declare

声明变量，设置或显示变量的值和属性。

语法

```
declare [-aAfFgilnrtux] [-p] [name[=value] ...]
```

主要用途

- 显示包含指定属性的全部变量和值
- 显示包含指定属性的一到多个变量和值
- 显示一到多个变量的属性和值
- 显示所有变量的属性和值并显示函数的定义
- 显示所有变量的属性和值
- 显示所有全局变量的属性和值
- 显示全部函数名和函数定义
- 只显示全部函数名
- 显示一到多个函数名和函数定义
- 只显示一到多个函数名
- 声明全局变量（可选：赋值）
- 声明变量（可选：赋值、属性）
- 增加、删除变量的属性（可选：赋值）

选项

```
-f 将操作或显示限制为函数名及函数定义。
-F 只显示函数名（调试时附加行号和源文件）。
-g 在shell函数中使用时创建全局变量；其他情况下忽略。
-p 显示每个名称的属性和值。

*设置属性的选项:
-a 创建数组（如果支持）。
-A 创建关联数组（如果支持）。
-i 增加整型属性。
+i 删除整型属性。
-l 增加小写属性，变量的值将转换为小写。
+l 删除小写属性。
-n 增加引用属性（如果该选项存在）。
+n 删除引用属性（如果该选项存在）。
-r 增加只读属性。
-t 增加追踪属性。
+t 删除追踪属性。
-u 增加大写属性，变量的值将转换为大写。
+u 删除大写属性。
-x 增加导出属性。
+x 删除导出属性。
```

参数

name（可选）：变量名或函数名。
value（可选）：变量的值。

返回值

declare 返回true除非你提供了非法选项或赋值错误。具体导致异常的情况请查看**讨论**章节的**关于异常情况**。

例子

```
# 声明变量，当然也欢迎您在这个网站（感谢本项目发起人 @jaywcjlove）查询linux命令。
declare reference_website='https://wangchujiang.com/linux-command/'

# 显示所有包含整型属性的变量和值。
declare -i
# 定义变量b并赋值为3，具有整型属性。
declare -i b=5
# 显示属性，返回 declare -i b="5"。
declare -p b
# 删除整型属性。
declare +i b
# 显示属性，返回 declare -- b="5"。
declare -p b
# 根据变量属性强制转换值的英文大小写。
declare -u uc_var='abc'
declare -l lc_var='ABC'
# 显示'ABC abc';
echo "${uc_var} ${lc_var}"
```

```
# 定义函数内的全局变量
function test(){
  declare -g a=3
 # 或者
  local -g b=3
 # 或者
  c=3
 # 让我们查看它们的属性。
  declare -p a b c
}
# 执行函数。
test
# 返回结果。
# declare -- a="3"
# declare -- b="3"
# declare -- c="3"

# 定义函数外的全局变量
declare a=3
b=3
declare –p a b
```

```
# 返回结果如下。
# declare -- a="3"
# declare -- b="3"

# 定义局部变量
function test2(){
  local -i a=3
  declare -i b=3
}
test2
# 没有该变量（已经被销毁了）
echo "${a} ${b}"
# 因此，我们日常脚本中最常见的类似于'a=3'实际上是声明并赋值了一个全局变量。
# 在接下来的 **讨论** 环节会延伸讨论全局和局部变量问题。
```

```
# 注意，不能使用 `+a` 或`+A` 取消数组，也不能使用 `+r` 取消只读属性。

# 定义只读数组，设置属性的同时定义赋值。
declare -ar season=('Spring' 'Summer' 'Autumn' 'Winter')
# 或者这样。
season=('Spring' 'Summer' 'Autumn' 'Winter')
declare -ar season
# 显示所有数组。
declare -a
# 定义关联数组。

declare -A fruits=(['apple']='red' ['banana']='yellow')
# 显示所有关联数组。
declare -A
```

```
# 显示所有变量的属性和值并显示函数的定义，输出很长。
declare
# 显示所有变量的属性和值。
declare -p
# 显示所有全局变量的属性和值。
declare -g
```

```
# 显示全部函数名和函数定义。
declare -f
# 只显示全部函数名。
declare -F

# 定义两个函数。
function func_a(){ echo $(date +"%F %T"); }
function func_b(){ cd /; ls -lh --sort=time; }
# 显示一到多个函数名和函数定义。
declare -f func_a func_b
# 只显示一到多个函数名，验证某个名称是否已经定义为函数时有用。
declare -F func_a func_b
# 最好不要让函数名和变量名相同。
```

讨论

1. 全局和局部变量

   正如上面**例子**指出的情况，我们在日常编写程序的时候需要了解这些概念，在这里做个简要地介绍，当然你也可以很方便的搜索到相关内容。

   - 全局变量：在整个脚本执行期间，只要没有被删除就**一直存在**
   - 局部变量：在函数内定义，函数执行后就被删除。

   建议函数内使用`local`命令，函数外使用`declare`命令。

   > *不要在脚本中定义过多的全局变量，那样可能会被其他函数调用造成意料之外的后果，并且也不方便检查出来。*
   >
   > *更不用说缺乏必要的注释了 —— ZhuangZhu-74*

   相关资料：

   - google提供的编码规范
   - 全局变量的讨论

2. 关于`declare typeset export local readonly`命令

   为什么`declare`能做到的事，还需要定义其他这些命令呢？

   因为这样语句含义会更加明确，例如：

   - 设置导出属性的变量时，`export var`和`declare -x var`
   - 在函数内声明变量时，使用`local`
   - 声明只读变量，使用`readonly`

   `typeset`和`declare`命令一样。

3. 关于异常情况

   有多种原因导致`declare`失败，关于这些情况可以参考bash在线文档declare部分(最新版)，或执行`info bash`查看`declare`部分最后一大串`an attempt is`开头的句子。

注意

1. 该命令是bash内建命令，相关的帮助信息请查看`help`命令。
2. 导出属性的相关介绍请查看'export'命令。
3. 只读属性的相关介绍请查看'readonly'命令。
4. 引用属性的相关介绍请查看'unset'命令的例子部分。