Linux学习：第五天

目录

RPM

rpm 用于互联网下载包的打包及安装工具，它包含在某些 Linux 分发版中。它生成具有.RPM 扩展名的文件。RPM 是 RedHat Package Manager（I包管理工具）的缩写，类似 windows 的 setup.exe，这一文件格式名称虽然打上了 RedHat 的标志，但理念是通用的。Linux 的分发版（suse,redhat, centos 等等），可以算是公认的行业标准了。

rpm 包的简单查询指令：看看当前系统，是否安装了 firefox ： rpm -qa | grep firefox

rpm包名基本格式：

> 一个 rpm 包名：firefox-60.2.2-1.el7.centos.x86_64 名称:firefox
>
> 版本号：60.2.2-1
>
> 适用操作系统: el7.centos.x86_64
>
> 表示 centos7.x 的 64 位系统
>
> 如果是 i686、i386 表示 32 位系统，noarch 表示通用

rpm包的其他查询指令

> rpm -qa :查询所安装的所有 rpm 软件包
>
> rpm -qa | more
>
> rpm -qa | grep X [rpm -qa | grep firefox]
>
> rpm -q 软件包名 :查询软件包是否安装。案例：rpm -q firefox
>
> rpm -qi 软件包名 ： 查询软件包信息。案例: rpm -qi firefox
>
> rpm -ql 软件包名 :查询软件包中的文件。比如： rpm -ql firefox
>
> rpm -qf 文件全路径名 查询文件所属的软件包比如：rpm -qf /etc/passwd

卸载rpm包：rpm -e RPM 包的名称 //erase

细节讨论：

LEVI_104

> 如果其它软件包依赖于您要卸载的软件包，卸载时则会产生错误信息。
>
> 如： $ rpm -e foo
>
> removing these packages would break dependencies:foo is needed by bar-1.0-1
>
> 如果我们就是要删除 foo 这个 rpm 包，可以增加参数 --nodeps ,就可以强制删除，但是一般不推荐这样做，因为依赖于该软件包的程序可能无法
>
> 如：$ rpm -e --nodeps foo

安装rpm包：rpm -ivh RPM 包全路径名称。参数说明：i=install 安装v=verbose 提示h=hash 进度条

YUM

Yum 是一个 Shell 前端软件包管理器。基于 RPM 包管理，能够从指定的服务器自动下载 RPM 包并且安装，可以自动处理依赖性关系，并且一次安
的软件包

查询 yum 服务器是否有需要安装的软件：yum list|grep xx 软件列表

安装指定的yum 包：yum install xxx 下载安装

Shell编程的简单学习

Shell 是一个命令行解释器，它为用户提供了一个向 Linux 内核发送请求以便运行程序的界面系统级程序，用户可以用 Shell 来启动、挂起、停止甚
些程序

脚本格式要求：脚本以#!/bin/bash 开头；脚本要有可执行权限

两种执行方式：（1）给予脚本执行权限+X，然后执行（./hello.sh）。（2）直接：sh hello.sh

Shell变量介绍

1. Linux Shell 中的变量分为，系统变量和用户自定义变量。
2. 系统变量：$HOME、$PWD、$SHELL、$USER 等等，比如： echo $HOME 等等..
3. 显示当前 shell 中所有变量：set

- shell 变量的定义
  - 声明静态变量：readonly 变量，注意：不能 unset
  - 撤销变量：unset 变量
  - 定义变量：变量名=值

例子

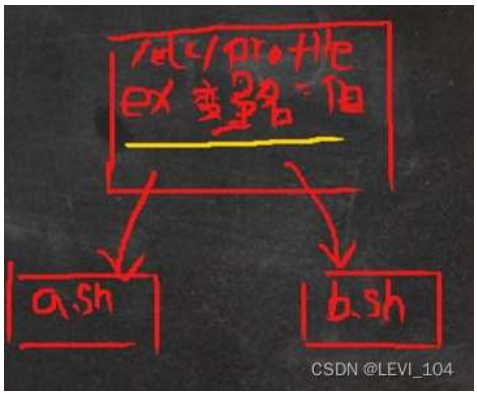
```
#!/bin/bash

#案例 1：定义变量 A A=100

#输出变量需要加上$ echo A=$A

echo "A=$A"

#案例 2：撤销变量 A unset A

echo "A=$A"

#案例 3：声明静态的变量 B=2，不能 unset readonly B=2

echo "B=$B" #unset B

#将指令返回的结果赋给变量

:<<!

C=`date` D=$(date) echo "C=$C" echo "D=$D"

!

#使用环境变量 TOMCAT_HOME

echo "tomcat_home=$TOMCAT_HOME"
```

