LaTeX3 教程二：变量，函数及基本程序结构

项子越

ziyue.alan.xiang@gmail.com

https://github.com/xziyue/latex3-chinese-video

2021 年 3 月 22 日

- 变量的声明和使用
- 函数的声明和使用
- 循环语句
- 条件语句

声明变量：使用new结尾的函数

- `\bool_new:N`
- `\int_new:N`
- `\seq_new:N`
- `\dim_new:N`
- `\fp_new:N`

- tl: 凭据表
- str：字符串
- int：整型
- fp：浮点数
- seq：队列
- dim：尺度/长度
- bool：布尔型

- N：接收一个命令，传递命令本身。
- n：接收一个凭据表。

设置变量：使用set结尾的函数

- `\int_set:Nn`
- `\dim_set:Nn`
- `\fp_set:Nn`
- `\bool_set_true:N`
- `\bool_set_false:N`

- tl: 凭据表
- str：字符串
- int：整型
- fp：浮点数
- seq：队列
- dim：尺度/长度
- bool：布尔型

- N：接收一个命令，传递命令本身。
- n：接收一个凭据表。

使用变量：使用use结尾的函数

- `\int_use:N`
- `\dim_use:N`
- `\fp_use:N`
- `\tl_use:N`
- `\str_use:N`

- tl: 凭据表
- str：字符串
- int：整型
- fp：浮点数
- seq：队列
- dim：尺度/长度
- bool：布尔型

- N：接收一个命令，传递命令本身。
- n：接收一个凭据表。

声明函数

使用\cs_set:Npn来声明函数。

```
\ExplSyntaxOn
\cs_set:Npn \my_function:n #1 {
    你输入了: #1
}
\par\my_function:n {一}
\par\my_function:n {二}
\ExplSyntaxOff
```

```
你输入了：一
你输入了：二
```

查阅函数文档

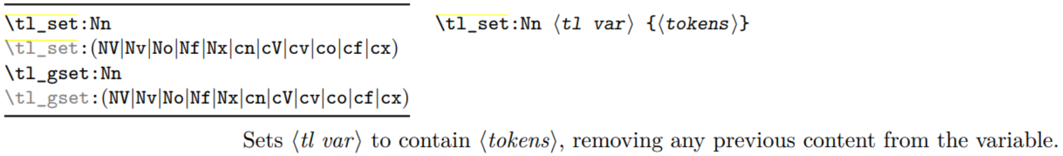
获取 LaTeX3 文档

- 搜索“CTAN l3kernel”
- 点击“The LATEX3 interfaces”

链接：`http://mirrors.ctan.org/macros/latex/contrib/l3kernel/interface3.pdf`

- 每一个章节对应一个 LaTeX3 库
- 每一个章节内的二级章节对应一系列功能类似的函数

LaTeX3 文档中的函数条目

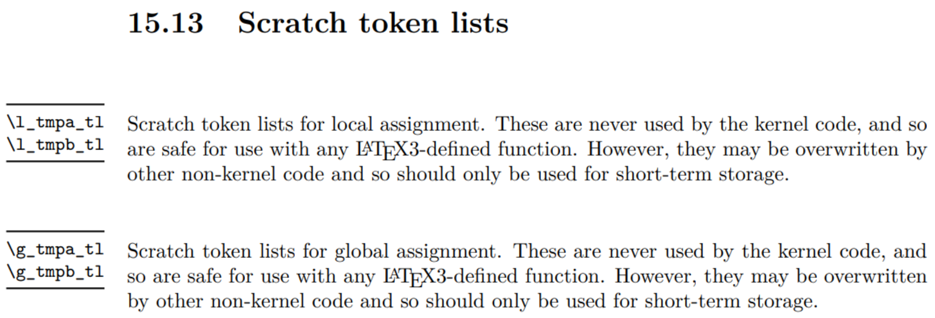


```
\tl_set:Nn
\tl_set:(NV|Nv|No|Nf|Nx|cn|cV|cv|co|cf|cx)
\tl_gset:Nn
\tl_gset:(NV|Nv|No|Nf|Nx|cn|cV|cv|co|cf|cx)
```

```
\tl_set:Nn ⟨tl var⟩ {⟨tokens⟩}
```

Sets ⟨*tl var*⟩ to contain ⟨*tokens*⟩, removing any previous content from the variable.

LaTeX3 文档中预定义的变量（scratch variables）



15.13 Scratch token lists

`\l_tmpa_tl` `\l_tmpb_tl`	Scratch token lists for local assignment. These are never used by the kernel code, and so are safe for use with any LaTeX3-defined function. However, they may be overwritten by other non-kernel code and so should only be used for short-term storage.
`\g_tmpa_tl` `\g_tmpb_tl`	Scratch token lists for global assignment. These are never used by the kernel code, and so are safe for use with any LaTeX3-defined function. However, they may be overwritten by other non-kernel code and so should only be used for short-term storage.

