zhnumber 宏包

李清
sobenlee@gmail.com

2014/09/09　　v2.0*

1　简介

zhnumber 宏包用于将阿拉伯数字按照中文格式输出。相比于 CJKnumb，它提供的三个格式转换命令 \zhnumber，\zhdigits 和 \zhnum 都是可以适当展开的，可以正常使用于 PDF 书签和交叉引用。

zhnumber 支持 GBK，Big5 和 UTF8 编码，依赖 LaTeX3 项目的 expl3，xparse 和 l3keys2e 宏包。

2　使用方法

encoding

Updated: 2014-09-09

encoding = ⟨*GBK*|*Big5*|*UTF8*⟩

用于指定编码的宏包选项，可以在调用宏包的时候设定，也可以用 \zhnumsetup 在导言区内设定。对于 upLaTeX、XeLaTeX 和 LuaLaTeX，不用指定编码，宏包将自动使用 UTF8 编码。只有 LaTeX 和 pdfLaTeX 需要指定编码，如果没有指定，默认将使用 GBK。

\zhnumber

Updated: 2014-09-09

\zhnumber {⟨*number*⟩}

以中文格式输出数字。这里的数字可以是整数、小数和分数。例如

二十亿零一千二百零二万零一百二十
二十亿零一千二百零二万零一百二十
二十亿零一千二百零二万零一百二十
二千零一十二点零二零一二零
二千零一十二点零
零点二零一二
二万零一百二十分之二万零一百二十
二千零一十二分之零
零分之二千零一十二
二百零一又一百二十分之二千零二十

```
\zhnumber{2012020120}\\
\zhnumber{2 012 020 120}\\
\zhnumber{2,012,020,120}\\
\zhnumber{2012.020120}\\
\zhnumber{2012.}\\
\zhnumber{.2012}\\
\zhnumber{20120/20120}\\
\zhnumber{/2012}\\
\zhnumber{2012/}\\
\zhnumber{201;2020/120}
```

\zhdigits

Updated: 2014-09-09

\zhdigits　{⟨*number*⟩}
\zhdigits * {⟨*number*⟩}

将阿拉伯数字转换为中文数字串。缺省状态下，\zhdigits 将 0 映射为〇，如果需要将其映射为零，可以使用带星号的形式。例如

二〇一二〇二〇一二〇
二零一二零二零一二零

```
\zhdigits{2012020120}\\
\zhdigits*{2012020120}
```

\zhnum

Updated: 2014-09-09

\zhnum {⟨*counter*⟩}

与 \roman 等类似，用于将 LaTeX 计数器的值转换为中文数字。例如

二

```
\zhnum{section}
```

*ctex-kit rev715.

\zhweekday　\zhweekday {⟨*yyyy/mm/dd*⟩}

New: 2012-5-25

输出日期当天的星期。例如

星期日

```
\zhweekday{2012/5/20}
```

\zhdate　\zhdate {⟨*yyyy/mm/dd*⟩}
\zhdate * {⟨*yyyy/mm/dd*⟩}

New: 2012-5-25

以中文格式输出日期,其中带 * 的命令还输出星期。例如

2012 年 5 月 21 日
2012 年 5 月 21 日星期一

```
\zhdate{2012/5/21}\\
\zhdate*{2012/5/21}
```

\zhtoday

New: 2012-5-25

与 \today 类似,以中文输出当天的日期。例如

2014 年 9 月 9 日

```
\zhtoday
```

\zhtime　\zhtime {⟨*hh:mm*⟩}

New: 2012-5-25

以中文格式输出时间。例如

23 时 56 分

```
\zhtime{23:56}
```

\zhcurrtime

New: 2012-5-25

输出当前的时间。例如

22 时 12 分

```
\zhcurrtime
```

\zhnumExtendScaleMap　\zhnumExtendScaleMap [⟨*character*⟩] {⟨*character*$_1$⟩, ⟨*character*$_2$⟩, ..., ⟨*character*$_n$⟩}

