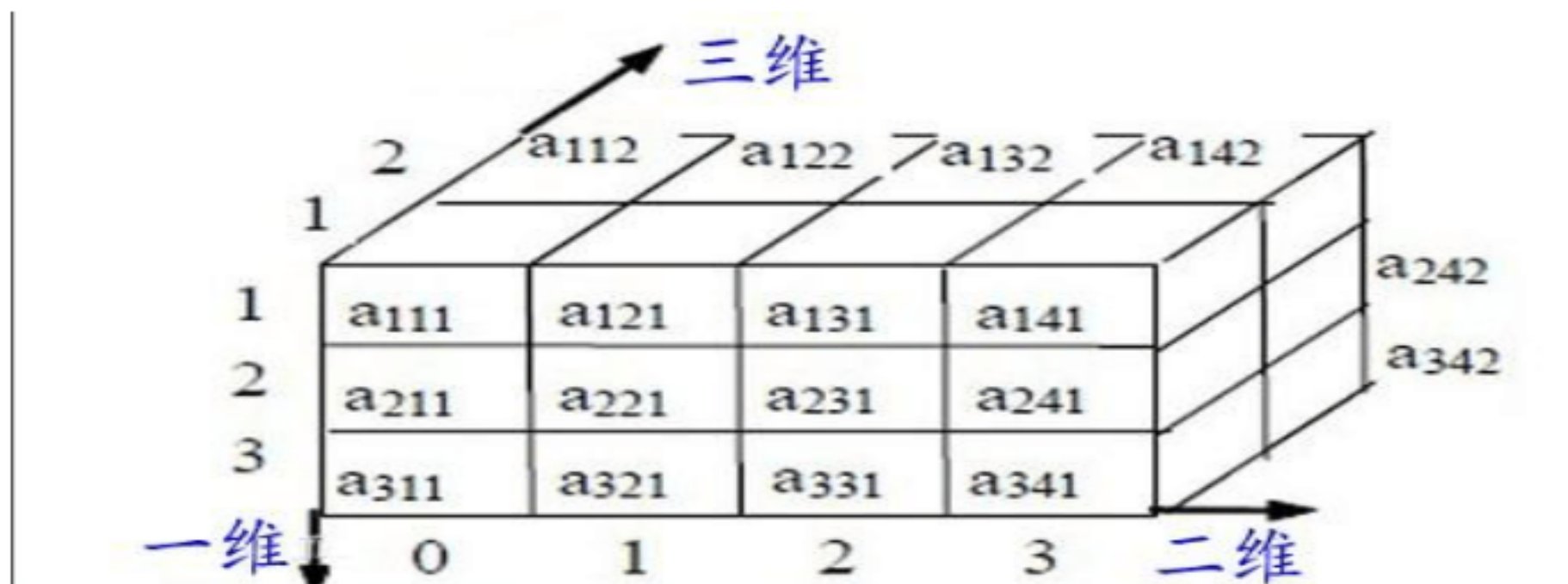
}

```

Apache_install 直接调用函数就会去运行函数里面定义的代码了。

12. Shell 数组编程

数组，就是相同数据类型的元素按一定顺序排列的集合，就是把有限个类型相同的变量用一个名字命名，然后用编号区分他们的变量的集合，这个名字成为数组名，编号成为下标。



今天这里我们来探讨一维数组的定义、统计、引用和删除等操作。首先来定义一个一维数组：

`A=(test1 test2 test3)`，定义数组一般以括号的方式来定义，数组的值可以随机定义。如何来引用呢？

`echo ${A[0]}`，代表引用第一个数组变量，结果会显示 `test1`，数组引用从 0 开始，代表第一个数组，依次类推。

`echo ${A[1]}`，代表引用第二个数组变量，结果会显示 `test2`，数组引用也是从 0 开始计算的。

如何显示该数组所有参数呢？`echo ${A[@]}` 将显示所有参数 `test1 test2 test3`。

如何显示该数组参数个数呢？`echo ${#A[@]}` 将显示该数组的参数个数 3。

如果替换某个数组呢？例如替换第二个 `test2` 数组为 `test5`：`echo ${A[@]/test2/test5}`

如何删除一个数组呢？例如删除 `test3` 数组命令为：`unset A[2];echo ${A[@]}` 查看效果。

那输入如何在编程来使用呢？请看下面例子：

