zhlipsum: 中文乱数假文（Lorem ipsum）

曾祥东

2020/04/10　v1.2.0*

如彭奇和瓦特曼的公共事业所证实的那样有一个胡子雪雪白的上帝超越时间超越空间确确实实存在他在神圣的冷漠神圣的疯狂神圣的喑哑的高处深深地爱着我们除了少数的例外不知什么原因但时间将会揭示他像神圣的密兰达一样和人们一起忍受着痛苦这班人不知什么原因但时间将会揭示生活在痛苦中生活在烈火中这烈火这火焰如果继续燃烧毫无疑问将使穹苍着火也就是说将地狱炸上天去天是那么蓝那么澄澈那么平静这种平静尽管时断时续总比没有好得多但是别这么快还要进一步考虑到泰斯丢和丘那德的人体测定学院的未完成的研究结果早已断定毫无疑问换句话说除了依附着人类的疑问之外别无其他疑问根据泰斯丢和丘那德的未完成的劳动的结果早已作出如下的论断但是别这么快不知什么原因根据彭奇和瓦特曼的公共事业的结果已毫无疑问地断定鉴于波波夫和贝尔契不知什么原因未完成的劳动以及泰斯丢和丘那德的未完成的劳动已经就业已被许多人所否认的论点作出论断认为泰斯丢和丘那德所假设的人认为实际存在的人认为人类总而言之统而言之尽管有进步的营养学和通大便药却在衰弱萎缩衰弱萎缩而且与此同时尤其是不知什么原因尽管体育运动在各方面都有很大进展如网球足球田径车赛游泳飞行划船骑马滑翔溜冰各式各样的网球各种各样致人死命的飞行运动各式各样的秋天夏天冬天冬天网球各种各样的曲棍球盘尼西林和代用品总之我接下去讲与此同时不知什么原因要萎缩要减少尽管有网球我接下去讲飞行滑翔九穴和十八穴的高尔夫球各种各样的网球总之不知什么原因在番克汉贝克汉福尔汉克莱普汉换句话说与此同时尤其是不知什么原因但时间将会揭示要减少减少我接下去讲福尔汉克莱普汉总之自从塞缪尔·约翰逊去世以后到现在每个人的全部损失共计每人一吋四哂只是大概约略粗粗计算到小数点分量很足保持整数赤裸裸的光穿着袜子在康纳马拉总之不知什么原因不管怎样无论如何事实俱在尤其是考虑到更加远为严肃的看来更加严肃的鉴于斯丹威格和彼特曼的徒劳看来更加严肃的鉴于鉴于鉴于斯丹威格和彼特曼徒劳在平原在山地在海洋在烈火沸腾的河里天空是一样的随后是大地换句话说天空随后是大地在一片寒冷一片漆黑中天空大地石头的住所在一片寒冷中哎哟哟在我们的主诞生六百年左右天空大地海洋大地石头的住所汪洋中一片寒冷中在海上在陆地在空中我接下去讲不知什么原因尽管有网球事实俱在但时间将会揭示我接下去讲哎哟哟总之一句话石头的住所谁能怀疑我接下去讲但是别这么快我接下去讲头颅要萎缩衰弱减少与此同时尤其是不知什么原因尽管有网球胡子火焰球队石头那么蓝那么平静哎哟哟头颅头颅头颅头颅在康纳马拉尽管有网球未完成的徒然的劳动更加严肃的石头的住所总之我接下去讲哎哟哟徒劳的未完成的头颅头颅在康纳马拉尽管有网球头颅哎哟石头丘那德（混战，最后的狂喊）网球……石头……那么平静……丘那德……未完成的……

——萨缪尔·贝克特《等待戈多》

*https://github.com/stone-zeng/zhlipsum.

