**linux shell bash 比较操作**

2012-02-24 14:48:41 [**我来说两句**](http://www.2cto.com/os/201202/120793.html#comment_iframe)

[收藏](javascript:;) [我要投稿](http://www.2cto.com/tg.html) [字体：[小](javascript:fontZoomA();) [大](javascript:fontZoomB();)]

整数比较  
-eq 等于,如:if [ "$a" -eq "$b" ]  
-ne 不等于,如:if [ "$a" -ne "$b" ]  
-gt 大于,如:if [ "$a" -gt "$b" ]  
-ge 大于等于,如:if [ "$a" -ge "$b" ]  
-lt 小于,如:if [ "$a" -lt "$b" ]  
-le 小于等于,如:if [ "$a" -le "$b" ]  
  
< 小于(需要双括号),如:(("$a" < "$b"))  
< = 小于等于(需要双括号),如:(("$a" <= "$b"))  
> 大于(需要双括号),如:(("$a" > "$b"))  
> = 大于等于(需要双括号),如:(("$a" >= "$b"))  
  
字符串比较  
  
= 等于,如:if [ "$a" = "$b" ]  
== 等于,如:if [ "$a" == "$b" ],与=等价  
  
注意: [[]]和[]中的行为在某些情况下是不同的:  
  
[[ $a == z\* ]] # 如果$a 以"z"开头(模式匹配)那么将为true  
[[ $a == "z\*" ]] # 如果$a 等于z\* (字符匹配),那么结果为true  
  
[ $a == z\* ] # File globbing 和word splitting 将会发生  
[ "$a" == "z\*" ] # 如果$a 等于z\* (字符匹配),那么结果为true  
  
File globbing 是一种关于文件的速记法,比如"\*.c"就是,再如~也是.  
但是file globbing 并不是严格的正则表达式,虽然绝大多数情况下结构比较像.  
(笔者不太懂, 假如在/tmp目录下有一个a.txt文件，则[ "a.txt" = a.\* ]会返回逻辑真,但如果/tmp下有a.txt和a.out两个文件，则这样写Shell会报错)  
  
!= 不等于,如:if [ "$a" != "$b" ]  
注意：!= 在[[]]结构中使用模式匹配.  
  
< 小于,在ASCII 字母顺序下.如:  
if [[ "$a" < "$b" ]]  
if [ "$a" \< "$b" ]  
注意:在[]结构中"<"需要被转义.  
  
> 大于,在ASCII 字母顺序下.如:  
if [[ "$a" > "$b" ]]  
if [ "$a" \> "$b" ]  
注意:在[]结构中">"需要被转义.  
  
-z 字符串为空.就是长度为0.  
-n 字符串不为空  
注意: 对于字符串或数字的比较, 加上双引号("")是没有坏处的, 而且能避免一些不必要的麻烦（请参考，但模式和正则表达式不能加双引号.  
  
  
混合比较  
-a 逻辑与  
exp1 -a exp2 如果exp1 和exp2 都为true 的话,这个表达式将返回true  
-o 逻辑或  
exp1 -o exp2 如果exp1 和exp2 中有一个为true 的话,那么这个表达式就返回true  
  
但在[[]]使用&&和|| 代替-a 和-o, -o 和-a 一般都是搭配test 命令或者[]  
  
看一些例子：  
01  
#!/bin/bash  
02  
# 数字和字符串比较  
03  
  
04  
# 这里的变量a 和b 既可以当作整型也可以当作是字符串.  
05  
# 因为Bash 变量并不是强类型的.  
06  
a=4  
07  
b=5  
08  
# a,b作为整数进行比较  
09  
if [ "$a" -ne "$b" ]  
10  
then  
11  
echo "$a is not equal to $b"  
12  
echo "(arithmetic comparison)"  
13  
fi  
14  
echo  
15  
# a,b作为字符串进行比较  
16  
if [ "$a" != "$b" ]  
17  
then  
18  
echo "$a is not equal to $b."  
19  
echo "(string comparison)"  
20  
fi  
21  
  
22  
# 在这个特定的例子中,"-ne"和"!="都可以.  
23  
  
24  
exit 0  
  
  
01  
#!/bin/bash  
02  
# 测试字符串是否为null  
03  
# $string1 没被声明和初始化  
04  
if [ -n $string1 ]  
05  
then  
06  
echo "String \"string1\" is not null."  
07  
else  
08  
echo "String \"string1\" is null."  
09  
fi  
10  
# 错误的结果.  
11  
# 显示$string1 为非空,虽然他没被初始化.  
12  
  
13  
echo  
14  
  
15  
# 给$string1加上双引号再试一下.  
16  
if [ -n "$string1" ]  
17  
then  
18  
echo "String \"string1\" is not null."  
19  
else  
20  
echo "String \"string1\" is null."  
21  
fi  
22  
# 这次输出我们预期的结果  
23  
  
24  
echo  
25  
  
26  
if [ $string1 ] # 这次$string1 变成"裸体"的了,相当于[ -n $string1 ]  
27  
then  
28  
echo "String \"string1\" is not null."  
29  
else  
30  
echo "String \"string1\" is null."  
31  
fi  
32  
  
33  
string1=initialized  
34  
if [ $string1 ] # 再来  
35  
then  
36  
echo "String \"string1\" is not null."  
37  
else  
38  
echo "String \"string1\" is null."  
39  
fi  
40  
# 这次输出我们期望的结果  
41  
  
42  
# 但最好还是给字符串加上双引号"", 因为这可以避免不必要的麻烦，也是一种好习惯