New: 2012-5-25

缺省状态下 \zhnumber 能正确中文格式化的最大整数是 $10^{48}-1$,\zhdigits 不受这个大小的限制。可以通过 \zhnumExtendScaleMap 来扩展 \zhnumber。⟨*character*$_i$⟩ 设置 $10^{4(i+11)}$。若给出可选项 ⟨*character*⟩,则当数字大于 $10^{4(n+12)}-1$ 时,统一用 ⟨*character*⟩ 设置输出数字的进位。

\zhnumsetup　\zhnumsetup {⟨*key*$_1$⟩=⟨*val*$_1$⟩, ⟨*key*$_2$⟩=⟨*val*$_2$⟩, ...}

用于在导言区或文档中,设置中文数字的输出格式。目前可以设置的 ⟨*key*⟩ 如下介绍。

time　time = {⟨*Arabic*⟩|⟨*Chinese*⟩}

New: 2012-5-25

设置日期和时间的数字格式,⟨*Arabic*⟩ 为阿拉伯数字,而 ⟨*Chinese*⟩ 为中文数字。默认使用阿拉伯数字。例如

二〇一四年九月九日二十二时十二分

```
\zhnumsetup{time=Chinese}
\zhtoday\zhcurrtime
```

style　style = {⟨*Simplified*⟩|⟨*Traditional*⟩|⟨*Normal*⟩|⟨*Financial*⟩|⟨*Ancient*⟩}

Updated: 2012-5-25

意义分别为

Simplified　以简体中文输出数字(对 Big5 编码无效);
Traditional　以繁体中文输出数字(对 Big5 编码无效);
Normal　以小写形式输出中文数字;
Financial　以大写形式输出中文数字;
Ancient　以廿输出 20,以卅输出 30,以卌输出 40,以皕输出 200。

可以设置 style 为其中一个,也可以是前三个与后两个的适当组合,默认是简体小写。例如

陆萬貳仟零壹拾貳點叁
廿一

```
\zhnumsetup{style={Traditional,Financial}}
\zhnumber{62012.3}\\
\zhnumsetup{style=Ancient}
\zhnumber{21}
```

null　null = ⟨*true*|***false***⟩

缺省状态下，除了 \zhdigits 外，其它的格式转换命令，将 0 映射成零，如果需要将 0 映射成〇，可以使用这个选项。

zhnumber 提供下列选项来控制阿拉伯数字的中文映射。

```
- -0 0 1 2 3 4 5 6 7 8 9 10 20 30 40 100 200 1000
E2 E3 E4 E8 E12 E16 E20 E24 E28 E32 E36 E40 E44
F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 F10 F100 F1000 FE2 FE3
dot and parts
year month day hour minute weekday mon tue wed thu fri sat sun
```

其中 - 设置负，-0 设置〇，dot 设置小数的点，and 和 parts 分别设置分数的“又”和“分之”，E*n* 设置 10^n，而 F*n* 设置数字 *n* 的大写。其它的选项同字面意思，不再赘述。例如

```
\zhnumsetup{2={两}}
```

可以将 2 映射成两。需要说明的是，zhnumber 将优先使用这里的设置，所以可能会影响到 style 选项。如果要恢复 style 的功能，可以使用 reset 选项。

reset　reset

Updated: 2014-09-09

用于恢复 zhnumber 对阿拉伯数字的初始化映射。zhnumber 的中文数字初始化设置见源代码（第 4 节）。

activechar　activechar = ⟨***true***|*false*⟩

New: 2014-09-09

在 LaTeX 或者 pdfLaTeX 下面输出汉字，传统的办法需要将汉字的首字节设置为活动字符，然后再通过特殊的宏技巧来实现。因此，zhnumber 在载入配置文件的时候，默认会将汉字的首字节设置为活动字符。禁用本选项将不会改变汉字首字节的类代码。需要在本选项之后，使用 encoding 或者 reset 选项才会有效果。

\zhnumber　\zhnumber　[⟨*options*⟩] {⟨*number*⟩}
\zhdigits　\zhdigits * [⟨*options*⟩] {⟨*number*⟩}
\zhnum　\zhnum　[⟨*options*⟩] {⟨*counter*⟩}