第 1 节　简介

zhlipsum 宏包用于输入中文乱数假文（拉丁语为 *Lorem ipsum*）。乱数假文是大段无意义的文字，常用来测试排版效果。支持其他语言乱数假文的宏包还有 lipsum、kantlipsum、blindtext 等。

zhlipsum 宏包支持 UTF-8、GBK 和 Big5 编码，依赖 LaTeX3 项目中的 expl3、xparse 和 l3keys2e 宏包。为正确输入中文，zhlipsum 需要与 CJK 宏包或 CTEX 宏集等配套使用。

第 2 节　使用说明

encoding

encoding = ⟨**utf8**|*gbk*|*big5*⟩

New: 2017-09-16
Updated: 2018-04-01

用于指定编码的宏包选项，可在调用宏包的时候设定。默认为 utf8。对于 XeLaTeX、LuaLaTeX 和 upLaTeX 等 Unicode 引擎，gbk 和 big5 编码无效，宏包将强制使用 utf8 编码。

如果使用了 CTEX 宏集，则编码会根据 CTEX 自动确定。但需注意，在 CTEX 宏集中，相应的宏包选项为大写的 UTF8 和 GBK，而本宏包中所有选项均为小写。

\zhlipsum

\zhlipsum[⟨段落⟩][⟨选项⟩]
\zhlipsum*[⟨段落⟩][⟨选项⟩]

Updated: 2020-04-08

插入假文文本。参数 ⟨段落⟩ 和 ⟨选项⟩ 都是可选的。注意各参数之间不可以有空格。

默认情况下，不带星号的命令 \zhlipsum 会在假文段落之后与之间进行分段（即插入 \par），而带星号的命令 \zhlipsum* 则不做额外处理。您可以利用后文给出的 before、after、inter 选项来更改默认设置。

第一个可选参数 ⟨段落⟩ 为英文逗号分隔的段落编号列表，举例如下：

例 1

```
% 假设假文最大段落数为 50
\zhlipsum[2-4]          % 可用 a-b 的形式指定
\zhlipsum[4,12,3-8]     % 也可用单个数字指定
\zhlipsum[-10,40-]      % 输出 1-10 段和 40-50 段
\zhlipsum[-]            % 输出全部段落，即 1-50 段
\zhlipsum               % 默认输出 1-3 段
\zhlipsum[48-52]        % 超出部分会自动忽略，即只输出 48-50 段
```

第二个可选参数 ⟨选项⟩ 通过英文逗号分隔的键值列表形式给出。支持的选项见下。

name

name = ⟨假文名称⟩

New: 2018-03-24

选择插入假文的名称。预定义的假文共有 6 种，见表 1。当 encoding=utf8 或 gbk 时，默认使用的假文为 simp；而当 encoding=big5 时，默认则为 trad。

您也可以通过 \newzhlipsum 命令来定义新的假文。

before
after
inter

before = ⟨内容⟩
after = ⟨内容⟩
inter = ⟨内容⟩

New: 2018-03-29

在假文段落之前、之后与之间插入内容。注意使用不带星号的 \zhlipsum 命令时插入的分段命令会被这里的设置覆盖。

表 1　预定义假文

名称	段落数	简体 / 繁体	描述	各编码下的支持情况		
				utf8	gbk	big5
simp	50	简	无意义随机假文	•	•	
trad	50	繁	无意义随机假文	•	•	•
nanshanjing	43	繁	《山海经·南山经》	•		
xiangyu	45	繁	司马迁《史记·项羽本纪》	•	•	•
zhufu	110	简	鲁迅《祝福》	•	•	
aspirin	66	简	维基百科条目：阿司匹林	•	•	

`\newzhlipsum`　`\newzhlipsum{⟨假文名称⟩}{⟨段落列表⟩}`

New: 2018-03-29

声明新的假文。假文名称区分大小写。段落列表以英文逗号分隔，示例如下：

例 2

```
% 注意区分中文逗号与英文逗号
\newzhlipsum{jingyesi}{%
  {床前明月光，},{疑是地上霜。},{举头望明月，},{低头思故乡。}}

\zhlipsum*[-][name=jingyesi]  % 输出全部四句假文，且不分段
```

第 3 节　编程接口

一般而言，第 2 节中列出的命令足够一般用户使用。如需使用编程接口，则可以考虑以下变量和函数。注意使用时需确保开启 LaTeX3 语法。

`\g_zhlipsum_seq`　假文名称列表。

`\zhlipsum_use:nn`　输出多段假文。
#1：假文名称
#2：段落编号列表

`\zhlipsum_if_exist:nTF`　判断是否存在对应名称的假文。
#1：假文名称

`\zhlipsum_new:nn`　声明假文。
#1：假文名称
#2：文本列表

第 4 节　兼容性信息

以下选项在测试版 zhlipsum 宏包中存在，但在 1.0.0 版本之后不建议继续使用。这里仅为兼容性保留。未来将可能删除对它们的支持。

`script`　过时选项。现在相当于 `name`。

第 5 节　已知问题

名称为 nanshanjing 和 xiangyu 的假文文本含有若干生僻字。如需正确显示，可使用 xeCJK 宏包，并设置后备字体为 SimSun-ExtB、Hanazono Mincho（花园明朝）等，具体方法请参考 xeCJK 宏包文档（仅针对编码为 UTF-8，且使用 XeLaTeX 编译的情况）。