- **将命令的返回值赋给变量**

1. A=`date`反引号，运行里面的命令，并把结果返回给变量 A
2. A=$(date) 等价于反引号

设置环境变量

1. export 变量名=变量值 （功能描述：将 shell 变量输出为环境变量/全局变量）
2. source 配置文件 （功能描述：让修改后的配置信息立即生效）
3. echo $变量名 （功能描述：查询环境变量的值）



例子：

1. 在/etc/profile 文件中定义 TOMCAT_HOME 环境变量
2. 查看环境变量 TOMCAT_HOME 的值
3. 在另外一个 shell 程序中使用 TOMCAT_HOME

注意：在输出 TOMCAT_HOME 环境变量前，需要让其生效

source /etc/profile

位置参数变量

当我们执行一个 shell 脚本时，如果希望获取到命令行的参数信息，就可以使用到位置参数变量

比如： ./myshell.sh 100 200 , 这个就是一个执行 shell 的命令行，可以在 myshell脚本中获取到参数信息

- 基本语法：
- $n （功能描述：n 为数字，$0 代表命令本身，$1-$9 代表第一到第九个参数，十以上的参数，十以上的参数需要用大括号包含，如${10}）
- $* （功能描述：这个变量代表命令行中所有的参数，$*把所有的参数看成一个整体）
- $@（功能描述：这个变量也代表命令行中所有的参数，不过$@把每个参数区分对待）
- $#（功能描述：这个变量代表命令行中所有参数的个数）

预定义变量

就是 shell 设计者事先已经定义好的变量，可以直接在 shell 脚本中使用

基本语法：

1. $$ （功能描述：当前进程的进程号（PID））
2. $! （功能描述：后台运行的最后一个进程的进程号（PID））
3. $? （功能描述：最后一次执行的命令的返回状态。如果这个变量的值为 0，证明上一个命令正确执行；如果这个变量的值为非 0（具体是哪个自己来决定），则证明上一个命令执行不正确了。）

运算符

基本语法：

1. “$((运算式))”或“$[运算式]”或者 expr m + n //expression 表达式
2. 注意 expr 运算符间要有空格, 如果希望将 expr 的结果赋给某个变量，使用 ``
3. expr m - n
4. expr　*, /, %　　乘，除，取余

条件判断

[condition]（注意 condition 前后要有空格）。#非空返回 true，可使用$?验证（0 为 true，>1 为 false）

常用判断条件：

（1)= 字符串比较

(2)两个整数的比较

-lt 小于

-le 小于等于 little equal

-eq 等于

-gt 大于

-ge 大于等于

-ne 不等于

(3)按照文件权限进行判断

-r 有读的权限

-w 有写的权限

-x 有执行的权限

(4)按照文件类型进行判断

-f 文件存在并且是一个常规的文件

-e 文件存在

-d 文件存在并是一个目录

例子：

if语句

> if [条件判断式] then
>
> 代码
>
> fi

> if [条件判断式] then
>
> 代码
>
> elif [条件判断式] then
>
> 代码
>
> fi

case语句

case $变量名 in "值 1")

如果变量的值等于值 1，则执行程序 1

;;

"值 2")

如果变量的值等于值 2，则执行程序 2

;;

…省略其他分支…

*)

