jq

一个灵活的轻量级命令行JSON处理器

补充说明

jq 是 stedolan 开发的一个轻量级的和灵活的命令行JSON处理器，源码请参考 jq 项目主页

jq 用于处理JSON输入，将给定过滤器应用于其JSON文本输入并在标准输出上将过滤器的结果生成为JSON

最简单的过滤器是.，它将jq的输入未经修改地复制到其输出中（格式设置除外）。

请注意jq 当前仅支持64位双精度浮点数IEEE754

安装

```
# Debian系，如 Ubuntu
sudo apt-get install jq

# RedHat系,如 CentOS
yum install jq
```

语法

```
jq [options] <jq filter> [file...]
jq [options] --args <jq filter> [strings...]
jq [options] --jsonargs <jq filter> [JSON_TEXTS...]
```

选项

```
-c        紧凑而不是漂亮的输出;
-n        使用`null`作为单个输入值;
-e        根据输出设置退出状态代码;
-s        将所有输入读取（吸取）到数组中；应用过滤器;
-r        输出原始字符串，而不是JSON文本;
-R        读取原始字符串，而不是JSON文本;
-C        为JSON着色;
-M        单色（不要为JSON着色）;
-S        在输出上排序对象的键;
--tab     使用制表符进行缩进;
--arg a v         将变量$a设置为value<v>;
--argjson a v     将变量$a设置为JSON value<v>;
--slurpfile a f   将变量$a设置为从<f>读取的JSON文本数组;
--rawfile a f     将变量$a设置为包含<f>内容的字符串;
--args    其余参数是字符串参数，而不是文件;
--jsonargs    其余的参数是JSON参数，而不是文件;
--        终止参数处理;
```

例子

.: 以漂亮的方式输出

```
$ echo '{ "foo": { "bar": { "baz": 123 } } }' | jq '.'
{
  "foo": {
    "bar": {
      "baz": 123
    }
  }
}
```

`.foo, .foo.bar, .foo?`: 获取一个键的值

```
$ echo '{"foo": 42, "bar": "less interesting data"}' | jq '.foo'
42
```

`.[], .[]?, .[2], .[10:15]`: 数组运算

```
$ echo '[{"name":"JSON", "good":true}, {"name":"XML", "good":false}]' | jq
'.[1]'
{
  "name": "XML",
  "good": false
}
```

`[], {}`: 构造一个数组/对象

```
$ echo '{"user":"stedolan","titles":["JQ Primer", "More JQ"]}' | jq '{user,
title: .titles[]}'

{
  "user": "stedolan",
  "title": "JQ Primer"
}
{
  "user": "stedolan",
  "title": "More JQ"
}
```

`length`: 计算一个值的长度

```
$ echo '[[1,2], "string", {"a":2}, null]' | jq '.[] | length'
2
6
1
0
```

`keys`: 取出数组中的键

```
$ echo '{"abc": 1, "abcd": 2, "Foo": 3}' | jq 'keys'
[
  "Foo",
```

```
  "abc",
  "abcd"
]
```

`,`: 使用多个过滤器

```
$ echo '{ "foo": 42, "bar": "something else", "baz": true}' | jq '.foo,
.bar'
42
"something else"
```

`|`: 通过管道将一个过滤器的输出当做下一个过滤器的输入

```
$ echo '[{"name":"JSON", "good":true}, {"name":"XML", "good":false}]' | jq
'.[] | .name'
"JSON"
"XML"
```

`select(foo)`: 如果foo返回true，则输入保持不变

```
$ echo '[1,5,3,0,7]' | jq 'map(select(. >= 2))'
[
  5,
  3,
  7
]
```

`map(foo)`: 每个输入调用过滤器

```
$ echo '[1,2,3]' | jq 'map(.+1)'
[
  2,
  3,
  4
]
```

`if-then-else-end`: 条件判断

```
 $ echo '2' | jq 'if . == 0 then "zero" elif . == 1 then "one" else "many"
end'

"many"
```

`\(foo)`: 在字符串中插入值并进行运算

```
$ echo '42' | jq '"The input was \(.), which is one less than \(.+1)"'

"The input was 42, which is one less than 43"
```