GBK 和 Big5 编码在第二字节并没有避开 ASCII 码的范围，因此部分汉字编码的第二字节恰好是 ASCII 编码中的一些特殊字符（如 {、}、\ 等），将导致编译失败。本宏包在这两种编码下的 .def 文件中采取了特殊技巧（见 6.6 小节），请避免修改这些文件。

如无特殊需要，始终建议您采用 UTF-8 编码，并使用 XeLaTeX、LuaLaTeX 等 Unicode 引擎编译。

特殊情况下，如果您必须使用 GBK 或 Big5 编码，并需要声明新的假文，可以采取以下手段临时回避上述问题。

例 3

```
% 文件编码需使用 Big5
% \usepackage[encoding=big5]{zhlipsum}

% 直接使用 \newzhlipsum{big5}{許蓋功，蓋功許，功許蓋} 会报错
% 原理：分组内使用 < > + 代替 { } \ 后，再将原先的 { } \ 设为“其他”类（即
% 类别码为 12）
\begingroup
  \catcode`\<=1
  \catcode`\>=2
  \catcode`\+=0
  \catcode`\{=12
  \catcode`\}=12
  \catcode`\\=12
  +newzhlipsum<big5><許蓋功，蓋功許，功許蓋>
+endgroup
\zhlipsum[name=big5]
```

第 6 节　实现细节

```
1 ⟨*package⟩
2 ⟨@@=zhlipsum⟩
```

检查 LaTeX3 编程环境。

```
3 \RequirePackage { xparse, l3keys2e }
4 \msg_new:nnn { zhlipsum } { l3-too-old }
5   {
6     Package~ "#1"~ is~ too~ old. \\\\
7     Please~ update~ an~ up-to-date~ version~ of~ the~ bundles \\
8     "l3kernel"~ and~ "l3packages"~ using~ your~ TeX~ package \\
9     manager~ or~ from~ CTAN.
10  }
11 \clist_map_inline:nn { expl3, xparse, l3keys2e }
12  {
13    \@ifpackagelater {#1} { 2018/05/12 }
14      { } { \msg_error:nnn { zhlipsum } { l3-too-old } {#1} }
15  }
```

6.1　内部变量和函数

\l__zhlipsum_tmpa_tl
\l__zhlipsum_tmpa_seq
\l__zhlipsum_tmpb_seq
\l__zhlipsum_tmpa_str

临时变量。

```
16 \tl_new:N  \l__zhlipsum_tmpa_tl
```

```
\seq_new:N \l__zhlipsum_tmpa_seq
\seq_new:N \l__zhlipsum_tmpb_seq
\str_new:N \l__zhlipsum_tmpa_str
```

\g_zhlipsum_seq　假文名称列表。

```
\seq_new:N \g_zhlipsum_seq
```

\c_zhlipsum_simp_seq
\c_zhlipsum_trad_seq　预定义的简体中文与繁体中文的假文名称列表。

```
\seq_const_from_clist:Nn \c_zhlipsum_simp_seq { simp, zhufu, aspirin }
\seq_const_from_clist:Nn \c_zhlipsum_trad_seq { trad, xiangyu, nanshanjing }
```

\file_input:x　LaTeX3 函数变体。

```
\cs_generate_variant:Nn \file_input:n { x }
```

__zhlipsum_if_unicode_engine:*TF*　判断是否为 Unicode 引擎。来自于 zhnumber 宏包。

```
\prg_new_conditional:Npnn \__zhlipsum_if_unicode_engine: { T, F, TF }
  {
    \bool_lazy_any:nTF
      {
        \sys_if_engine_xetex_p:
        \sys_if_engine_luatex_p:
        \sys_if_engine_uptex_p:
      }
      { \prg_return_true: } { \prg_return_false: }
  }
```

