Shell

基本概述

- shell是一个命令行解释器，它接受应用程序/用户命令，然后调用操作系统内核

Shell中的变量

常见的系统变量

```
$HOME  $PWD    $SHELL  $USER
```

自定义变量

基础语法

1. 定义变量：变量=值 **（注意等号左右不能有空格）**
2. 撤销变量：`unset 变量`
3. 声明静态变量：`readonly 变量`，注意：不能`unset`

定义变量的规则

1. 变量名称可以由字母，数字和下划线组成，但是不能以数字开头，**环境变量名建议大写**
2. **等号左右不能有空格**
3. 在`bash`中，**变量默认类型都是字符串类型**，无法直接进行数值运算
4. 变量的值**如果有空格**，需要使用**双引号或者单引号括起来**
5. 使用`export 变量名`，可以把变量提升为全局环境变量，可供其他Shell程序使用

特殊变量 $n

- `$n`（功能描述：`n`为数字，`$0`表示该脚本的名称，`$1-$9`表示第一道第九个参数，十以上的参数需要用大括号包含，如`${10}`）

特殊变量 $#

- `$#`（功能描述：获取所有输入参数的个数，常用于循环）

特殊变量 $* $@

- `$*`（功能描述：这个变量表示命令行中所有的参数，`$*`把所有的参数看成一个整体）
- `$@`（功能描述：这个变量也代表命令行中所有的参数，不过`$@`把每一个参数区分对待）

特殊变量 $?

- `$?`（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；如果这个变量的值非0，则证明上一个命令执行不正确。）

Shell 运算符

基本语法

1. `$((运算式))` 或 `$[运算式]`
2. `expr` + - ***** / % 加，减，乘，除，取余 **（注意：`expr`运算符之间要有空格）**

运算案例：

```
# 计算 (2 + 3) * 4
$ expr `expr 2 + 3` \* 4   # 用``包裹起来的可以简单看作为小括号，提升执行的优先级
>> 20
$ s = $[(2+3)*4]
$ echo $s
>> 20
```

条件判断

基础语法

`[ condition ]` **（注意 condition 前后要有空格）**

注意：条件非空即为`true`，空返回`false`

常用的条件判断

两个整数之间的比较

```
=       字符串比较
-lt     小于
-le     小于等于
-gt     大于
-ge     大于等于
-eq     等于
-ne     不等于
```

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
```

按照文件类型判断

```
-f      文件存在并且是一个常规的文件
-e      文件存在
-d      文件存在并且十一个目录
```

多条件判断

- `&&` 表示前一条命令执行成时，才执行后一条指令，`||`表示命令执行失败后，才执行下一条指令

流程控制

if

```
if [ 条件判断式 ];then
    程序
fi
或者
if [ 条件判断式子 ]
    then
        程序
fi
```

注意事项：

- [条件判断式]，**中括号和条件判断式之间必须有空格**
- `if` **后要有空格**

case

```
case $变量名 in
    "值1")
        如果变量值等于值1，则执行程序1
    ;;
    "值2")
        如果变量值等于值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
```

注意事项：

- `case` 行尾必须为单词 `in`，每一个模式匹配必须以右括号 `）` 结束
- 双分号 `;;` 表示命令序列的结束，相当于 `java` 中的 `break`
- 最后的 `*）` 表示默认模式，相当于 `java` 中的 `default`

for

```
for(( 初始值;循环控制变量;变量变换 ))
    do
        程序
    done
```

```
for 变量 in 值1 值2 值3 ...
    do
        程序
    done
```

while

```
while [ 条件判断式子 ]
    do
        程序
    done
```

read 读取控制台输入

基础语法

```
read(选项)(参数)
选项：
    -p：指定读取值是的提示符
    -t：指定读取值时等待的时间（秒）
参数：
    变量：指定读取值的变量名
```

Shell工具

cut

- `cut`的工作就是“剪”，具体的说就是在文件中负责剪切数据用的。cut命令从文件的每一行中剪切字节，字符和字段并将这些字节，字符和字段输出

基本用法

- `cut [选项参数] filename`默认的分隔符是制表符

选项参数

选项参数	功能
-f	列号，提取第几**列**
-d	分隔符，按照指定的分隔符分割**列**

sed

- `sed`是一种流编辑器，它一次性处理一行内容。处理时，把当前处理的行存储在临时缓冲区中，称为“模式空间”，接着用`sed`命令处理缓冲区中的内容，处理完成后，把缓冲区的内容送往屏幕。接触处理下一行，直到文章末尾。**文件内容并没有改变**，除非你使用重定向存储输出。

基本用法

```
sed [选项参数] 'command'  filename`
```

选项参数

选项参数	功能
-e	直接在指令列模式上进行`sed`的动作编辑

命令功能描述

命令	功能描述
a	新增，a的后面可以接字符串，在下一行出现
d	删除
s	查找并替换

例子：

```
# 在第二行加入ab cd
$ sed "2a ab cd" sed.txt
# 删除 sed.txt 文件所有包含 wo 的行
$ sed "/wo/d" sed.txt
# 将sed.txt 文件中wo替换为ni
$ sed "s/wo/ni/g" sed.txt
```

awk

- 一个强大的文本分析工具，把文件逐行读入，以空格为默认分隔符将每行切片，切开的部分再进行分析处理

基本用法

```
awk [选项参数] 'pattern1{action1}   pattern2{action2}...'   filename
```

- `pattern`：表示`awk`在数据中查找的内容，就是匹配模式
- `action`：在找到匹配的过程中所执行的一系列命令

选项参数

选项参数	功能
-F	指定输入文件的分隔符
-v	赋值一个用户定义的变量

内置变量

变量	说明
FILENAME	文件名
NR	已读的记录数
NF	浏览记录的域的个数（切割后，列的个数）

sort

- `sort`命令是在`Linux`里非常有用，它将文件进行排序，并将排序结果标准输出

基本语法

```
sort (选项)(参数)
```

选项参数

选项	说明
-n	依照数值大小排序
-r	以相反的顺序来排序
-t	设置排序时所用的分隔字符
-k	指定需要排序的列