**第二十二章 跳出循环-shift参数左移-函数的使用**

**本节所讲内容：**

**22.1 跳出循环**

**22.2 Shift参数左移指令**

**22.3 函数的使用**

**22.4 实战-自动备份mysql数据库和nginx服务启动脚本**

# 22.1 跳出循环

**在我们使用循环语句进行循环的过程中，有时候需要在未达到循环结束条件时强制跳出循环，那么Shell给我们提供了两个命令来实现该功能：break和continue**

## 22.1.1 break和continue

**Break：跳出整个循环**

**Continue：跳过本次循环，进行下次循环**

**break概述：跳出当前整个循环或结束当前循环，在for、while等循环语句中，用于跳出当前所在的循环体，执行循环体之后的语句，后面如果什么也不加，表示跳出当前循环等价于break 1，也可以在后面加数字,假设break 3表示跳出第三层循环**

**continue概述：忽略本次循环剩余的代码，直接进行下一次循环；在for、while等循环语句中，用于跳出当前所在的循环体，执行循环体之后的语句，如果后面加的数字是1，表示忽略本次条件循环，如果是2的话，忽略下来2次条件的循环**

**互动：先演示一下效果，让大家思考一下，这个脚本的思路，然后我再带着你去写脚本。**

**例1：写一个shell菜单，当按数字键4时退出，否则一直循环显示**

**[root@xuegod63 ~]# vim break-continue.sh**

**#! /bin/sh**

**while true**

**do**

**echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"**

**echo "Please select your operation:"**

**echo " 1 Copy"**

**echo " 2 Delete"**

**echo " 3 Backup"**

**echo " 4 Quit"**

**echo "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"**

**read op**

**case $op in**

**1)**

**continue #这里加了continue后，后面的echo命令就不执行了**

**echo "your selection is Copy"**

**;;**

**2)**

**echo "your selection is Delete"**

**;;**

**3)**

**echo "your selection is Backup"**

**;;**

**4)**

**echo "Exit ..."**

**break #跳出循环体**

**;;**

**\*)**

**echo "invalide selection,please try again"**

**esac**

**done**

**例2：使用交互式方法批量添加用户**

**[root@xuegod63 ~]# cat adduser.sh**

**#!/bin/bash**

**while :**

**do**

**read -p "Please enter prefix & password & num:" pre pass num**

**printf "user information:**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**user prefix: $pre**

**user password: $pass**

**user number: $num**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**"**

**read -p "Are you sure?[y/n] " action**

**if [ "$action" == "y" ];then**

**break**

**fi**

**done**

**for i in $(seq $num) # 从i =1开始，取到 $num 。 seq 表示 1-$num**

**#$(seq $num) 等于 ` seq $num ` ； $（ 命令 ） ； ${ 变量 } ； [ 表达式/条件 ]**

**do**

**user=${pre}${i}**

**id $user &> /dev/null**

**if [ $? -ne 0 ];then**

**useradd $user**

**echo "$pass"|passwd --stdin $user &> /dev/null**

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

**echo -e "\033[31m$user\033[0m creat" #以红色来显示用户名**

**fi**

**else**

**echo "user $user exist"**

**fi**

**done**

**扩展： seq命令： seq命令用于产生从某个数到另外一个数之间的所有整数。**

**[root@xuegod63 ~]# seq 5 #输出 1-5的数字**

**1**

**2**

**3**

**4**

**5**

# 22.2 Shift参数左移指令

**shift命令用于对参数的移动(左移)，通常用于在不知道传入参数个数的情况下依次遍历每个参数然后进行相应处理（常见于Linux中各种程序的启动脚本）**

**在扫描处理脚本程序的参数时，经常要用到的shift命令，如果你的脚本需要10个或10个以上的参数，你就需要用shift命令来访问第10个及其后面的参数**

**作用：每执行一次，参数序列顺次左移一个位置，$#的值减1，用于分别处理每个参数，移出去的参数，不再可用**

****

**例子：加法计算器**

**[root@xuegod63 ~]# cat shift.sh**

**#!/bin/bash**

**if [ $# -le 0 ];then**

**echo “没有足够的参数”**

**exit**

**fi**

**sum=0**

**while [ $# -gt 0 ] ; do**

**#sum=$(expr $sum + $1)**

**sum=$[$sum+$1]**

**shift**

**# shift 2 一次移动2个参数**

**done**

**echo result is $sum**

**测试：**

**[root@xuegod63 ~]# bash a shift.sh 11 2 3 4**

**result is 20**

# 22.3 函数的使用

**函数是一个脚本代码块，你可以对它进行自定义命名，并且可以在脚本中任意位置使用这个函数，要使用这个函数，只要使用这个函数名称就可以了。