__zhlipsum_if_encoding:n*TF*
\g__zhlipsum_encoding_str　判断当前编码。

```
\prg_new_conditional:Npnn \__zhlipsum_if_encoding:n #1 { T, F, TF }
  {
    \str_if_eq:VnTF \g__zhlipsum_encoding_str {#1}
      { \prg_return_true: } { \prg_return_false: }
  }
\prg_generate_conditional_variant:Nnn \__zhlipsum_if_encoding:n { V } { T, F, TF }
\str_new:N \g__zhlipsum_encoding_str
```

__zhlipsum_msg_new:nn
__zhlipsum_error:n
__zhlipsum_error:nn
__zhlipsum_warning:n
__zhlipsum_warning:nn
__zhlipsum_warning:nnn
__zhlipsum_warning:nxxx
__zhlipsum_info:nn　各种信息函数的缩略形式。

```
\cs_new:Npn \__zhlipsum_msg_new:nn   { \msg_new:nnn       { zhlipsum } }
\cs_new:Npn \__zhlipsum_error:n      { \msg_error:nn      { zhlipsum } }
\cs_new:Npn \__zhlipsum_error:nn     { \msg_error:nnn     { zhlipsum } }
\cs_new:Npn \__zhlipsum_warning:n    { \msg_warning:nn    { zhlipsum } }
\cs_new:Npn \__zhlipsum_warning:nn   { \msg_warning:nnn   { zhlipsum } }
\cs_new:Npn \__zhlipsum_warning:nnn  { \msg_warning:nnnn  { zhlipsum } }
\cs_new:Npn \__zhlipsum_warning:nxxx { \msg_warning:nnxxx { zhlipsum } }
\cs_new:Npn \__zhlipsum_info:nn      { \msg_info:nnn      { zhlipsum } }
```

6.2　宏包选项

encoding　设置编码。

```
\keys_define:nn { zhlipsum / option }
  {
    encoding .choices:nn =
      { utf8, gbk, big5 }
      { \str_gset:Nn \g__zhlipsum_encoding_str {#1} },
    encoding / unknown .code:n =
      { \__zhlipsum_error:nn { invalid-encoding } {#1} },
    encoding .value_required:n = true,
```

处理未知选项。

```
    unknown .code:n = { \__zhlipsum_error:n { unknown-option } }
  }
```

提示信息。

```
\__zhlipsum_msg_new:nn { invalid-encoding }
  {
    Encoding~"#1"~is~invalid. \\
    Available~encodings~are~"utf8",~"gbk"~and~"big5".
  }
\__zhlipsum_msg_new:nn { unknown-option }
  { Package~option~'\l_keys_key_tl'~is~unknown. }
```

__zhlipsum_check_unicode_engine_encoding: Unicode 引擎下编码始终设为 UTF-8。

```
\cs_new_protected:Npn \__zhlipsum_check_unicode_engine_encoding:
  {
    \__zhlipsum_if_unicode_engine:T
      {
        \str_if_empty:NF \g__zhlipsum_encoding_str
          {
            \__zhlipsum_if_encoding:nF { utf8 }
              { \__zhlipsum_warning:n { unicode-engine } }
          }
        \str_gset:Nn \g__zhlipsum_encoding_str { utf8 }
      }
  }
\__zhlipsum_msg_new:nn { unicode-engine }
  {
    You~are~now~using~Unicode~engine~\c_sys_engine_str\c_space_tl~so~
    encoding~"\g__zhlipsum_encoding_str"~is~invalid. \\
    Changed~into~"utf8".
  }
```

__zhlipsum_check_ctex_encoding: 如果调用了 CTEX 宏集, 则自动确定编码。

```
\cs_new_protected:Npn \__zhlipsum_check_ctex_encoding:
  {
    \tl_if_exist:NT \l__ctex_encoding_tl
      {
        \str_set:Nx \l__zhlipsum_tmpa_str
          { \str_lower_case:f { \l__ctex_encoding_tl } }
        \str_if_empty:NF \g__zhlipsum_encoding_str
          {
            \__zhlipsum_if_encoding:VF \l__zhlipsum_tmpa_str
              { \__zhlipsum_warning:n { ctex-encoding-conflict } }
          }
        \str_gset_eq:NN \g__zhlipsum_encoding_str \l__zhlipsum_tmpa_str
      }
  }
\__zhlipsum_msg_new:nn { ctex-encoding-conflict }
  {
    Package~option~"encoding=\g__zhlipsum_encoding_str"~is~in~conflict~with~
    ctex's~option~"\l__ctex_encoding_tl". \\
    Changed~into~"encoding=\l__zhlipsum_tmpa_str".
  }
```