```
#!/bin/sh
```

```
#Auto Make KVM Virtualization
```

```
#Auto config bond scripts
```

```
eth_bond()
```

```
{
```

```
NETWORK=(
```

```
    HWADDR=`ifconfig eth0 |egrep "HWaddr|Bcast" |tr "\n" " "|awk
```

```
'{print $5,$7,$NF}'|sed -e 's/addr://g' -e 's/Mask://g'|awk
```

```
'{print $1}'`
```

```
    IPADDR=`ifconfig eth0 |egrep "HWaddr|Bcast" |tr "\n" " "|awk
```

```
'{print $5,$7,$NF}'|sed -e 's/addr://g' -e 's/Mask://g'|awk
```

```
'{print $2}'`
```

```
    NETMASK=`ifconfig eth0 |egrep "HWaddr|Bcast" |tr "\n" "
```

```
"|awk '{print $5,$7,$NF}'|sed -e 's/addr://g' -e
```

```
's/Mask://g'|awk '{print $3}'`
```

```
    GATEWAY=`route -n|grep "UG"|awk '{print $2}'`
```

```
)
```

```
cat >ifcfg-bond0<<EOF
```

```
DEVICE=bond0
```

```
BOOTPROTO=static
```

```
${NETWORK[1]}
```

```
${NETWORK[2]}
```

```
${NETWORK[3]}
```

```
ONBOOT=yes
```

```
TYPE=Ethernet
```

```
NM_CONTROLLED=no
```

```
EOF
```

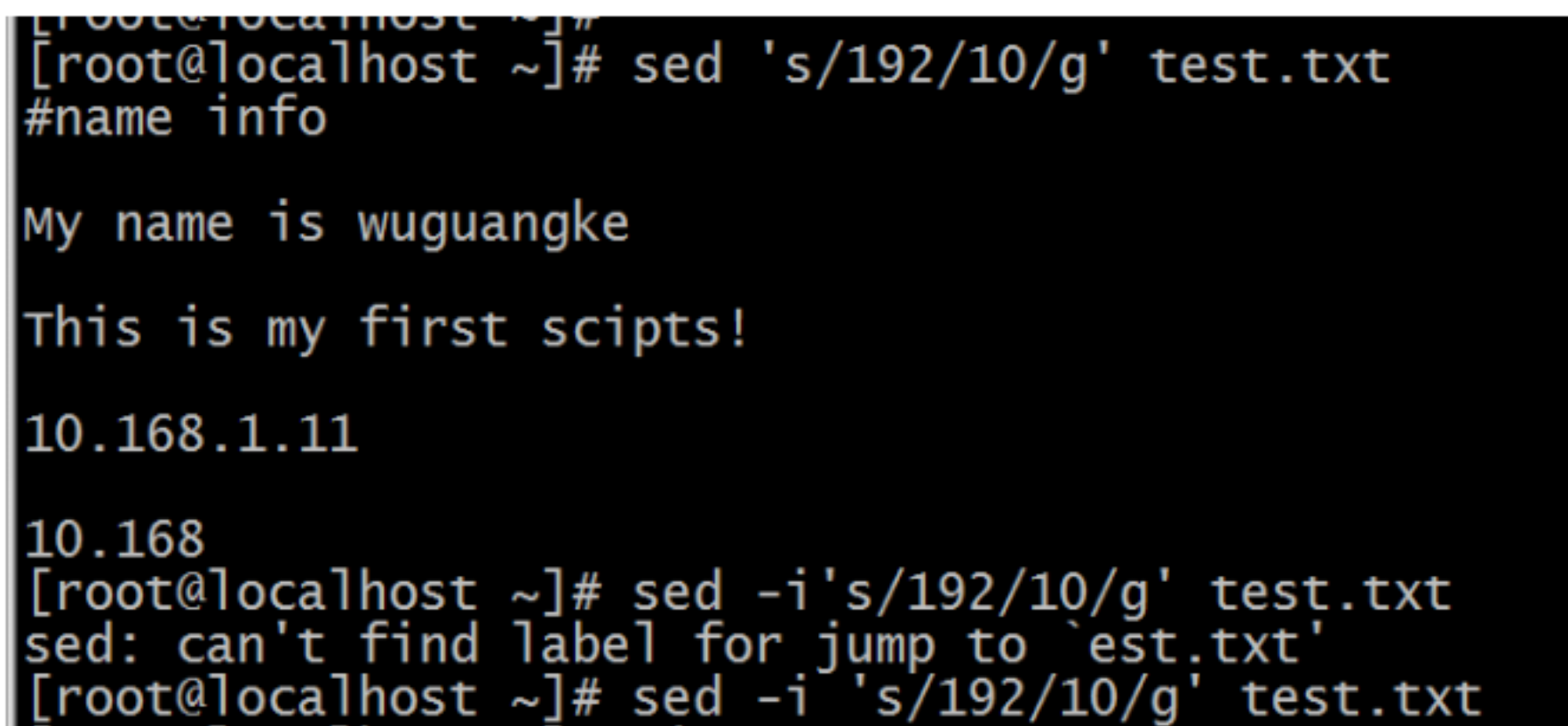
如上脚本为定义三个数组变量，然后分别来引用，这样让脚本可读性更强，更整洁。关于数组就简单的介绍到这里。

13. Shell 编程之 awk sed 命令案例分析

在我们日常的 Shell 编程中我们会用到很多的一些语句，有的语句，如果用好了，可以让我们的脚本更上一层楼，让我们的功能更容易满足企业的需求。

Sed命令：

sed 's/192/10/g' test.txt 把文件里面的 192 字符替换为 10