Updated: 2014-09-09

如果只改变当前数字的中文输出格式，可以使用带选项的格式转换命令，其中 ⟨*options*⟩ 与 \zhnumsetup 的参数相同，如上所介绍。这些带了选项的命令是不可展开的，在某些场合使用时要小心。

3　zhnumber 宏包代码实现

```
⟨*package⟩
⟨@@=zhnum⟩
\msg_new:nnn { zhnumber } { l3-too-old }
  {
    Support~package~'expl3'~too~old. \\\\
    Please~update~an~up~to~date~version~of~the~bundles\\\\
    'l3kernel'~and~'l3packages'\\\\
    using~your~TeX~package~manager~or~from~CTAN.
  }
\@ifpackagelater { expl3 } { 2014/08/25 } { }
  { \msg_error:nn  { zhnumber }  { l3-too-old } }
\RequirePackage { xparse , l3keys2e }
```

\zhnumber　用于将输入的数字按照中文格式输出。

```
\DeclareExpandableDocumentCommand \zhnumber { +o +m }
  {
    \IfNoValueTF {#1}
      { \zhnum_number:n {#2} }
      { \zhnumberwithoptions {#1} {#2} }
  }
```

(*End definition for* \zhnumber. *This function is documented on page* 3.)

\zhnumberwithoptions 带选项的用户函数。

```
\NewDocumentCommand \zhnumberwithoptions { +m +m }
 {
   \group_begin:
     \keys_set:nn { zhnum / options } {#1}
     \zhnum_number:n {#2}
   \group_end:
 }
```

(*End definition for* `\zhnumberwithoptions`. *This function is documented on page* **??**.)

\zhnum_number:n 先判断输入的是小数还是分数。

__zhnum_number:www

```
\cs_new:Npn \zhnum_number:n #1
  { \__zhnum_number:www #1 . \q_nil . \q_stop }
\cs_new:Npn \__zhnum_number:www #1 . #2 . #3 \q_stop
  {
    \quark_if_nil:nTF {#2}
      { \__zhnum_integer_or_fraction:www #1 / \q_nil / \q_stop }
      { \zhnum_decimal:nn {#1} {#2} }
  }
```

(*End definition for* `\zhnum_number:n`.)

__zhnum_integer_or_fraction:www 判断是否输入的是分数。

```
\cs_new:Npn \__zhnum_integer_or_fraction:www #1 / #2 / #3 \q_stop
  {
    \quark_if_nil:nTF {#2}
      { \zhnum_integer:n {#1} }
      { \__zhnum_fraction:wwww #2 \q_mark #1 ; \q_nil ; \q_stop }
  }
```

(*End definition for* `\__zhnum_integer_or_fraction:www`.)

__zhnum_fraction:wwww 对分数进行预处理。

```
\cs_new:Npn \__zhnum_fraction:wwww #1 \q_mark #2 ; #3 ; #4 \q_stop
  {
    \quark_if_nil:nTF {#3}
      {
        \zhnum_blank_to_zero:f {#1}
        \c__zhnum_parts_tl
        \zhnum_blank_to_zero:f {#2}
      }
      {
        \tl_if_blank:fF {#2}
          {
            \zhnum_number:n {#2}
            \c__zhnum_and_tl
          }
        \zhnum_blank_to_zero:f {#1}
        \c__zhnum_parts_tl
        \zhnum_blank_to_zero:f {#3}
      }
  }
\cs_generate_variant:Nn \tl_if_blank:nF { f }
```

(*End definition for* `\__zhnum_fraction:wwww`.)

\zhnum_decimal:nn 对小数进行预处理。

```
\cs_new:Npn \zhnum_decimal:nn #1#2
  {
    \zhnum_blank_to_zero:f {#1} \c__zhnum_dot_tl
    \tl_if_blank:fTF {#2}
      { \c__zhnum_zero_tl }
      { \zhnum_digits_zero:n {#2} }
  }
\cs_generate_variant:Nn \tl_if_blank:nTF { f }
```

(*End definition for* `\zhnum_decimal:nn`.)