将宏包选项传给 zhlipsum/option。

```
\ProcessKeysOptions { zhlipsum / option }
```

检查编码兼容性。

```
\__zhlipsum_check_unicode_engine_encoding:
\__zhlipsum_check_ctex_encoding:
\str_if_empty:NT \g__zhlipsum_encoding_str
  { \str_gset:Nn \g__zhlipsum_encoding_str { utf8 } }
```

6.3　函数选项

\l__zhlipsum_name_tl　保存假文名称。

```
\tl_new:N \l__zhlipsum_name_tl
```

\l__zhlipsum_before_tl
\l__zhlipsum_after_tl
\l__zhlipsum_inter_tl　保存假文之前、之后与之间插入的内容。

```
\tl_new:N \l__zhlipsum_before_tl
\tl_new:N \l__zhlipsum_after_tl
\tl_new:N \l__zhlipsum_inter_tl

\keys_define:nn { zhlipsum }
  {
```

name　假文名称。Big5 编码不支持简体中文。

```
    name .code:n =
      {
        \tl_set:Nn \l__zhlipsum_name_tl {#1}
        \__zhlipsum_if_encoding:nT { big5 }
          {
            \seq_if_in:NVT \c_zhlipsum_simp_seq \l__zhlipsum_name_tl
              {
                \__zhlipsum_warning:nn { big5-require-trad } {#1}
                \tl_set:Nn \l__zhlipsum_name_tl { trad }
              }
          }
      },
```

script　选择输入简体中文或繁体中文。过时选项。

```
    script .code:n =
      {
        \__zhlipsum_warning:nn { deprecated-option }
          { Option~ "name=#1"~ will~ be~ set. }
        \keys_set:nn { zhlipsum } { name = #1 }
      },
```

before
after
inter　假文之前、之后与之间插入的内容。

```
    before .tl_set:N = \l__zhlipsum_before_tl,
    after  .tl_set:N = \l__zhlipsum_after_tl,
    inter  .tl_set:N = \l__zhlipsum_inter_tl
  }
```

提示信息。

```
\__zhlipsum_msg_new:nn { big5-require-trad }
  {
    Name~ "#1"~ is~ not~ available~ in~ "Big5"~ encoding. \\
    Changed~ into~ "trad".
  }
\__zhlipsum_msg_new:nn { deprecated-option }
  { Option~ "\l_keys_key_tl"~ is~ deprecated. \\ #1 }
```

初始选项设置。

```
\__zhlipsum_if_encoding:nTF { big5 }
  { \keys_set:nn { zhlipsum } { name = trad } }
  { \keys_set:nn { zhlipsum } { name = simp } }
```

6.4　输出假文

\zhlipsum
__zhlipsum:n

输出假文，第一个可选参数表示段落数，默认为 1-3；第二个可选参数为选项列表。

```
\NewDocumentCommand \zhlipsum { s o +o }
  {
    \group_begin:
      \IfBooleanF {#1}
        {
          \tl_set:Nn \l__zhlipsum_before_tl { }
          \tl_set:Nn \l__zhlipsum_after_tl  { \par }
          \tl_set:Nn \l__zhlipsum_inter_tl  { \par }
        }
      \IfValueTF {#3}
        {
          \keys_set:nn { zhlipsum } {#3}
          \__zhlipsum:n {#2}
        }
        {
          \IfValueTF {#2}
            {
```

如果只带一个参数，那么根据其是否含有 = 来判断该参数是段落数还是选项列表。

```
              \__zhlipsum_if_key_value_list:nTF {#2}
                {
                  \keys_set:nn { zhlipsum } {#2}
                  \__zhlipsum:n { 1 - 3 }
                }
                { \__zhlipsum:n {#2} }
            }
            { \__zhlipsum:n { 1 - 3 } }
        }
    \group_end:
  }
\cs_new_protected:Npn \__zhlipsum:n #1
  { \exp_args:No \zhlipsum_use:nn { \l__zhlipsum_name_tl } {#1} }
```

