

# 数组

网道 (WangDoc.com) , 互联网文档计划

数组 (array) 是一个包含多个值的变量。成员的编号从0开始, 数量没有上限, 也没有要求成员被连续索引。

## 目录 [隐藏]

1. 创建数组
2. 读取数组
  - 2.1 读取单个元素
  - 2.2 读取所有成员
  - 2.3 默认位置
3. 数组的长度
4. 提取数组序号
5. 提取数组成员
6. 追加数组成员
7. 删除数组
8. 关联数组

## 1. 创建数组

数组可以采用逐个赋值的方法创建。

```
ARRAY[INDEX]=value
```

## 📖 Bash 脚本教程

- 📖 1. 简介
- 📖 2. 基本语法
- 📖 3. 模式扩展
- 📖 4. 引号和转义
- 📖 5. 变量
- 📖 6. 字符串操作
- 📖 7. 算术运算
- 📖 8. 操作历史
- 📖 9. 行操作
- 📖 10. 目录堆栈
- 📖 11. 脚本入门
- 📖 12. read 命令
- 📖 13. 条件判断
- 📖 14. 循环
- 📖 15. 函数
- 📖 16. 数组
- 📖 17. set 命令, shopt 命令

上面语法中， **ARRAY** 是数组的名字，可以是任意合法的变量名。 **INDEX** 是一个大于或等于零的整数，也可以是算术表达式。注意数组第一个元素的下标是0，而不是1。

下面创建一个三个成员的数组。

```
$ array[0]=val
$ array[1]=val
$ array[2]=val
```

数组也可以采用一次性赋值的方式创建。

```
ARRAY=(value1 value2 ... valueN)
```

# 等同于

```
ARRAY=(
    value1
    value2
    value3
)
```

采用上面方式创建数组时，可以按照默认顺序赋值，也可以在每个值前面指定位置。

```
$ array=(a b c)
$ array=([2]=c [0]=a [1]=b)

$ days=(Sun Mon Tue Wed Thu Fri Sat)
$ days=([0]=Sun [1]=Mon [2]=Tue [3]=Wed [4]=Thu [5]=Fri)
```


只为某些值指定位置，也是可以的。

```
names=(hatter [5]=duchess alice)
```


上面例子中， **hatter** 是数组的0号位置， **duchess** 是5号位置， **alice** 是6号位置。

没有赋值的数组元素的默认值是空字符串。

 **18.** 脚本除错

 **19.** mktemp 命令, trap 命令

 **20.** 启动环境

 **21.** 命令提示符

## 链接

 本文源码

 代码仓库

 反馈

定义数组的时候，可以使用通配符。

```
$ mp3s=( *.mp3 )
```

上面例子中，将当前目录的所有 MP3 文件，放进一个数组。

先用 `declare -a` 命令声明一个数组，也是可以的。

```
$ declare -a ARRAYNAME
```

`read -a` 命令则是将用户的命令行输入，存入一个数组。

```
$ read -a dice
```

上面命令将用户的命令行输入，存入数组 `dice`。

## 2. 读取数组

### 2.1 读取单个元素

读取数组指定位置的成员，要使用下面的语法。

```
$ echo ${array[i]}      # i 是索引
```

上面语法里面的大括号是必不可少的，否则 Bash 会把索引部分 `[i]` 按照原样输出。

```
$ array[0]=a
```

```
$ echo ${array[0]}
```

```
a
```

```
$ echo $array[0]
```

```
a[0]
```

上面例子中，数组的第一个元素是 `a`。如果不加大括号，`Bash` 会直接读取 `$array` 首成员的值，然后将 `[0]` 按照原样输出。

## 2.2 读取所有成员

`@` 和 `*` 是数组的特殊索引，表示返回数组的所有成员。

```
$ foo=(a b c d e f)
$ echo ${foo[@]}
a b c d e f
```

这两个特殊索引配合 `for` 循环，就可以用来遍历数组。

```
for i in "${names[@]"}; do
    echo $i
done
```

`@` 和 `*` 放不放在双引号之中，是有差别的。

```
$ activities=( swimming "water skiing" canoeing "white-water"
$ for act in ${activities[@]}; \
do \
echo "Activity: $act"; \
done
```

```
Activity: swimming
Activity: water
Activity: skiing
Activity: canoeing
Activity: white-water
Activity: rafting
Activity: surfing
```

上面的例子中，数组 `activities` 实际包含5个成员，但是 `for...in` 循环直接遍历 `${activities[@]}`，导致返回7个结果。为了避免这种情况，一般把 `${activities[@]}` 放在双引号之中。

```
$ for act in "${activities[@]"}; \  
do \  
echo "Activity: $act"; \  
done
```

```
Activity: swimming  
Activity: water skiing  
Activity: canoeing  
Activity: white-water rafting  
Activity: surfing
```

上面例子中， `${activities[@]}` 放在双引号之中，遍历就会返回正确的结果。

`${activities[*]}` 不放在双引号之中，跟 `${activities[@]}` 不放在双引号之中是一样的。

```
$ for act in ${activities[*]}; \  
do \  
echo "Activity: $act"; \  
done
```

```
Activity: swimming  
Activity: water  
Activity: skiing  
Activity: canoeing  
Activity: white-water  
Activity: rafting  
Activity: surfing
```

`${activities[*]}` 放在双引号之中，所有成员就会变成单个字符串返回。

```
$ for act in "${activities[*]}"; \  
do \  
echo "Activity: $act"; \  
done
```

```
Activity: swimming water skiing canoeing white-water
```