\zhnum_blank_to_zero:n 输出小数的整数位。

```
\cs_new:Npn \zhnum_blank_to_zero:n #1
  {
    \tl_if_blank:nTF {#1}
      { \c__zhnum_zero_tl }
      { \zhnum_number:n {#1} }
  }
\cs_generate_variant:Nn \zhnum_blank_to_zero:n { f }
```

(*End definition for* `\zhnum_blank_to_zero:n`.)

\zhnum 用于将 LaTeX 计数器按中文格式输出。

\zhnumberwithoptions

```
\DeclareExpandableDocumentCommand \zhnum { o m }
  {
    \IfNoValueTF {#1}
      { \zhnum_counter:n {#2} }
      { \zhnumwithoptions {#1} {#2} }
  }
\NewDocumentCommand \zhnumwithoptions { m m }
  {
    \group_begin:
      \keys_set:nn { zhnum / options } {#1}
      \zhnum_counter:n {#2}
    \group_end:
  }
```

(*End definition for* `\zhnum` *and* `\zhnumberwithoptions`. *These functions are documented on page* 3.)

\zhnum_counter:n 可以直接通过比较 LaTeX 计数器的值来得到符号和绝对值。

```
\cs_new_nopar:Npn \zhnum_counter:n #1
  { \exp_args:Nc \zhnum_counter:Nn  { c@#1 } {#1} }
\cs_new:Npn \zhnum_counter:Nn #1#2
  {
    \token_if_int_register:NTF #1
      { \zhnum_int:n {#1} }
      { \__msg_expandable_error:n { `#2' is not a LaTeX counter. } }
  }
\cs_new:Npn \zhnum_int:n #1
  {
    \int_compare:nNnTF {#1} > \c_zero
      { \zhnum_parse_number:f { \int_eval:n {#1} } }
      {
        \int_compare:nNnTF {#1} < \c_zero
          {
            \c__zhnum_minus_tl
            \zhnum_parse_number:f { \int_eval:n { - #1 } }
          }
          { \c__zhnum_zero_tl }
      }
  }
```

(*End definition for* `\zhnum_counter:n`.)

\zhnum_integer:n 对整数的处理。这个函数基本抄录自 l3bigint 的 `\__bingint_read_do:nn`。它可以正确取得符号，去掉多余的零，还可以循环展开数字。但它在遇到非数字的时候就停止了循环，我们可能需要非数字(例如逗号)来作为分隔符号。因此对它略作修改，跳过非数字。

```
\cs_new:Npn \zhnum_integer:n #1
  {
    \exp_after:wN \__zhnum_read_integer:www
    \tex_number:D
      \exp_after:wN \__zhnum_read_sign_loop:N
      \tex_romannumeral:D -`0 \use:n
      #1 \exp_stop_f: \q_recursion_tail \q_recursion_stop
         \__zhnum_result:nn { \c_zero } { } ;
  }
\cs_new:Npn \__zhnum_read_sign_loop:N #1
  {
    \if:w + \if:w - \exp_not:N #1 + \fi: \exp_not:N #1
```

```
      \exp_after:wN \__zhnum_read_sign_loop:N
      \tex_romannumeral:D -`0 \exp_after:wN \use:n
    \else:
      1 \exp_after:wN ;
      \tex_romannumeral:D -`0
        \exp_after:wN \__zhnum_read_zeros_loop:N
        \exp_after:wN #1
    \fi:
  }
\cs_new:Npn \__zhnum_read_zeros_loop:N #1
  {
    \if:w 0 \exp_not:N #1
      \exp_after:wN \__zhnum_read_zeros_loop:N
      \tex_romannumeral:D -`0 \exp_after:wN \use:n
    \else:
      \exp_after:wN \__zhnum_read_abs_loop:Nw
      \exp_after:wN #1
    \fi:
  }
```

(*End definition for* `\zhnum_integer:n`.)