使用函数的好处：模块化，代码可读性强。**

## 22.3.1 函数创建语法

**方法1：**

**function name {**

**commands**

**}**

**注意：name是函数唯一的名称**

**方法2：name后面的括号表示你正在定义一个函数**

**name(){**

**commands**

**}**

**调用函数语法：**

**函数名 参数1 参数2 …**

**调用函数时，可以传递参数。在函数中用$1、$2…来引用传递的参数**

## 22.3.2 函数的使用

**例1：**

**[root@xuegod63 ~]# cat fun-1.sh**

**#!/bin/bash**

**function fun\_1 { #定义函数**

**echo "this is function"**

**}**

**fun\_1 #调用函数**

**注意：函数名的使用，如果在一个脚本中定义了重复的函数名，那么以最后一个为准**

**[root@xuegod63 ~]# cat fun-1.sh**

**#!/bin/bash**

**function fun\_1 {**

**echo "this is function"**

**}**

**function fun\_1 {**

**echo "this is 2222222"**

**}**

**fun\_1**

**[root@xuegod63 ~]# bash fun-1.sh**

**this is 2222222**

## 22.3.3 返回值

**使用return命令来退出函数并返回特定的退出码**

**例1:**

**[root@xuegod63 ~]# vim fun-1.sh**

**#!/bin/bash**

**function fun\_1 {**

**echo "this is function"**

**ls /etc/passwd**

**return 3**

**}**

**fun\_1**

**[root@xuegod63 ~]# bash fun-1.sh #查看结果**

**this is function**

**/etc/passwd**

**[root@xuegod63 ~]# echo $?**

**3**

**注：状态码的确定必需要在函数一结束就运行return返回值；状态码的取值范围（0~255）**

**互动： exit 数字 和return 数字的区别？**

**exit整个脚本就直接退出，往回数字 ； return 只是在函数最后添加一行，然后返回数字，只能让函数后面的命令不执行，无法强制退出整个脚本的。**

## 22.3.4 把函数值赋给变量使用

**例子： 函数名就相当于一个命令**

**[root@xuegod63 ~]# cat fun-3.sh**

**#!/bin/bash**

**fun1(){**

**read -p "Input a value: " va**

**echo $[$va\*5]**

**}**

**num=$(fun1)**

**echo current num is $num**

**[root@xuegod63 ~]# sh fun-3.sh**

**Input a value: 22**

**current num is 110**

## 22.3.5 函数的参数传递

**第一种：通过脚本传递参数给函数中的位置参数$1**

**[root@xuegod63 ~]# cat fun-4.sh a.txt**

**#!/bin/bash**

**fun1(){**

**rm -rf $1**

**}**

**fun1 $1**

**第二种：调用函数时直接传递参数**

**[root@xuegod63 ~]# touch /root/a.txt #创建一个测试文件**

**[root@xuegod63 ~]# cat fun-4.sh**

**#!/bin/bash**

**fun1(){**

**rm -rf $1**

**}**

**fun1 /root/a.txt**

**[root@xuegod63 ~]# bash fun-1.sh #测试**

**[root@xuegod63 ~]# ls /root/a.txt**

**ls: 无法访问/root/a.txt: 没有那个文件或目录**

**第三种：函数中多参数传递和使用方法**

**[root@xuegod63 ~]# cat fun-5.sh**

**#!/bin/bash**

**fun1(){**

**echo $[$1\*5]**

**echo $[$2\*2]**

**}**

**fun1 5 2 #直接传两个参数**

**[root@xuegod63 ~]# bash fun-1.sh #测试**

**25**

**4**

## 22.3.6 函数中变量的处理

**函数使用的变量类型有两种：**

* **局部变量**
* **全局变量**

**1、全局变量，默认情况下，你在脚本中定义的变量都是全局变量，你在函数外面定义的变量在函数内也可以使用**

**例子：**

**[root@xuegod63 ~]# cat fun-5.sh**

**#!/bin/bash**

**function fun1 {**

**num1=$[var1\*2]**

**}**

**read -p "input a num:" var1**

**fun1**

**echo the new value is: $num1**

**[root@xuegod63 ~]# bash fun-1.sh**

**input a num:2**

**the new value is: 4**

# 22.4 实战-自动备份mysql数据库脚本和nginx服务启动脚本

## 22.4.1 自动备份mysql数据库脚本

**从centos7.0开始，系统中自带的mysql数据库包，改为mariadb数据库。**

**MariaDB数据库概述：MariaDB数据库管理系统是MySQL的一个分支，主要由开源社区在维护，采用GPL授权许可 MariaDB的目的是完全兼容MySQL，包括API和命令行，使之能轻松成为MySQL的代替品。MariaDB由MySQL的创始人Michael Widenius（迈克尔·维德纽斯）主导开发，他早前曾以10亿美元的价格，将自己创建的公司MySQL AB卖给了SUN，此后，随着SUN被甲骨文收购，MySQL的所有权也落入Oracle的手中。MariaDB名称来自Michael Widenius的女儿Maria（玛丽亚）的名字。**

**甲骨文公司收购了MySQL后，有将MySQL闭源的潜在风险，因此社区采用分支的方式来避开这个风险。 过去一年中，大型互联网用户以及Linux发行商纷纷抛弃MySQL，转投MariaDB阵营。MariaDB是目前最受关注的MySQL数据库衍生版，也被视为开源数据库MySQL的替代品。**

