Shell

▼ Content

The third-part libraries

📖 expect

Basic

解释器

`#!/bin/bash` or `#!/bin/sh`

各种括号⭐

代码	解释	例子
`${...}`	变量使用	omit

代码	解释	例子
`$(...)` or `` `...` ``	shell命令使用，`` `…` `` 通用性更高，`$(...)` 不是所有shell都支持	`v1=`printf "%02d" 1``
`$[...]` or `$((...))`	进行算术运算的区域，变量不必加`${...}`	`a=5;b=7;c=2` `echo $((a+b))`
`test ...` or `[ ... ]`	用来比较判断，后者的中括号前后必须要有括号，注意它们都不能直接用`>` 或`<` ，需要用`-gt` 或`-lt`	`a=3;b=4;` `if [ ${a} -lt ${b} ] ...`
`((...))` or `[[...]]`	比较判断的加强版，前者针对数学比较表达式，后者针对字符串表达式，可以直接使用`>` 或`<`	`a=3;b=4;` `if (( ${a} < ${b} )) ...`

💡 Original Shell doesn't support float number

Arguments & Variables

Arguments

Example: `./test.sh -p 9532`

Arguments	Meaning	Example
`$n`	n is a number，表示第几个参数（参数0是执行的脚本名）	`echo $0` → `./test.sh`
`$#`	参数个数，不包含文件名	`echo $#` → `2`

Variable

```
# variables declaration
v1="123"
v2=123
v3="hi"
v4=hi # can omit quotation marks

# variables usage
echo ${v4} # better usages
echo $v4
```

Flow Control

Condition Control

```
v1=5
v2=10
if [[ ${v1} < ${v2} ]]
then
    echo "Yes"
elif
then
    echo "No"
else
    echo "Else"
fi
```

Numeric test

代码	意义
-eq	等于则为真
-lt	小于则为真
-le	小于等于则为真
-ge	大于等于则为真
-gt	大于则为真
-ne	不等于则为真

String test

代码	意义
==	等于则真
`!=`	不等于
-z 字符串	字符串长度为0
-n 字符串	字符串长度不为零

```
num1=100
num2=100
if test $[num1] -eq $[num2]
then
    echo '两个数相等！'
else
    echo '两个数不相等！'
fi

if [ ${num1} -eq ${num2} ]
then
    echo '两个数相等！'
else
```

```
    echo '两个数不相等！'
fi
```

File test

File test

Aa 参数	说明
-e 文件名	如果文件存在则为真
-r 文件名	如果文件存在且可读则为真
-w 文件名	如果文件存在且可写则为真
-x 文件名	如果文件存在且可执行则为真
-s 文件名	如果文件存在且至少有一个字符则为真
-d 文件名	如果文件存在且为目录则为真
-f 文件名	如果文件存在且为普通文件则为真
-c 文件名	如果文件存在且为字符型特殊文件则为真
-b 文件名	如果文件存在且为块特殊文件则为真

Loop Control

IO

Input

echo

自带换行符

```
echo -e "hi\nhi" # 要转义需要-e
echo "to file" > test.txt # 可以重定向，重写
echo "to file" >> test.txt # 重定向，追加
```

printf

需要手动换行，可以格式化，无法重定向

```
printf "hi\nhi\n"
```

spawn id exp4 not open

Sed

`/` 是分割符

re正则表达式

元字符	功 能	示 例	示例的匹配对象
^	行首定位符	/^love/	匹配所有以 love 开头的行
$	行尾定位符	/love$/	匹配所有以 love 结尾的行
.	匹配除换行外的单个字符	/l..e/	匹配包含字符 l、后跟两个任意字符、再跟字母 e 的行
*	匹配零个或多个前导字符	/*love/	匹配在零个或多个空格紧跟着模式 love 的行
[]	匹配指定字符组内任一字符	/[Ll]ove/	匹配包含 love 和 Love 的行
[^]	匹配不在指定字符组内任一字符	/[^A-KM-Z]ove/	匹配包含 ove，但 ove 之前的那个字符不在 A 至 K 或 M 至 Z 间的行
\(..\)	保存已匹配的字符		
&	保存查找串以便在替换串中引用	s/love/**&**/	符号&代表查找串。字符串 love 将替换前后各加了两个**的引用，即 love 变成**love**
\<	词首定位符	/\<love/	匹配包含以 love 开头的单词的行
\>	词尾定位符	/love\>/	匹配包含以 love 结尾的单词的行
x\{m\}	连续 m 个 x	/o\{5\}/	分别匹配出现连续 5 个字母 o、至少 5 个连续的 o、或 5~10 个连续的 o 的行
x\{m,\}	至少 m 个 x	/o\{5,\}/	
x\{m,n\}	至少 m 个 x，但不超过 n 个 x	/o\{5,10\}/	

常用选项

选项	说明
-n	使用安静模式，在一般情况所有的 STDIN 都会输出到屏幕上，加入-n 后只打印 被 sed 特殊处理的行
-e	多重编辑，且命令顺序会影响结果
-f	指定一个 sed 脚本文件到命令行执行，
-r	Sed 使用扩展正则
-i	直接修改文档读取的内容，不在屏幕上输出

操作指令

命 令	说 明
a\	在当前行后添加一行或多行
c\	用新文本修改（替换）当前行中的文本
d	删除行
i\	在当前行之前插入文本
h	把模式空间里的内容复制到暂存缓存区
H	把模式空间里的内容追加到暂存缓存区
g	取出暂存缓冲区里的内容，将其复制到模式空间，覆盖该处原有内容
G	取出暂存缓冲区里的内容，将其复制到模式空间，追加在原有内容后面
l	列出非打印字符
p	打印行
n	读入下一输入行，并从下一条命令而不是第一条命令开始处理
q	结束或退出 sed
r	从文件中读取输入行
!	对所选行意外的所有行应用命令
s	用一个字符串替换另一个

- `g`：对于全部

Example

```
# test.txt
This is a test file.
option domain-name 123456789;
```

```
#!/bin/bash

file=test.txt

# replace
sed -i 's/option domain-name .*;$/123/g' ${file} # 将text.txt中的第三行（以及所有符合该正则的字符串）替换成123

# delete
sed -i 's/option domain-name .*;$//g' ${file} #就是替换成空字符串

# add
sed -i '/This is a test file./a 123' ${file} # 第一行下面加一行"123"
```