`\__zhnum_read_abs_loop:Nw` 当数字很大时，l3bigint 的实现会造成 TEX 内存溢出：

```
! TeX capacity exceeded, sorry [expansion depth=10000].
```

我们在这里参考 `\__tl_act:NNNnn` 的实现对它进行了改进。

```
\cs_new:Npn \__zhnum_read_abs_loop:Nw #1#2 \q_recursion_stop
  {
    \zhnum_if_digit:NTF #1
      { \__zhnum_output:nnwnn { + \c_one } #1 }
      { \quark_if_recursion_tail_stop_do:Nn #1 { \__zhnum_loop_end:wnn } }
    \exp_after:wN \__zhnum_read_abs_loop:Nw
      \tex_romannumeral:D -`0 \use:n #2 \q_recursion_stop
  }
\cs_new:Npn \__zhnum_output:nnwnn #1#2#3 \__zhnum_result:nn #4#5
  { #3 \__zhnum_result:nn { #4#1 } { #5#2 } }
\cs_new:Npn \__zhnum_loop_end:wnn #1 \__zhnum_result:nn #2#3
  { \int_eval:n {#2} ; #3 }
```

(*End definition for* `\__zhnum_read_abs_loop:Nw`.)

`\__zhnum_read_integer:www` `#1` 符号，`#3` 是绝对值，`#2` 是绝对值的长度。

```
\cs_new:Npn \__zhnum_read_integer:www #1 ; #2 ; #3 ;
  {
    \int_compare:nNnTF {#2} = \c_zero
      { \c__zhnum_zero_tl }
      {
        \int_compare:nNnF {#1} = \c_one
          { \c__zhnum_minus_tl }
        \zhnum_parse_number:nn {#2} {#3}
      }
  }
```

(*End definition for* `\__zhnum_read_integer:www`.)

`\zhnum_if_digit:NTF` 判断 `#1` 是否为数字位。

```
\cs_new:Npn \zhnum_if_digit:NTF #1
  {
    \if_int_compare:w \c_nine < 1 \exp_not:N #1 \exp_stop_f:
      \exp_after:wN \use_i:nn
    \else:
      \exp_after:wN \use_ii:nn
    \fi:
  }
```

(*End definition for* `\zhnum_if_digit:NTF`.)

\zhnum_parse_number:n
\zhnum_parse_number:nn

```
\cs_new_nopar:Npn \zhnum_parse_number:n #1
  { \exp_args:Nf \zhnum_parse_number:nn { \tl_count:n {#1} } {#1} }
\cs_new_nopar:Npn \zhnum_parse_number:nn #1#2
  { \exp_args:Nf \__zhnum_parse_number:nnn { \int_mod:nn {#1} \c_four } {#1} {#2} }
\cs_new_nopar:Npn \__zhnum_parse_number:nnn #1#2#3
  {
    \int_compare:nNnTF {#2} < \c_two
      { \zhnum_digit_map:n {#3} }
      {
        \zhnum_split_number:fNNfn { \zhnum_insert_zeros:n {#1} #3 }
          \c_true_bool \c_true_bool
          {
            \int_compare:nNnTF {#1} = \c_zero
              { \int_eval:n { (#2) / \c_four - \c_one } }
              { \int_div_truncate:nn {#2} \c_four }
          }
          { \c_zero }
      }
  }
\cs_generate_variant:Nn \zhnum_parse_number:n { f }
```

(*End definition for* `\zhnum_parse_number:n` *and* `\zhnum_parse_number:nn`.)

\zhnum_insert_zeros:n

为了处理的方便,在整数前面补上适当的 0,使其位数可以被 4 整除。

```
\cs_new_nopar:Npn \zhnum_insert_zeros:n #1
  {
    \if_case:w \etex_numexpr:D #1 - \c_one \scan_stop:
      \or: \exp_after:wN \use_none:n
      \or: \exp_after:wN \use_none:nn
    \else: \exp_after:wN \use_none:nnn
    \fi:
    0 0 0
  }
```

(*End definition for* `\zhnum_insert_zeros:n`.)

