**Shell 传递参数**

我们可以在执行 Shell 脚本时，向脚本传递参数，脚本内获取参数的格式为：$n。n 代表一个数字，1 为执行脚本的第一个参数，2 为执行脚本的第二个参数，以此类推……

**实例**

以下实例我们向脚本传递三个参数，并分别输出，其中 $0 为执行的文件名：

#!/bin/bash

# author:菜鸟教程

# url:www.runoob.com

echo "Shell 传递参数实例！";

echo "执行的文件名：$0";

echo "第一个参数为：$1";

echo "第二个参数为：$2";

echo "第三个参数为：$3";

为脚本设置可执行权限，并执行脚本，输出结果如下所示：

$ chmod +x test.sh

$ ./test.sh 1 2 3

Shell 传递参数实例！

执行的文件名：./test.sh

第一个参数为：1

第二个参数为：2

第三个参数为：3

另外，还有几个特殊字符用来处理参数：

**参数处理 说明**

$# 传递到脚本的参数个数

$\* 以一个单字符串显示所有向脚本传递的参数。

如"$\*"用「"」括起来的情况、以"$1 $2 … $n"的形式输出所有参数。

$$ 脚本运行的当前进程ID号

$! 后台运行的最后一个进程的ID号

$@ 与$\*相同，但是使用时加引号，并在引号中返回每个参数。

如"$@"用「"」括起来的情况、以"$1" "$2" … "$n" 的形式输出所有参数。

$- 显示Shell使用的当前选项，与set命令功能相同。

$? 显示最后命令的退出状态。0表示没有错误，其他任何值表明有错误。

#!/bin/bash

# author:菜鸟教程

# url:www.runoob.com

echo "Shell 传递参数实例！";

echo "第一个参数为：$1";

echo "参数个数为：$#";

echo "传递的参数作为一个字符串显示：$\*";

执行脚本，输出结果如下所示：

$ chmod +x test.sh

$ ./test.sh 1 2 3

Shell 传递参数实例！

第一个参数为：1

参数个数为：3

传递的参数作为一个字符串显示：1 2 3

$\* 与 $@ 区别：

相同点：都是引用所有参数。

不同点：只有在双引号中体现出来。假设在脚本运行时写了三个参数 1、2、3，，则 " \* " 等价于 "1 2 3"（传递了一个参数），而 "@" 等价于 "1" "2" "3"（传递了三个参数）。

#!/bin/bash

# author:菜鸟教程

# url:www.runoob.com

echo "-- \$\* 演示 ---"

for i in "$\*"; do

echo $i

done

echo "-- \$@ 演示 ---"

for i in "$@"; do

echo $i

done

执行脚本，输出结果如下所示：

$ chmod +x test.sh

$ ./test.sh 1 2 3

-- $\* 演示 ---

1 2 3

-- $@ 演示 ---

1

2

3

在为shell脚本传递的参数中如果包含空格，应该使用单引号或者双引号将该参数括起来，以便于脚本将这个参数作为整体来接收。

在有参数时，可以使用对参数进行校验的方式处理以减少错误发生：

if [ -n "$1" ]; then

echo "包含第一个参数"

else

echo "没有包含第一参数"

fi

注意：中括号 [] 与其中间的代码应该有空格隔开

Shell 里面的中括号（包括单中括号与双中括号）可用于一些条件的测试：

算术比较, 比如一个变量是否为0, [ $var -eq 0 ]。

文件属性测试，比如一个文件是否存在，[ -e $var ], 是否是目录，[ -d $var ]。

字符串比较, 比如两个字符串是否相同， [[ $var1 = $var2 ]]。

[] 常常可以使用 test 命令来代替，具体可参看：Shell 中的中括号用法总结。