__zhlipsum_if_key_value_list:nTF

判断是否为键值列表，即是否含有 =。

```
\cs_new_protected:Npn \__zhlipsum_if_key_value_list:nTF #1
  { \tl_if_in:nnTF {#1} {=} }
```

\l__zhlipsum_par_num_seq

保存段落编号。

```
\seq_new:N \l__zhlipsum_par_num_seq
```

\zhlipsum_use:nn

输出多段假文。#1 = 假文名称，#2 = 段落编号列表。解析段落编号之后，按次序逐项输出，并在前后插入相应内容。注意最后一段需要单独处理。

```
\cs_new_protected:Npn \zhlipsum_use:nn #1#2
  {
    \__zhlipsum_if_cjk_valid_encoding:TF
      {
        \zhlipsum_if_exist:nTF {#1}
          {
            \__zhlipsum_parse_par:nn {#1} {#2}
            \seq_if_empty:NF \l__zhlipsum_par_num_seq
              {
                \seq_pop_right:NN  \l__zhlipsum_par_num_seq \l__zhlipsum_tmpa_tl
                \l__zhlipsum_before_tl
                \seq_map_inline:Nn \l__zhlipsum_par_num_seq
                  {
                    \__zhlipsum_use:nn {#1} {##1}
                    \l__zhlipsum_inter_tl
                  }
```

```
                \__zhlipsum_use:nn {#1} { \l__zhlipsum_tmpa_tl }
                \l__zhlipsum_after_tl
              }
          }
          { \__zhlipsum_error:nn { invalid-name } {#1} }
      }
      { \__zhlipsum_error:n { CJK-invalid-encoding } }
  }
\__zhlipsum_msg_new:nn { invalid-name }
  {
    Name~ "#1"~ is~ unknown. \\
    Please~ use~ the~ pre-defined~ Chinese~ dummy~ texts~ or~
    declare~ new~ one.
  }
\__zhlipsum_msg_new:nn { CJK-invalid-encoding }
  {
    The~ current~ CJK~ environment~ uses~ "\CJK@@@enc"~ encoding,~
    but~ zhlipsum~ package~ has~ been~ loaded~ with~ the~ option~
    "encoding=\g__zhlipsum_encoding_str". \\
    Please~ check~ the~ package~ options.
  }
```

__zhlipsum_if_cjk_valid_encoding:TF　检查 CJK 环境编码。

```
\prg_new_protected_conditional:Npnn \__zhlipsum_if_cjk_valid_encoding: { TF }
  {
    \tl_if_exist:NTF \CJK@@@enc
      {
        \exp_args:NV \str_case:nn \g__zhlipsum_encoding_str
          {
            { utf8 } { \str_if_eq:VnTF \CJK@@@enc { UTF8 } }
            { gbk  } { \str_if_in:NnTF \CJK@@@enc { GB   } }
            { big5 } { \str_if_eq:VnTF \CJK@@@enc { Bg5  } }
          }
        { \prg_return_true: } { \prg_return_false: }
      }
      { \prg_return_true: }
  }
```

\zhlipsum_if_exist:nTF　判断是否存在对应名称的假文。

```
\prg_new_protected_conditional:Npnn \zhlipsum_if_exist:n #1 { T, F, TF }
  {
    \seq_if_in:NnTF \g_zhlipsum_seq {#1}
      { \prg_return_true: } { \prg_return_false: }
  }
```

\l__zhlipsum_begin_int
\l__zhlipsum_end_int
\l__zhlipsum_max_int

```
\int_new:N \l__zhlipsum_begin_int
\int_new:N \l__zhlipsum_end_int
\int_new:N \l__zhlipsum_max_int
```

\l__zhlipsum_modified_range_bool
\l__zhlipsum_invalid_range_bool

```
\bool_new:N \l__zhlipsum_modified_range_bool
\bool_new:N \l__zhlipsum_invalid_range_bool
```

