笔记：bash 脚本入门

编写 bash 脚本，首先在文件的第一行加入如下代码：
#! /bin/bash
比如文件 test1.sh 的完整代码：
#! /bin/bash
echo "Just a test!"

执行如下指令：
chmod +x test1.sh && ./test1.sh
即可看到效果。

一、变量相关

1、输入变量

read var

2、输出变量

echo $var

demo 代码如下：
#! /bin/bash
read var
echo $var

二、判断相关

1、逻辑判断

逻辑卷标	表示意思
1.	关于档案与目录的侦测逻辑卷标！
-f	常用！侦测『档案』是否存在 eg: if [-f filename]
-d	常用！侦测『目录』是否存在
-b	侦测是否为一个『 block 档案』
-c	侦测是否为一个『 character 档案』
-S	侦测是否为一个『 socket 标签档案』
-L	侦测是否为一个『 symbolic link 的档案』
-e	侦测『某个东西』是否存在！
2.	关于程序的逻辑卷标！
-G	侦测是否由 GID 所执行的程序所拥有
-O	侦测是否由 UID 所执行的程序所拥有
-p	侦测是否为程序间传送信息的 name pipe 或是 FIFO （老实说，这个不太懂！）

3.	关于档案的属性侦测！
-r	侦测是否为可读的属性
-w	侦测是否为可以写入的属性
-x	侦测是否为可执行的属性
-s	侦测是否为『非空白档案』
-u	侦测是否具有『 SUID 』的属性
-g	侦测是否具有『 SGID 』的属性
-k	侦测是否具有『 sticky bit 』的属性
4.	两个档案之间的判断与比较 ； 例如[test file1 -nt file2]
-nt	第一个档案比第二个档案新
-ot	第一个档案比第二个档案旧
-ef	第一个档案与第二个档案为同一个档案（ link 之类的档案）
5.	逻辑的『和(and)』『或(or)』
&&	逻辑的 AND 的意思
||	逻辑的 OR 的意思

2、运算符号

运算符号	代表意义
=	等于 应用于：整型或字符串比较 如果在[] 中，只能是字符串
!=	不等于 应用于：整型或字符串比较 如果在[] 中，只能是字符串
<	小于 应用于：整型比较 在[] 中，不能使用 表示字符串
>	大于 应用于：整型比较 在[] 中，不能使用 表示字符串
-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
-n	非空字符串

3、逻辑表达式

test 命令：

```
# test -d /etc/ && echo 'ok'
ok
```

[] 表达式：

```
# [ 1 -eq 1 ] && echo 'ok'
ok
```

[[]] 表达式：

```
# [[ 2 < 3 && 4 > 5 ]] && echo 'ok'
ok
```

4、条件判断

if then fi 的方式

结构如下：

```
if [ expression ]
then
    statments
fi
```

或者

```
if [ expression ]
then
    statments
else
    statments
fi
```

或者

```
if [ expression ]
then
    statments
else if [ expression ]
    then
```

```
        statments
    else
        statments
fi
```

或者

```
if [ expression ]
then
    statments
elif [ expression ]
    then
        statments
    else
        statments
fi
```

示例代码如下：

```
#! /bin/bash
echo "Press y to continue"
read yn
if [ "$yn" = "y" ] ; then
        echo "Script is running ..."
else
        echo "Break!"
fi
```

case ...esac 方式

结构如下：

```
case "$var" in
  condition1 )
    statments1;;
```

```
  condition2 )
    statments2;;
  ...
  * )
    default statments;;
esac
```

示例代码如下：

```
#! /bin/bash
echo "This program will print your selection!"

case $1 in
A)
        echo "your choice is A"
        ;; # the break
B)
        echo "your choice is B"
        ;;
C)
        echo "your choice is C"
        ;;
*) # the default way
        echo "usage {A|B|C}"
        exit 1
esac
```

三、循环相关

for 循环

结构如下：

```
for $var in [list]
do
    statments
```

done

示例代码如下：
```
#! /bin/bash
declare -i s
for (( i=1; i<=100; i=i+1 ))
do
        s=s+i
done
echo "The count is ==> $s"
```

while 循环

结构如下：
```
while [ condition ]
do
    statments
done
```

until 循环

结构如下：

```
until [ condition is TRUE ]
do
    statments
done
```

四、其它

1、命令行参数

类似 c 语言中 main 函数的 argv 参数，通常的调用如下:
myscript param1 param2

demo 代码如下：
```
#! /bin/bash
echo $#
echo $@
echo '$0 = '$0
echo '$1 = '$1
echo '$2 = '$2
```

解释如下：

$# 是传给脚本的参数个数
$@ 是传给脚本的所有参数的列表
$0 是脚本本身的名字
$1 是传递给该 shell 脚本的第一个参数
$2 是传递给该 shell 脚本的第二个参数

2、调试
sh [-nvx] scripts
-n :不要执行 scripts ,查询 scripts 内的语法,若有错误则予以列出!
-v :在执行 scripts 之前,先将 scripts 的内容显示在屏幕上;
-x :将有使用到的 scripts 内容显示在屏幕上,与 -v 稍微不同!