```
[root@localhost ~]# sed 's/192/10/g' test.txt
#name info

My name is wuguangke

This is my first scripts!

10.168.1.11

10.168
[root@localhost ~]# sed -i 's/192/10/g' test.txt
sed: can't find label for jump to `est.txt'
[root@localhost ~]# sed -i 's/192/10/g' test.txt
```

在文件开头或者结尾添加字符：

```
[root@localhost ~]#  
[root@localhost ~]# sed '/wuguangke/a #####' test.txt  
#name info  
My name is wuguangke  
#####  
This is my first scripts!  
10.168.1.11  
10.168  
[root@localhost ~]# sed '/wuguangke/i #####' test.txt  
#name info  
#####  
My name is wuguangke  
This is my first scripts!
```

Awk命令：

Find 命令：

Grep 正则：

14. 全备和增量备份 Linux 系统脚本

在 Linux 中，我们经常需要备份系统重要文件，例如 /etc 、/boot 分区、重要网站数据等等，如果每天都完整备份的话，会占用很大的空间，那我们该如何来备份呢？

这里采用如下方法来备份：

每周日进行完整备份，其余每天为增量备份。

那使用什么命令来备份呢，我们使用 tar 命令：

全备份：tar -g /tmp/snapshot -czvf

/tmp/2014_full_system_data.tar.gz /data/sh/

增量备：tar -g /tmp/snapshot -czvf

/tmp/2014_add01_system_data.tar.gz /data/sh/

```

[root@localhost tmp]# ll /data/sh
total 12
-rw-r--r-x 1 root root 1281 Nov 24 23:00 auto_device.sql.sh
-rw-r--r-- 1 root root 1920 Nov 24 23:46 auto_system.sh
drwxr-xr-x 2 root root 4096 Nov 23 10:15 wugk
[root@localhost tmp]# pwd
/tmp
[root@localhost tmp]#
[root@localhost tmp]# tar -g /tmp/snapshot -czvf /tmp/2014_full_system_data.tar.gz /data/sh/
tar: /data/sh: Directory is new
tar: /data/sh/wugk: Directory is new
tar: Removing leading `/' from member names
/data/sh/
/data/sh/wugk/
/data/sh/auto_device.sql.sh
/data/sh/auto_system.sh
[root@localhost tmp]# echo ok >/data/sh/test.txt
[root@localhost tmp]#
[root@localhost tmp]# tar -g /tmp/snapshot -czvf /tmp/2014_add01_system_data.tar.gz /data/sh/
tar: Removing leading `/' from member names
/data/sh/
/data/sh/wugk/
/data/sh/test.txt
[root@localhost tmp]#

```

#!/bin/sh

#Automatic Backup Linux System Files

#Author wugk 2013-11-22

#Define Variable

SOURCE_DIR=(

 \$*

)

TARGET_DIR=/data/backup/

YEAR=`date +%Y`

MONTH=`date +%m`

DAY=`date +%d`

WEEK=`date +%u`

A_NAME=`date +%H%M`

FILES=system_backup.tgz

CODE=\$?

if


```
[ -z "$*" ];then

    echo -e "\033[32mUsage:\nPlease  Enter  Your Backup Files  or
Directories\n-----\n

\nUsage: { $0 /boot /etc }\033[0m"

    exit
fi

#Determine Whether the Target Directory Exists

if

    [ ! -d $TARGET_DIR/$YEAR/$MONTH/$DAY ];then

        mkdir -p $TARGET_DIR/$YEAR/$MONTH/$DAY

        echo -e "\033[32mThe $TARGET_DIR Created

Successfully !\033[0m"

    fi

#EXEC Full_Backup Function Command

Full_Backup()