__zhlipsum_parse_par:nn　解析段落编号列表。#1 = 假文名称, #2 = 段落编号列表。

编号列表用逗号分隔, 其中的每一项为单个数字或为 a-b 的形式。若 a、b 为空, 则分别取为 1 或允许的最大值(即段落数)。超过范围的数字则忽略。

```
\cs_new_protected:Npn \__zhlipsum_parse_par:nn #1#2
  {
    \seq_clear:N \l__zhlipsum_par_num_seq
    \int_set_eq:Nc \l__zhlipsum_max_int { g__zhlipsum_ #1 _int }
    \clist_map_inline:nn {#2}
      {
```

```
        \__zhlipsum_parse_par_aux:n {##1}
        \bool_if:NTF \l__zhlipsum_invalid_range_bool
          { \__zhlipsum_warning:nnn { invalid-range } {##1} {#2} }
          {
            \bool_if:NT \l__zhlipsum_modified_range_bool
              {
                \__zhlipsum_warning:nxxx { modified-range }
                  {##1} {#2} { \__zhlipsum_par_range: }
              }
            \seq_concat:NNN \l__zhlipsum_par_num_seq
              \l__zhlipsum_par_num_seq \l__zhlipsum_tmpa_seq
          }
      }
  }
```

__zhlipsum_parse_par_aux:n

```
\cs_new_protected:Npn \__zhlipsum_parse_par_aux:n #1
  {
    \bool_set_false:N \l__zhlipsum_modified_range_bool
    \bool_set_false:N \l__zhlipsum_invalid_range_bool
    \seq_clear:N \l__zhlipsum_tmpa_seq
    \tl_if_in:nnTF {#1} { - }
      {
        \seq_set_split:Nnn \l__zhlipsum_tmpb_seq { - } {#1}
```

“-”左侧的数字。

```
        \seq_pop_left:NN \l__zhlipsum_tmpb_seq \l__zhlipsum_tmpa_tl
        \tl_if_empty:NTF \l__zhlipsum_tmpa_tl
          { \int_set_eq:NN \l__zhlipsum_begin_int \c_one_int }
          {
            \int_set:Nn \l__zhlipsum_begin_int { \l__zhlipsum_tmpa_tl }
            \int_compare:nNnT \l__zhlipsum_begin_int < \c_one_int
              {
                \int_set_eq:NN \l__zhlipsum_begin_int \c_one_int
                \bool_set_true:N \l__zhlipsum_modified_range_bool
              }
          }
```

“-”右侧的数字。注意左右数字均由 `\seq_pop_left:NN` 得到，因此 `-3-4` 实际相当于 `-3`，进而被解析为 `1-3`。

```
        \seq_pop_left:NN \l__zhlipsum_tmpb_seq \l__zhlipsum_tmpa_tl
        \tl_if_empty:NTF \l__zhlipsum_tmpa_tl
          { \int_set_eq:NN \l__zhlipsum_end_int \l__zhlipsum_max_int }
          {
            \int_set:Nn \l__zhlipsum_end_int { \l__zhlipsum_tmpa_tl }
            \int_compare:nNnT \l__zhlipsum_end_int > \l__zhlipsum_max_int
              {
                \int_set_eq:NN \l__zhlipsum_end_int \l__zhlipsum_max_int
                \bool_set_true:N \l__zhlipsum_modified_range_bool
              }
          }
```

检查取值范围。

```
        \bool_lazy_or:nnTF
          { \int_compare_p:nNn \l__zhlipsum_begin_int > \l__zhlipsum_max_int }
          { \int_compare_p:nNn \l__zhlipsum_begin_int > \l__zhlipsum_end_int }
          { \bool_set_true:N \l__zhlipsum_invalid_range_bool }
          {
            \int_step_inline:nnn
              { \l__zhlipsum_begin_int } { \l__zhlipsum_end_int }
              { \seq_put_right:Nn \l__zhlipsum_tmpa_seq {##1} }
          }
      }
      {
```

单个数字的处理。

```
      \bool_lazy_or:nnTF
        { \int_compare_p:nNn {#1} > \l__zhlipsum_max_int }
        { \int_compare_p:nNn {#1} < \c_one_int }
        { \bool_set_true:N \l__zhlipsum_invalid_range_bool }
        { \seq_put_right:Nn \l__zhlipsum_tmpa_seq {#1} }
    }
}
```

__zhlipsum_par_range:　显示段落范围(用在提示信息中, 可以完全展开)。

```
\cs_new:Npn \__zhlipsum_par_range:
  {
    \int_compare:nNnTF \l__zhlipsum_begin_int = \l__zhlipsum_end_int
      { \int_use:N \l__zhlipsum_begin_int }
      { \int_use:N \l__zhlipsum_begin_int - \int_use:N \l__zhlipsum_end_int }
  }
```