- 在文档比较庞大时，尽量避免使用这些变量以防止冲突

案例：加法

```
\ExplSyntaxOn
\int_new:N \l_my_tmpa_int
\int_new:N \l_my_tmpb_int
\int_set:Nn \l_my_tmpa_int {200}
\int_set:Nn \l_my_tmpb_int {10}
\int_eval:n {\l_my_tmpa_int + \l_my_tmpb_int}
\ExplSyntaxOff
```

```
210
```

基于整数的循环

```
\ExplSyntaxOn
\int_step_inline:nn {20} {
    #1,~
}
\ExplSyntaxOff
```

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
15, 16, 17, 18, 19, 20,

基于整数的循环

改变起始数值

```
\ExplSyntaxOn
\int_step_inline:nnn {10} {20} {
    #1,~
}
\ExplSyntaxOff
```

```
10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
```

基于整数的循环

将循环变量保存在凭据表中

```
\ExplSyntaxOn
\int_step_variable:nNn {20} \l_tmpa_tl {
    \tl_use:N \l_tmpa_tl,~
}
\ExplSyntaxOff
```

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,

基于整数的循环

二重循环

```
\ExplSyntaxOn
\int_step_inline:nn {5} {
    \int_step_inline:nn {5} {
        (#1,##1), ~
    }
}
\ExplSyntaxOff
```

(1,1), (1,2), (1,3), (1,4), (1,5), (2,1), (2,2), (2,3), (2,4), (2,5), (3,1), (3,2), (3,3), (3,4), (3,5), (4,1), (4,2), (4,3), (4,4), (4,5), (5,1), (5,2), (5,3), (5,4), (5,5),

案例：$1 + 2 + \ldots + 100 = ?$

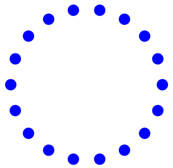
```
\ExplSyntaxOn
\int_set:Nn \l_tmpa_int {0}
\int_step_inline:nn {100} {
    \int_add:Nn \l_tmpa_int {#1}
}
\int_use:N \l_tmpa_int
\ExplSyntaxOff
```

```
5050
```

案例：圆上的点

圆的参数方程： $\begin{cases} x = r\cos\theta \\ y = r\sin\theta \end{cases}$

```
\ExplSyntaxOn
\begin{tikzpicture}
\int_step_inline:nnn {0} {17} {
    \fp_set:Nn \l_tmpa_fp {20 * (#1) *
     ↪  \c_one_degree_fp}
    \node[minimum~width=1.5mm, fill=blue,
        draw=none, circle, inner~sep=0pt] at
    (\fp_eval:n {cos(\l_tmpa_fp)},
     \fp_eval:n {sin(\l_tmpa_fp)}) {};
}
\end{tikzpicture}
\ExplSyntaxOff
```



整数判断

```
\ExplSyntaxOn
\cs_set:Npn \my_if_less_than_two:n #1 {
    \int_compare:nNnTF {#1} < {2} {
        \zhnumber{#1} 小于二
    } {
        \zhnumber{#1} 大于等于二
    }
}
\par\my_if_less_than_two:n {1}
\par\my_if_less_than_two:n {2}
\par\my_if_less_than_two:n {3}
\ExplSyntaxOff
```

一小于二
二大于等于二
三大于等于二

整数判断

```latex
\ExplSyntaxOn
\cs_set:Npn \my_if_less_than_two:n #1 {
    \int_compare:nTF {#1 <= 2} {
        \zhnumber{#1} 小于等于二
    } {
        \zhnumber{#1} 大于二
    }
}
\par\my_if_less_than_two:n {1}
\par\my_if_less_than_two:n {2}
\par\my_if_less_than_two:n {3}
\ExplSyntaxOff
```

```
一小于等于二
二小于等于二
三大于二
```

布尔判断

- 使用`\bool_if:nTF`可以进行布尔判断；其表达式参数支持&&, ||, ()等逻辑运算符
- 一般的判断语句还有_p变体，例如`\int_compare_p:n`, `\bool_if_p:n`等。这些函数不是根据判断结果执行分支，而是直接返回判断结果为真或为假
- 这些“判别式”（predicate）可以帮助我们构建复杂的逻辑语句

案例：偶数判断

```
\ExplSyntaxOn
\cs_gset:Npn \my_if_even_p:n #1 {
    \int_compare_p:nNn {\int_mod:nn {#1}{2}} = {0}
}
\cs_set:Npn \my_even_check:n #1 {
    \bool_if:nTF { \my_if_even_p:n {#1}} {
        \zhnumber{#1}是偶数
    } {
        \zhnumber{#1}是奇数
    }
}
\par \my_even_check:n{1}
\par \my_even_check:n{2}
\par \my_even_check:n{3}
\ExplSyntaxOn
```

一是奇数
二是偶数
三是奇数

案例：双偶数判断

```
\ExplSyntaxOn
\cs_set:Npn \my_double_even_check:nn #1#2 {
    \bool_if:nTF { \my_if_even_p:n {#1} && \my_if_even_p:n {#2}} {
        \zhnumber{#1}和\zhnumber{#2}都是偶数
    } {
        \zhnumber{#1}和\zhnumber{#2}不都是偶数
    }
}
\par \my_double_even_check:nn {1} {3}
\par \my_double_even_check:nn {1} {2}
\par \my_double_even_check:nn {2} {4}
\ExplSyntaxOn
```

一和三不都是偶数
一和二不都是偶数
二和四都是偶数

条件循环语句

诸如\int_do_while:nNnn, \bool_do_while:nn等语句每一次循环就进行一次判断，直到判断为假。

```
\ExplSyntaxOn
\int_set:Nn \l_tmpa_int {1}
\int_set:Nn \l_tmpb_int {0}
\int_do_while:nNnn {\l_tmpa_int} < {101} {
    \int_add:Nn \l_tmpb_int {\l_tmpa_int}
    \int_incr:N \l_tmpa_int
}
\int_use:N \l_tmpb_int
\ExplSyntaxOff
```

5050

条件循环语句

诸如\int_do_until:nNnn, \bool_do_until:nn等语句每一次循环就进行一次判断，直到判断为真。

```
\ExplSyntaxOn
\int_set:Nn \l_tmpa_int {1}
\int_set:Nn \l_tmpb_int {0}
\int_do_until:nNnn {\l_tmpa_int} > {100} {
    \int_add:Nn \l_tmpb_int {\l_tmpa_int}
    \int_incr:N \l_tmpa_int
}
\int_use:N \l_tmpb_int
\ExplSyntaxOff
```

```
5050
```