(75条消息) Bash中的字符串处理_天冷吃个橘子的博客-CSDN博客_bash 字符串处理

Bash中的字符串处理

字符串切片

```
//示例
[root@node1 ~]# DOMAIN=www.baidu.com
[root@node1 ~]# echo $DOMAIN
www.baidu.com
```

- 1
- 2
- 3
- 4

- `${var#*bai}`，自左而右，查找var变量中存储的字符串中第一次出现的由bai所指明的字符，删除此字符及其左侧的所有内容

```
[root@node1 ~]# echo ${DOMAIN#*bai}
du.com
```

- 1
- 2

- `${var##*ba}`，自左而右，查找var变量中存储的字符串中最后一次出现的由ba所指明的字符，删除此字符及其左侧的所有内容

```
[root@node1 ~]# echo ${DOMAIN##*ba}
idu.com
```

- 1
- 2

- `${var%du*}`，自右而左，查找var变量中存储的字符串中第一次出现的由du所指明的字符，删除此字符及其右侧的所有内容

```
[root@node1 ~]# echo ${DOMAIN%du*}
www.bai
```

- 1
- 2

- `${var%%b*}`，自右而左，查找var变量中存储的字符串中最后一次出现的由b所指明的字符，删除此字符及其右侧的所有内容

```
[root@node1 ~]# echo ${DOMAIN%%b*}
www.
```

- 1
- 2

字符串切片

- 偏移4个字符显示

```
[root@node1 ~]# echo ${DOMAIN:4}
baidu.com
```

- 1
- 2

- 偏移4个字符，取5个字符

```
[root@node1 ~]# echo ${DOMAIN:4:5}
baidu
```

- 1
- 2

- 取出字符串的最后几个字符，`${var: -length}`，自右向左取

```
[root@node1 ~]# echo ${DOMAIN: -3}
com
```

- 1
- 2

注意：冒号后必须有一空白字符

查找替换

- 查找var所表示的字符串中，第一次被baidu所匹配到字符串，以cwt替换

```
[root@node1 ~]# echo ${DOMAIN/baidu/cwt}
www.cwt.com
```

- 1
- 2

- ${var//w/W}，查找var变量存储的字符中所有能够由w匹配到的内容，并替换为W

```
[root@node1 ~]# echo ${DOMAIN//w/W}
WWW.baidu.com
```

- 1
- 2

- ${var/#w/c}，行首被w所匹配到字符串，以c替换之，只会匹配一次

```
[root@node1 ~]# echo ${DOMAIN/#w/c}
cww.baidu.com
```

- 1
- 2

- ${var/%c/C}，查找var变量存储的字符中行尾能够由c匹配到的内容，并替换为C，只会匹配一次

```
[root@node1 ~]# unset DOMAIN
[root@node1 ~]# DOMIAN=www.baidu.ccc
[root@node1 ~]# echo $DOMIAN
www.baidu.ccc
[root@node1 ~]# echo ${DOMIAN/%c/C}
www.baidu.ccC
```

- 1
- 2
- 3
- 4
- 5
- 6

查找删除

```
${var/baidu}         //查找var所表示的字符串中，第一次被baidu所匹配到字符串，删除
${var//w}           //查找var所表示的字符串中，所有被w所匹配到字符串，删除
${var/#c}           //查找var所表示的字符串中，行首被c所匹配到字符串，删除
${var/%c}           //查找var所表示的字符串中，行尾被c所匹配到字符串，删除

注：删除比较简单，这里不作示例
```

- 1
- 2
- 3
- 4
- 5
- 6

字符大小写转换

- ${var^^}，把var变量中的所有小写字母，全部替换为大写字母

```
[root@node1 ~]# echo ${DOMAIN^^}
WWW.BAIDU.COM
```

- 1
- 2

- ${var,,}，把var变量中的所有大写字母，全部替换为小写字母

```
[root@node1 ~]# echo $DOMAIN
WWW.BAIDU.COM
[root@node1 ~]# echo ${DOMAIN,,}
www.baidu.com
```

- 1

- 2
- 3
- 4

变量赋值

- ${var:-word}，如果变量var为空或未声明，则返回word所表示的字符串；否则，则返回var变量的值

```
[root@node1 ~]# echo $FILE                                  //未赋值
[root@node1 ~]# echo ${FILE:-无效的变量}
无效的变量
[root@node1 ~]# echo $FILE                                  //原变量不受影响


[root@node1 ~]# FILE=root                                 //赋值
[root@node1 ~]# echo $FILE
root
[root@node1 ~]# echo ${FILE:-cwt}
root
```

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11

- ${var:+word}，如果变量var为空或未声明，忽略；否则，则返回word

```
[root@node1 ~]# unset FILE
[root@node1 ~]# echo ${FILE:+cwt}
[root@node1 ~]# FILE=tom
[root@node1 ~]# echo ${FILE:+cwt}
cwt
```

- 1
- 2
- 3
- 4
- 5

- ${var:=word}，如果变量var为空或未声明，则返回word所表示的字符串，并且把word赋值为var变量，否则，则返回var变量的值

```
[root@node1 ~]# unset FILE
[root@node1 ~]# echo $FILE                                  //未赋值

[root@node1 ~]# echo ${FILE:=cwt}
cwt
[root@node1 ~]# echo $FILE
cwt
```

- 1
- 2
- 3
- 4
- 5
- 6
- 7

- ${var:?error}，如果变量var为空或未声明，则返回error为错误信息，否则，则返回var变量的值

```
[root@node1 ~]# echo ${FILE:?error}
-bash: FILE: error
[root@node1 ~]# FILE=cwt
[root@node1 ~]# echo ${FILE:?error}
cwt
```

- 1
- 2
- 3
- 4
- 5

变量的间接引用

- 如果第一个变量的值恰好是第二个变量的变量名，从第一个变量引用第二个变量的值得方法，就称为变量的间接引用，也称为间接变量引用

```
[root@node1 ~]# A=TOM
[root@node1 ~]# TOM=B
[root@node1 ~]# eval C=\$$A
[root@node1 ~]# echo $C
B
```

- 1
- 2
- 3
- 4
- 5

获取字符串的长度

```
[root@node1 ~]# A=Hello
[root@node1 ~]# echo ${#A}
5
```

- 1
- 2
- 3