所以，拷贝一个数组的最方便方法，就是写成下面这样。

```
$ hobbies=( "${activities[@]}" )
```

上面例子中，数组 `activities` 被拷贝给了另一个数组 `hobbies`。

这种写法也可以用来为新数组添加成员。

```
$ hobbies=( "${activities[@]}" diving )
```

上面例子中，新数组 `hobbies` 在数组 `activities` 的所有成员之后，又添加了一个成员。

## 2.3 默认位置

如果读取数组成员时，没有读取指定哪一个位置的成员，默认使用 `0` 号位置。

```
$ declare -a foo
$ foo=A
$ echo ${foo[0]}
A
```

上面例子中，`foo` 是一个数组，赋值的时候不指定位置，实际上是给 `foo[0]` 赋值。

引用一个不带下标的数组变量，则引用的是 `0` 号位置的数组元素。

```
$ foo=(a b c d e f)
$ echo ${foo}
a
$ echo $foo
a
```

上面例子中，引用数组元素的时候，没有指定位置，结果返回的是 `0` 号位置。

## 3. 数组的长度

要想知道数组的长度（即一共包含多少成员），可以使用下面两种语法。

```
${#array[*]}  
${#array[@]}
```

下面是一个例子。

```
$ a[100]=foo  
  
$ echo ${#a[*]}  
1  
  
$ echo ${#a[@]}  
1
```

上面例子中，把字符串赋值给 100 位置的数组元素，这时的数组只有一个元素。

注意，如果用这种语法去读取具体的数组成员，就会返回该成员的字符串长度。这一点必须小心。

```
$ a[100]=foo  
$ echo ${#a[100]}  
3
```

上面例子中，`${#a[100]}` 实际上是返回数组第100号成员 `a[100]` 的值（`foo`）的字符串长度。

## 4. 提取数组序号

`${!array[@]}` 或 `${!array[*]}`，可以返回数组的成员序号，即哪些位置是有值的。

```
$ arr=( [5]=a [9]=b [23]=c )  
$ echo ${!arr[@]}
```

```
5 9 23
$ echo ${!arr[*]}
5 9 23
```

上面例子中，数组的5、9、23号位置有值。

利用这个语法，也可以通过 `for` 循环遍历数组。

```
arr=(a b c d)

for i in ${!arr[@]};do
    echo ${arr[i]}
done
```

## 5. 提取数组成员

`${array[@]:position:length}` 的语法可以提取数组成员。

```
$ food=( apples bananas cucumbers dates eggs fajitas)
$ echo ${food[@]:1:1}
bananas
$ echo ${food[@]:1:3}
bananas cucumbers dates
```

上面例子中，`${food[@]:1:1}` 返回从数组1号位置开始的1个成员，`${food[@]:1:3}` 返回从1号位置开始的3个成员。

如果省略长度参数 `length`，则返回从指定位置开始的所有成员。

```
$ echo ${food[@]:4}
eggs fajitas grapes
```

上面例子返回从4号位置开始到结束的所有成员。

## 6. 追加数组成员



数组末尾追加成员，可以使用 `+=` 赋值运算符。它能够自动地把值追加到数组末尾。否则，就需要知道数组的最大序号，比较麻烦。

```
$ foo=(a b c)
$ echo ${foo[@]}
a b c
```

```
$ foo+=(d e f)
$ echo ${foo[@]}
a b c d e f
```

## 7. 删除数组

删除一个数组成员，使用 `unset` 命令。

```
$ foo=(a b c d e f)
$ echo ${foo[@]}
a b c d e f
```

```
$ unset foo[2]
$ echo ${foo[@]}
a b d e f
```

上面例子中，删除了数组中的第三个元素，下标为2。

将某个成员设为空值，可以从返回值中“隐藏”这个成员。

```
$ foo=(a b c d e f)
$ foo[1]=''
$ echo ${foo[@]}
a c d e f
```

上面例子中，将数组的第二个成员设为空字符串，数组的返回值中，这个成员就“隐藏”了。

注意，这里是“隐藏”，而不是删除，因为这个成员仍然存在，只是值变成了空值。

```
$ foo=(a b c d e f)
$ foo[1]=''
$ echo ${#foo[@]}
6
$ echo ${!foo[@]}
0 1 2 3 4 5
```

上面代码中，第二个成员设为空值后，数组仍然包含6个成员。

由于空值就是空字符串，所以下面这样写也有隐藏效果，但是不建议这种写法。

```
$ foo[1]=
```

上面的写法也相当于“隐藏”了数组的第二个成员。

直接将数组变量赋值为空字符串，相当于“隐藏”数组的第一个成员。

```
$ foo=(a b c d e f)
$ foo=''
$ echo ${foo[@]}
b c d e f
```

上面的写法相当于“隐藏”了数组的第一个成员。

`unset ArrayName` 可以清空整个数组。

```
$ unset ARRAY

$ echo ${ARRAY[*]}
<--no output-->
```

## 8. 关联数组

Bash 的新版本支持关联数组。关联数组使用字符串而不是整数作为数组索引。

`declare -A` 可以声明关联数组。

```
declare -A colors
colors["red"]="ff0000"
colors["green"]="00ff00"
colors["blue"]="0000ff"
```

关联数组必须用带有 `-A` 选项的 `declare` 命令声明创建。相比之下，整数索引的数组，可以直接使用变量名创建数组，关联数组就不行。

访问关联数组成员的方式，几乎与整数索引数组相同。

```
echo ${colors["blue"]}
```

🔖 函数

set 命令, shopt 命令 🔖

---

本教程采用知识共享 署名-相同方式共享 3.0协议。

分享本文



---

0条评论

 登录 ▼

开始讨论...

通过以下方式登录

或注册一个 **DISQUS** 帐号 

姓名



分享

最佳

最新

最早

来做第一个留言的人吧！

---

订阅

隐私

不要出售我的数据

联系：contact@wangdoc.com