```
# VAR='1 2 $N'
# echo $VAR
1 2 $N
```

单引号是告诉 Shell 忽略特殊字符，而双引号则解释特殊符号原有的意义，比如$、！。

1.7 注释

Shell 注释也很简单，只要在每行前面加个#号，即表示 Shell 忽略解释。

第二章　Shell 字符串处理之${}

上一章节讲解了为什么用${}引用变量，${}还有一个重要的功能，就是文本处理，单行文本基本上可以满足你所有需求。

2.1 获取字符串长度

```
# VAR='hello world!'
# echo $VAR
hello world!
# echo ${#VAR}
12
```

2.2 字符串切片

格式：
${parameter:offset}
${parameter:offset:length}
截取从 offset 个字符开始，向后 length 个字符。

```
截取 hello 字符串：
# VAR='hello world!'
# echo ${VAR:0:5}
hello
截取 wo 字符：
# echo ${VAR:6:2}
wo
截取 world!字符串：
# echo ${VAR:5}
world!
截取最后一个字符：
# echo ${VAR:(-1)}
!
截取最后二个字符：
# echo ${VAR:(-2)}
```

```
d!
截取从倒数第 3 个字符后的 2 个字符：
# echo ${VAR:(-3):2}
ld
```

2.3 替换字符串

格式：${parameter/pattern/string}

```
# VAR='hello world world!'
将第一个 world 字符串替换为 WORLD：
# echo ${VAR/world/WORLD}
hello WORLD world!
将全部 world 字符串替换为 WORLD：
# echo ${VAR//world/WORLD}
hello WORLD WORLD!
替换正则匹配为空：
# VAR=123abc
# echo ${VAR//[^0-9]/}
123
# echo ${VAR//[0-9]/}
abc
```

patterm 前面开头一个正斜杠为只匹配第一个字符串，两个正斜杠为匹配所有字符。

2.4 字符串截取

格式：

```
${parameter#word}     # 删除匹配前缀
${parameter##word}
${parameter%word}     # 删除匹配后缀
${parameter%%word}
# 去掉左边，最短匹配模式，##最长匹配模式。
% 去掉右边，最短匹配模式，%%最长匹配模式。
```

```
# URL="http://www.baidu.com/baike/user.html"
以//为分隔符截取右边字符串：
# echo ${URL#*//}
www.baidu.com/baike/user.html
以/为分隔符截取右边字符串：
# echo ${URL##*/}
user.html
以//为分隔符截取左边字符串：
# echo ${URL%%//*}
http:
以/为分隔符截取左边字符串：
# echo ${URL%/*}
http://www.baidu.com/baike
```

```
以.为分隔符截取左边：
# echo ${URL%.*}
http://www.baidu.com/baike/user
以.为分隔符截取右边：
# echo ${URL##*.}
html
```

2.5 变量状态赋值

```
${VAR:-string}    如果 VAR 变量为空则返回 string
${VAR:+string}    如果 VAR 变量不为空则返回 string
${VAR:=string}    如果 VAR 变量为空则重新赋值 VAR 变量值为 string
${VAR:?string}    如果 VAR 变量为空则将 string 输出到 stderr
```

```
如果变量为空就返回 hello world!：
# VAR=
# echo ${VAR:-'hello world!'}
hello world!
如果变量不为空就返回 hello world!：
# VAR="hello"
# echo ${VAR:+'hello world!'}
hello world!
如果变量为空就重新赋值：
# VAR=
# echo ${VAR:=hello}
hello
# echo $VAR
hello
如果变量为空就将信息输出 stderr：
# VAR=
# echo ${VAR:?value is null}
-bash: VAR: value is null
```

${}主要用途大概就这么多了，另外还可以获取数组元素，在后面章节会讲到。

2.6 字符串颜色

再介绍下字符串输出颜色，有时候关键地方需要醒目，颜色是最好的方式：

字体颜色	字体背景颜色	显示方式
30：黑 31：红 32：绿 33：黄 34：蓝色 35：紫色 36：深绿 37：白色	40：黑 41：深红 42：绿 43：黄色 44：蓝色 45：紫色 46：深绿 47：白色	0：终端默认设置 1：高亮显示 4：下划线 5：闪烁 7：反白显示 8：隐藏
格式：		

```
\033[1;31;40m    # 1 是显示方式，可选。31 是字体颜色。40m 是字体背景颜色。
\033[0m          # 恢复终端默认颜色，即取消颜色设置。
```

示例：

```
#!/bin/bash
# 字体颜色
for i in {31..37}; do
      echo -e "\033[$i;40mHello world!\033[0m"
done
# 背景颜色
for i in {41..47}; do
      echo -e "\033[47;${i}mHello world!\033[0m"
done
# 显示方式
for i in {1..8}; do
      echo -e "\033[$i;31;40mHello world!\033[0m"
done
```

第三章　Shell 表达式与运算符

3.1 条件表达式

表达式	示例
[expression]	[1 -eq 1]
[[expression]]	[[1 -eq 1]]