

read 命令

网道 (WangDoc.com) , 互联网文档计划

目录 [隐藏]

- 1. 用法
- 2. 参数
- 3. IFS 变量

1. 用法

有时，脚本需要在执行过程中，由用户提供一部分数据，这时可以使用 `read` 命令。它将用户的输入存入一个变量，方便后面的代码使用。用户按下回车键，就表示输入结束。

`read` 命令的格式如下。

```
read [-options] [variable...]
```

上面语法中，`options` 是参数选项，`variable` 是用来保存输入数值的一个或多个变量名。如果没有提供变量名，环境变量 `REPLY` 会包含用户输入的一整行数据。

下面是一个例子 `demo.sh` 。

```
#!/bin/bash
```

```
echo -n "输入一些文本 > "
```

📖 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

```
read text
echo "你的输入: $text"
```

上面例子中，先显示一行提示文本，然后会等待用户输入文本。用户输入的文本，存入变量 `text`，在下一行显示出来。

```
$ bash demo.sh
输入一些文本 > 你好，世界
你的输入: 你好，世界
```

`read` 可以接受用户输入的多个值。

```
#!/bin/bash
echo Please, enter your firstname and lastname
read FN LN
echo "Hi! $LN, $FN !"
```

上面例子中，`read` 根据用户的输入，同时为两个变量赋值。

如果用户的输入项少于 `read` 命令给出的变量数目，那么额外的变量值为空。如果用户的输入项多于定义的变量，那么多余的输入项会包含到最后一个变量中。


如果 `read` 命令之后没有定义变量名，那么环境变量 `REPLY` 会包含所有的输入。


```
#!/bin/bash
# read-single: read multiple values into default v
echo -n "Enter one or more values > "
read
echo "REPLY = '$REPLY'"
```


上面脚本的运行结果如下。

```
$ read-single
Enter one or more values > a b c d
REPLY = 'a b c d'
```

 **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](#)

`read` 命令除了读取键盘输入，可以用来读取文件。

```
#!/bin/bash

filename='/etc/hosts'

while read myline
do
    echo "$myline"
done < $filename
```

上面的例子通过 `read` 命令，读取一个文件的内容。 `done` 命令后面的定向符 `<`，将文件内容导向 `read` 命令，每次读取一行，存入变量 `myline`，直到文件读取完毕。

2. 参数

`read` 命令的参数如下。

(1) -t 参数

`read` 命令的 `-t` 参数，设置了超时的秒数。如果超过了指定时间，用户仍然没有输入，脚本将放弃等待，继续向下执行。

```
#!/bin/bash

echo -n "输入一些文本 > "
if read -t 3 response; then
    echo "用户已经输入了"
else
    echo "用户没有输入"
fi
```

上面例子中，输入命令会等待3秒，如果用户超过这个时间没有输入，这个命令就会执行失败。 `if` 根据命令的返回值，转入 `else` 代码块，继续往下执行。

环境变量 `TMOUT` 也可以起到同样作用，指定 `read` 命令等待用户输入的时间（单位为秒）。

```
$ TMOUT=3
$ read response
```

上面例子也是等待3秒，如果用户还没有输入，就会超时。

(2) -p 参数

-p 参数指定用户输入的提示信息。

```
read -p "Enter one or more values > "
echo "REPLY = '$REPLY'"
```

上面例子中，先显示 `Enter one or more values >`，再接受用户的输入。

(3) -a 参数

-a 参数把用户的输入赋值给一个数组，从零号位置开始。

```
$ read -a people
alice duchess dodo
$ echo ${people[2]}
dodo
```

上面例子中，用户输入被赋值给一个数组 `people`，这个数组的2号成员就是 `dodo`。

(4) -n 参数

-n 参数指定只读取若干个字符作为变量值，而不是整行读取。

```
$ read -n 3 letter
abcdefghijkl
$ echo $letter
abc
```

上面例子中，变量 `letter` 只包含3个字母。

(5) -e 参数

`-e` 参数允许用户输入的时候，使用 `readline` 库提供的快捷键，比如自动补全。具体的快捷键可以参阅《行操作》一章。

```
#!/bin/bash

echo Please input the path to the file:

read -e fileName

echo $fileName
```

上面例子中，`read` 命令接受用户输入的文件名。这时，用户可能想使用 Tab 键的文件名“自动补全”功能，但是 `read` 命令的输入默认不支持 `readline` 库的功能。`-e` 参数就可以允许用户使用自动补全。

(6) 其他参数

- `-d delimiter`：定义字符串 `delimiter` 的第一个字符作为用户输入的结束，而不是一个换行符。
- `-r`：raw 模式，表示不把用户输入的反斜杠字符解释为转义字符。
- `-s`：使得用户的输入不显示在屏幕上，这常常用于输入密码或保密信息。
- `-u fd`：使用文件描述符 `fd` 作为输入。

3. IFS 变量

`read` 命令读取的值，默认是以空格分隔。可以通过自定义环境变量 `IFS`（内部字段分隔符，Internal Field Separator 的缩写），修改分隔标志。

`IFS` 的默认值是空格、Tab 符号、换行符号，通常取第一个（即空格）。

如果把 `IFS` 定义成冒号（`:`）或分号（`;`），就可以分隔以这两个符号分隔的值，这对读取文件很有用。

```
#!/bin/bash
# read-ifs: read fields from a file

FILE=/etc/passwd

read -p "Enter a username > " user_name
file_info="$(grep "^$user_name:" $FILE)"

if [ -n "$file_info" ]; then
    IFS=":" read user pw uid gid name home shell <<<
    echo "User = '$user'"
    echo "UID = '$uid'"
    echo "GID = '$gid'"
    echo "Full Name = '$name'"
    echo "Home Dir. = '$home'"
    echo "Shell = '$shell'"
else
    echo "No such user '$user_name'" >&2
    exit 1
fi
```

上面例子中，`IFS` 设为冒号，然后用来分解 `/etc/passwd` 文件的一行。`IFS` 的赋值命令和 `read` 命令写在一行，这样的话，`IFS` 的改变仅对后面的命令生效，该命令执行后 `IFS` 会自动恢复原来的值。如果不写在一行，就要采用下面的写法。

```
OLD_IFS="$IFS"
IFS=":"
read user pw uid gid name home shell <<< "$file_in
IFS="$OLD_IFS"
```

另外，上面例子中，`<<<` 是 Here 字符串，用于将变量值转为标准输入，因为 `read` 命令只能解析标准输入。

如果 `IFS` 设为空字符串，就等同于将整行读入一个变量。

```
#!/bin/bash
input="/path/to/txt/file"
while IFS= read -r line
```

```
do
    echo "$line"
done < "$input"
```

上面的命令可以逐行读取文件，每一行存入变量 `line`，打印出来以后再读取下一行。

[脚本入门](#)

[条件判断](#)

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      

联系：contact@wangdoc.com