提示信息。

```
\__zhlipsum_msg_new:nn { modified-range }
  {
    Your~ required~ range~ "#1"~ in~ "#2"~ will~ be~ modified. \\
    Changed~ into~ "#3".
  }
\__zhlipsum_msg_new:nn { invalid-range }
  {
    Your~ required~ range~ "#1"~ in~ "#2"~ is~ invalid. \\
    Nothing~ will~ be~ output.
  }
```

__zhlipsum_use:nn　输出一段假文。#1 = 假文名称, #2 = 段落编号。

```
\cs_new_protected:Npn \__zhlipsum_use:nn #1#2
  { \tl_use:c { c__zhlipsum_ #1 @ #2 _tl } }
```

6.5　声明假文

\newzhlipsum
\zhlipsum_new:nn　声明假文。#1 = 假文名称, #2 = 文本 clist。

```
\NewDocumentCommand \newzhlipsum { m m }
  { \zhlipsum_new:nn {#1} {#2} }
\cs_new_protected:Npn \zhlipsum_new:nn #1#2
  {
    \zhlipsum_if_exist:nTF {#1}
      { \__zhlipsum_error:nn { already-defined } {#1} }
      {
        \seq_gput_left:Nn \g_zhlipsum_seq {#1}
        \int_new:c { g__zhlipsum_ #1 _int }
        \clist_map_inline:nn {#2} { \__zhlipsum_new:nn {#1} {##1} }
        \__zhlipsum_info:nn { defining-text } {#1}
      }
  }
\__zhlipsum_msg_new:nn { already-defined }
  {
    Chinese~ dummy~ text~ "#1"~ has~ been~ used.~
    Please~ use~ another~ name.
  }
\__zhlipsum_msg_new:nn { defining-text }
  {
    Chinese~ dummy~ text~ "#1"~ is~ created.~
    It~ has~ \int_use:c { g__zhlipsum_ #1 _int }~ paragraphs.
  }
```

__zhlipsum_new:nn　定义新的假文段落。#1 = 假文名称, #2 = 文本。

```
\cs_new_protected:Npn \__zhlipsum_new:nn #1#2
  {
    \int_gincr:c { g__zhlipsum_ #1 _int }
    \tl_const:cn
      { c__zhlipsum_ #1 @ \int_use:c { g__zhlipsum_ #1 _int } _tl } {#2}
  }
```

根据编码读入假文文本定义文件。

```
\file_input:x { zhlipsum- \g__zhlipsum_encoding_str .def }
⟨/package⟩
```

6.6　假文文本

__zhlipsum_set_special_catcode:　在声明预定义文本时, 为了兼容 CJK 宏包的特殊处理, 需要临时更改类别码。具体来说, 在 GBK/Big5 编码下, 由于汉字的第二个字节会与 TEX 中的特殊符号 \、{、}、~ 冲突, 所以需要将它们的类别码改为 12(其他), 并分别用 +、<、> 和 ∗ 代替。星号 ∗ 在开启 LATEX3 语法后实际相当于空格(类别码为 10)。

```
⟨*text⟩
\cs_new_protected:Npn \__zhlipsum_set_special_catcode:
  {
⟨!utf8⟩   \__zhlipsum_active_first_byte:
    \char_set_catcode_escape:N      \+
    \char_set_catcode_group_begin:N \<
    \char_set_catcode_group_end:N   \>
    \char_set_catcode_space:N       \*
    \char_set_catcode_other:N \\
    \char_set_catcode_other:N \{
    \char_set_catcode_other:N \}
    \char_set_catcode_other:N \~
  }
```

__zhlipsum_active_first_byte:　将汉字的首字节设为活动字符(类别码 12)。UTF-8 编码下不需要该操作。

```
⟨*!utf8⟩
\cs_new_protected:Npx \__zhlipsum_active_first_byte:
  {
    \int_step_function:nnN { "81 } { "FE }
      \char_set_catcode_active:n
  }
⟨/!utf8⟩
⟨/text⟩
```

预定义假文的声明放置在分组内, 利用 __zhlipsum_set_special_catcode: 切换类别码后可以不再需要 CJK 的预处理操作。具体声明此处不再列出。

版本历史

代码索引

意大利体的数字表示描述对应索引项的页码; 带下划线的数字表示定义对应索引项的代码行号; 罗马字体的数字表示使用对应索引项的代码行号。

Z