1 比较 LaTeX3 中不同的展开方式

在本测试中首先定义的参数如下:

```
\tl_set:Nn \tla {April~Fool!}
\tl_set:Nn \tlb {Ahahaha!}
\tl_set:Nx \tlc {\tla~text1~\tlb~text2}
\tl_set:Ne \tld {\tla~text1~\tlb~text2}
\tl_set:Nf \tle {\tla~text1~\tlb~text2}
```

同时定义的用于测试的宏如下:

```
\newcommand\foo{April Fool!}
\NewDocumentCommand{\baz}{}{\foo Ahahahaha!}
\NewDocumentCommand{\foobaz}{}{\foo Ahahahaha! \baz}
\NewExpandableDocumentCommand{\expbaz}{}{\foo Ahahahaha!}
\NewExpandableDocumentCommand{\expfoobaz}{}{\foo Ahahahaha! \baz}
```

1.1 使用 x 型,e 型和 f 型参数展开文本

使用 x 型展开实现完全展开, 此时宏\tla与\tlb均被展开; 使用 e 型展开等同于pdfLaTeX中的\expanded命令, 同样也将其完全展开了; 而 f 型展开在遇到第一个不可展开的参数之后则会停止展开, 因此\tlb没有被展开.

macro:->April Fool!text1 Ahahaha!text2
macro:->April Fool!text1 Ahahaha!text2
macro:->April Fool!text1 \tlb text2

1.2 使用 f 型参数展开不同定义的宏

测试使用 f 型参数展开:f 型参数将进行递归展开, 当遇到不可展开的参数时停止展开, 当遇到的这个不可展开的参数是空格 <spacetoken> 时, 将去掉 (gobble) 这一空格之后停止展开. 因此在下面的测试结果中, 对于\tlc与\tle, 均只展开到第一个空格处停止,\baz并不会被展开.

The expansion of parameter 'f':
macro:->April Fool!
macro:->April Fool!Ahahahaha!
macro:->April Fool!Ahahahaha! \baz

macro:->April Fool!Ahahahaha!
macro:->April Fool!Ahahahaha! \baz

1.3 使用 x 型和 e 型参数展开不同定义的宏

测试使用 x 型和 e 型参数展开:x 型展开进行的是完全展开, 展开行为等同于`\edef`进行的展开;e 型展开进行的是类似于pdfLaTeX中提供的`\expanded`命令的展开. 而这两种展开只会展开非 protected 宏. 下面逐一分析这些宏的原始定义可得:

- `\foo`使用`\newcomand`定义, 不是 protected 宏, 因此会被展开
- `\baz`和`\foobaz`使用`\NewDocumentCommand`定义, 属于 protected 宏, 因此不会被展开
- `\expbaz`使用`\NewExpandableDocumentCommand`定义, 不属于 protected 宏, 因此这两种展开可以展开`\expbaz`, 首先得到`\foo Ahahahaha!`, 之后递归地进行展开, 由于`\foo`是非 protected 宏是可以展开的, 最终得到`April Fool!Ahahahaha!`
- `\expfoobaz`类似于上面, 可以先被第一次展开得到`\foo Ahahahaha! \baz`, 又由于`\foo`为非 protected 宏而`\baz`为不可展开的 protected 宏, 最终得到`April Fool!Ahahahaha! \baz`

The expansion of parameter 'x':
macro:->April Fool!
macro:->\baz
macro:->\foobaz
macro:->April Fool!Ahahahaha!
macro:->April Fool!Ahahahaha! \baz

The expansion of parameter 'e':
macro:->April Fool!
macro:->\baz
macro:->\foobaz
macro:->April Fool!Ahahahaha!
macro:->April Fool!Ahahahaha! \baz

2 使用 expl3 进行的小测试

非常感谢雾月大神提供的例子和详细的介绍

```
\int_zero:N \l_tmpa_int
\flag_clear:N \l_tmpa_flag
\tl_set:Ne \l_tmpa_tl{%使用e型展开 int 与 flag

    \int_incr:N \l_tmpa_int
    \int_use:N  \l_tmpa_int
}
\tl_set:Ne \l_tmpb_tl{
    \flag_raise:N  \l_tmpa_flag
    \flag_height:N \l_tmpa_flag
}
```

```
\int_zero:N \l_tmpa_int
\flag_clear:N \l_tmpa_flag
\tl_set:Nx \l_tmpa_tl{%使用x型展开 int 与 flag
    \int_incr:N \l_tmpa_int
    \int_use:N  \l_tmpa_int
}
\tl_set:Nx \l_tmpb_tl{
    \flag_raise:N  \l_tmpa_flag
    \flag_height:N \l_tmpa_flag
}
```

```
\int_zero:N \l_tmpa_int
\flag_clear:N \l_tmpa_flag
\tl_set:Nf \l_tmpa_tl{%使用f型展开 int 与 flag
    \int_incr:N \l_tmpa_int
    \int_use:N  \l_tmpa_int
}
\tl_set:Nf \l_tmpb_tl{
    \flag_raise:N  \l_tmpa_flag
    \flag_height:N \l_tmpa_flag
}
```

使用 e 型展开的结果:

int result = 0

flag result = 1

使用 x 型展开的结果:

int result = 0

flag result = 1

使用 f 型展开的结果:

int result = 1

flag result = 1

查阅文档 interface.pdf 之后发现`\int_incr:N`没有任何星号后缀, 属于正常展开的命令, 而`\flag_incr:N`带有实心星号后缀, 将会被 e 型参数和 x 型参数提前完全展开, 因此上述实验中前两者的展开结果为`\flag_height=1`.