**安装mariadb数据库：**

**[root@xuegod63 ~]# yum install -y mariadb mariadb-server -y**

**# mariadb 是mysql的客户端命令 ；mariadb mariadb-server 是mysql服务端命令**

**[root@xuegod63 ~]# rpm -qf /usr/bin/mysql**

**mariadb-5.5.56-2.el7.x86\_64**

**[root@xuegod63 ~]# systemctl start mariadb**

**登录mysql：**

**[root@xuegod63 ~]# mysqladmin -u root password "123456" #给root用户配置一个密码123456**

**[root@xuegod63 ~]# mysql -u root -p123456 #登录mysql数据库**

**MariaDB [(none)]> show databases;**

**MariaDB [(none)]> create database xuegod ; #创建xuegod数据库**

**MariaDB [(none)]> use xuegod #选择数据库**

**MariaDB [xuegod]> create table user (id int); #创建user表，只有一个id字段**

**MariaDB [xuegod]> insert into user values(1); #插入一条记录，id字段值1**

**MariaDB [xuegod]> insert into user values(2); #插入一条记录，id字段值2**

**MariaDB [xuegod]> select \* from user; #查看表中的数据**

**+------+**

**| id |**

**+------+**

**| 1 |**

**| 2 |**

**+------+**

**mysql自动化备份脚本：**

**思路：**

**1、检查一下运行环境： 目录是否存在，时间，权限，用户**

**2、运行要执行的命令：备份，导出数据。。。**

**3、把命令执行过程中的没有用的文件删除一下**

**4、弹出命令运行成功的消息**

**[root@localhost shell]# cat mysql-back-auto.sh**

**#!/bin/sh**

**#auto backup mysql**

**#Define PATH定义变量**

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

**MYSQLDB=xuegod**

**#MYSQLDB=webapp**

**MYSQLUSR=root**

**#MYSQLUSR=backup**

**MYSQLPW=123456**

**#MYSQLPW=backup #mysql数据库密码**

**#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....**

**exit**

**fi**

**#Use mysqldump backup mysql 使用mysqldump备份数据库**

**/usr/bin/mysqldump -u$MYSQLUSR -p$MYSQLPW $MYSQLDB > $BAKDIR/${MYSQLDB}\_db.sql**

**cd $BAKDIR ; tar -czf ${MYSQLDB}\_db.tar.gz \*.sql**

**#查找备份目录下以.sql结尾的文件并删除**

**find $BAKDIR -type f -name \*.sql -exec rm -rf {} \;**

**#或**

**#如果数据库备份成功，则打印成功，并删除备份目录30天以前的目录**

**[ $? -eq 0 ] && echo “This `date +%Y-%m-%d` MySQL BACKUP is SUCCESS”**

**cd /data/backup/mysql/ && find . -type d -mtime +30 |xargs rm -rf**

**echo "The mysql backup successfully "**

## 22.4.2 nginx服务启动脚本

**此nginx脚本中使用了函数功能，让脚本具有更强的可读性**

**[root@xuegod63 ~]# vim /etc/init.d/nginx**

**#!/bin/bash**

**#chkconfig: 2345 80 90**

**#description:nginx run**

**# nginx启动脚本**

**# @author Devil**

**# @version 0.0.1**

**# @date 2018-05-29**

**PATH=/data/soft/nginx**

**DESC="nginx daemon"**

**NAME=nginx**

**DAEMON=$PATH/sbin/$NAME #/data/soft/nginx/sbin/nginx**

**CONFIGFILE=$PATH/$NAME.conf**

**PIDFILE=$PATH/$NAME.pid**

**SCRIPTNAME=/etc/init.d/$NAME**

**[ -x "$DAEMON" ] || exit 0**

**do\_start()**

**{**

**$DAEMON -c $CONFIGFILE || echo -n "nginx already running"**

**}**

**do\_stop()**

**{**

**$DAEMON -s stop || echo -n "nginx not running"**

**}**

**do\_reload()**

**{**

**$DAEMON -s reload || echo -n "nginx can't reload"**

**}**

**case "$1" in**

**start)**

**echo -n "Starting $DESC: $NAME"**

**do\_start**

**echo "."**

**;;**

**stop)**

**echo -n "Stopping $DESC: $NAME"**

**do\_stop**

**echo "."**

**;;**

**reload|graceful)**

**echo -n "Reloading $DESC configuration..."**

**do\_reload**

**echo "."**

**;;**

**restart)**

**echo -n "Restarting $DESC: $NAME"**

**do\_stop**

**do\_start**

**echo "."**

**;;**

**\*)**

**echo "Usage: $SCRIPTNAME {start|stop|reload|restart}" >&2**

**exit 3**

**;;**

**esac**

**exit 0**

**总结：**

**22.1 跳出循环**

**22.2 Shift参数左移指令**

**22.3 函数的使用**

**22.4 实战-自动备份mysql数据库和nginx服务启动脚本**