{

    if

        [ "$WEEK" -eq "7" ];then

            rm -rf $TARGET_DIR/snapshot

            cd $TARGET_DIR/$YEAR/$MONTH/$DAY ;tar -g

$TARGET_DIR/snapshot -czvf $FILES ${SOURCE_DIR[@]}
```

```

[ "$CODE" == "0" ]&&echo -e
"-----\n\033[32mThes
e Full_Backup System Files Backup Successfully !\033[0m"
fi
}
#Perform incremental BACKUP Function Command
Add_Backup()
{
    if
        [ $WEEK -ne "7" ];then
            cd $TARGET_DIR/$YEAR/$MONTH/$DAY ;tar -g
$TARGET_DIR/snapshot -czvf $A_NAME$FILES ${SOURCE_DIR[@]}
            [ "$CODE" == "0" ]&&echo -e
"-----\n\033[32mThese
Add_Backup System Files
$TARGET_DIR/$YEAR/$MONTH/$DAY/${YEAR}_${A_NAME}$FILES Backup
Successfully !\033[0m"
        fi
    }
    sleep 3
Full_Backup;Add_Backup

```

15. Shell 编程之系统硬件信息数据库收集

首先我们创建数据库表，格式如下：

```
CREATE TABLE `audit_audit_system` (  
    `id` int(11) NOT NULL AUTO_INCREMENT,  
    `ip_info` varchar(50) NOT NULL,  
    `serv_info` varchar(50) NOT NULL,  
    `cpu_info` varchar(50) NOT NULL,  
    `disk_info` varchar(50) NOT NULL,  
    `mem_info` varchar(50) NOT NULL,  
    `load_info` varchar(50) NOT NULL,  
    `mark_info` varchar(50) NOT NULL,  
    PRIMARY KEY (`id`),  
    UNIQUE KEY `ip_info` (`ip_info`),  
    UNIQUE KEY `ip_info_2` (`ip_info`)  
);
```

然后编写脚本如下：

```
#!/bin/sh  
  
#auto get system info  
  
#author wugk 2014-08-29  
  
echo -e "\033[34m \033[1m"  
  
cat <<EOF
```

+++++

+++++Welcome to use system Collect+++++

+++++

EOF

ip_info=`ifconfig |grep "Bcast"|tail -1 |awk '{print \$2}'|cut -d: -f 2`

cpu_info1=`cat /proc/cpuinfo |grep 'model name'|tail -1 |awk -F: '{print \$2}'|sed 's/^ //g'|awk '{print \$1,\$3,\$4,\$NF}'`

cpu_info2=`cat /proc/cpuinfo |grep "physical id"|sort |uniq -c|wc -l`

serv_info=`hostname |tail -1`

disk_info=`fdisk -l|grep "Disk"|grep -v "identifier"|awk '{print \$2,\$3,\$4}'|sed 's/,//g`

mem_info=`free -m |grep "Mem"|awk '{print "Total",\$1,\$2"M"}'`

load_info=`uptime |awk '{print "Current Load: "\$(NF-2)}'|sed 's^,/^g`

mark_info='BeiJing_IDC'

echo -e "\033[32m-----\033[1m"

echo IPADDR:\${ip_info}

echo HOSTNAME:\${serv_info}

echo CPU_INFO:\${cpu_info1} X\${cpu_info2}

echo DISK_INFO:\${disk_info}

echo MEM_INFO:\${mem_info}

echo LOAD_INFO:\${load_info}

```
echo -e "\033[32m-----\033[0m"
```

```
echo -e -n "\033[36mYou want to write the data to the databases?
```

```
\033[1m" ;read ensure
```

```
if [ "$ensure" == "yes" -o "$ensure" == "y" -o "$ensure" ==  
"Y" ];then
```

```
    echo "-----"
```

```
    echo -e '\033[31mmysql -uaudit -p123456 -D audit -e "'
```

```
"insert into audit_audit_system
```

```
values(','${ip_info}','$serv_info','${cpu_info1}
```

```
X${cpu_info2}','$disk_info','$mem_info','$load_info','$mark_info')"'
```

```
\033[0m '
```

```
else
```

```
    echo "wait exit....."
```

```
    exit
```

```
fi
```

读取数据库信息：

```
mysql -uroot -p123 -e 'use wugk1 ;select * from audit_audit_system;|sed
```

```
's/-//g'|grep -v "id"
```

这样，我们可以把数据库的内容在 shell 脚本里面调用出来。

16. Shell 编程之磁盘报警高级脚本

Mail -s ‘邮件主题 ’ -c ‘抄送地址 ’ -b ‘密送地址 ’ -f ‘发送人邮件地址 ’ -F ‘发件人姓名 ’ < ‘要发送的邮件内容’

17. Shell 编程之服务监控检查脚本

18. Shell 编程之实战 WEB界面展示一

19. Shell 编程之实战 WEB界面展示二

20. Shell 编程之学习心得分享及拓展