如果变量的值都不是以上的值，则执行此程序

;;

esac

```
#!/bin/bash
#案例1 ： 当命令行参数是 1 时，输出 '
case $1 in
"1")
echo "周一"
;;
"2")
echo "周二"
;;
*)
echo "other..."
;;
esac
```


for循环语句

```
#!/bin/bash
#案例1 ： 打印命令行输入的参数 [这里可以看出$* 和 $@ 的区别]
#注意 $* 是把输入的参数，当做一个整体，所以，只会输出一句
for i in "$*"
do
        echo "num is $i"
done
#使用 $@来获取输入的参数，注意，这时是分别对待，所以有几个参数，就
echo "========================================="
for j in "$@"
do
        echo "num is $j"
done
```


while循环语句

#!/bin/bash

#案例 1： 从命令行输入一个数 n，统计从 1

```
SUM=0
i=0
while [ $i -le $1 ] do
SUM=$[$SUM+$i]
#i 自增
i=$[$i+1]
done
echo "执行结果=$SUM"
```

read读取控制台输入

read(选项)(参数) 选项：

-p：指定读取值时的提示符；

-t：指定读取值时等待的时间（秒），如果没有在指定的时间内输入，就不再等待了。。

```
例子
#!/bin/bash
#案例 1：读取控制台输入一个 NUM1 值read -p "请输入一个数 NUM1=" NUM1 echo "你输入的 NUM1=$NUM1"
#案例 2：读取控制台输入一个 NUM2 值，在 10 秒内输入。
read -t 10 -p "请输入一个数 NUM2=" NUM2 echo "你输入的 NUM2=$NUM2"
```

函数

basename /home/aaa/test.txt：返回了test.txt

dirname /home/aaa/test.txt：返回了/home/aaa

自定义函数

- 基本语法

```
[ function ] funname[()]
{
Action; [return int;]
}
```

调用直接写函数名：funname　[值]

综合案例：数据库备份

出现了问题，我不会做，先插个眼，以后会了再来做

1. 每天凌晨 2:30 备份 数据库 hspedu 到 /data/backup/db
2. 备份开始和备份结束能够给出相应的提示信息
3. 备份后的文件要求以备份时间为文件名，并打包成 .tar.gz 的形式，比如：2021-03-12_230201.tar.gz
4. 在备份的同时，检查是否有 10 天前备份的数据库文件，如果有就将其删除。

以下是代码：

```
#备份目录BACKUP=/data/backup/db #当前时间

DATETIME=$(date +%Y-%m-%d_%H%M%S)

echo $DATETIME #数据库的地址HOST=localhost

#数据库用户名DB_USER=root #数据库密码

DB_PW=hspedu100 #备份的数据库名DATABASE=hspedu

#创建备份目录, 如果不存在，就创建

[ ! -d "${BACKUP}/${DATETIME}" ] && mkdir -p "${BACKUP}/${DATETIME}"


#备份数据库

mysqldump -u${DB_USER} -p${DB_PW} --host=${HOST} -q -R --databases ${DATABASE}
gzip >

${BACKUP}/${DATETIME}/$DATETIME.sql.gz


#将文件处理成 tar.gz cd ${BACKUP}

tar -zcvf $DATETIME.tar.gz ${DATETIME} #删除对应的备份目录

rm -rf ${BACKUP}/${DATETIME}


#删除 10 天前的备份文件

find ${BACKUP} -atime +10 -name "*.tar.gz" -exec rm -rf {} \; echo "备份数据库${DATABASE} 成功~"
```

我的问题（百度了也不会搞，有无大佬留言帮下忙）：

```
[root@Q1nChanCentos7 2022-07-05_132331]# /usr/sbin/mysql_db_backup.sh
2022-07-05_133202
mysqldump: [Warning] Using a password on the command line interface can be insecure.
mysqldump: Got error: 2005: Unknown MySQL server host '{HOST}' (2) when trying to con
[root@Q1nChanCentos7 2022-07-05_132331]# vim /usr/sbin/mysql_db_backup.sh
[root@Q1nChanCentos7 2022-07-05_132331]# /usr/sbin/mysql_db_backup.sh
2022-07-05_133345
mysqldump: [Warning] Using a password on the command line interface can be insecure.
mysqldump: Got error: 2005: Unknown MySQL server host '{HOST}' (2) when trying to con
[root@Q1nChanCentos7 2022-07-05_132331]# ^C
```


LEVI_104