\zhnum_split_number:nNNnn

将输入的整数由高位到低位,以四位为一段进行处理。

```
\cs_new_nopar:Npn \zhnum_split_number:nNNnn #1#2#3#4#5
  {
    \__zhnum_split_number:fnNNnn { \zhnum_number_item:nn {#1} {#5} }
      {#1} #2#3 {#4} {#5}
  }
\cs_new_nopar:Npn \__zhnum_split_number:nnNNnn #1#2#3#4#5#6
  {
    \int_compare:nNnTF {#1} = \c_zero
      { \__zhnum_split_number_aux:NNfnnn \c_false_bool \c_true_bool }
      {
        \bool_if:NF #3 { \c__zhnum_zero_tl }
        \zhnum_process_number:NNn #3#4 {#1}
        \exp_args:Nf \zhnum_scale_map:n { \int_eval:n { #5 - #6 } }
        \int_compare:nNnTF { \int_mod:nn {#1} \c_ten } = \c_zero
          { \__zhnum_split_number_aux:NNfnnn \c_false_bool \c_true_bool }
          { \__zhnum_split_number_aux:NNfnnn \c_true_bool \c_false_bool }
      }
    { \int_eval:n { #6 + \c_one } } {#2} {#5} {#6}
  }
\cs_new_nopar:Npn \__zhnum_split_number_aux:NNnnnn #1#2#3#4#5#6
  {
    \int_compare:nNnF {#5} = {#6}
      { \zhnum_split_number:nNNnn {#4} #1#2 {#5} {#3} }
  }
\cs_generate_variant:Nn \zhnum_split_number:nNNnn { fNNf }
\cs_generate_variant:Nn \__zhnum_split_number:nnNNnn { f }
\cs_generate_variant:Nn \__zhnum_split_number_aux:NNnnnn { NNf }
```

(*End definition for* `\zhnum_split_number:nNNnn`.)

\zhnum_number_item:nn 截取整数的其中四位数。

```
\cs_new_nopar:Npn \zhnum_number_item:nn #1#2
  { \__zhnum_number_item:nNNNN {#2} #1 \q_recursion_stop }
\cs_new_nopar:Npn \__zhnum_number_item:nNNNN #1#2#3#4#5
  {
    \int_compare:nNnTF {#1} = \c_zero
      { \__zhnum_recursion_stop:NNNNw #2#3#4#5 }
      { \__zhnum_number_item:fNNNN { \int_eval:n { #1 - \c_one } } }
  }
\cs_generate_variant:Nn \__zhnum_number_item:nNNNN { f }
\cs_new_nopar:Npn \__zhnum_recursion_stop:NNNNw #1#2#3#4#5 \q_recursion_stop
  { #1#2#3#4 }
```

(*End definition for* `\zhnum_number_item:nn`.)

\zhnum_process_number:NNn 对四位数字按情况进行处理。

zhnum_process_number:NNNNNN

```
\cs_new_nopar:Npn \zhnum_process_number:NNn #1#2#3
  { \zhnum_process_number:NNNNNN #3#1#2 }
\cs_new_nopar:Npn \zhnum_process_number:NNNNNN #1#2#3#4#5#6
  {
    \int_compare:nNnTF {#1} = \c_zero
      { \bool_if:NF #6 { \c__zhnum_zero_tl } }
      { \zhnum_digit_map:n {#1} \c__zhnum_thousand_tl }
    \int_compare:nNnTF {#2} = \c_zero
      { \int_compare:nNnT { #1 * (#3#4) } > \c_zero { \c__zhnum_zero_tl } }
      {
        \bool_if:nTF
          { \l__zhnum_ancient_bool && \int_compare_p:nNn {#2} = \c_two }
          { \zhnum_digit_map:n { #2 00 } }
          { \zhnum_digit_map:n {#2} \c__zhnum_hundred_tl }
      }
    \int_compare:nNnTF {#3} = \c_zero
      { \int_compare:nNnT { #2 * #4 } > \c_zero { \c__zhnum_zero_tl } }
      {
        \bool_if:nF
          {
            \int_compare_p:nNn {#3}   = \c_one  &&
            \int_compare_p:nNn {#1#2} = \c_zero && #6 && #5
          }
          {
            \bool_if:nTF
              {
                \l__zhnum_ancient_bool                    &&
                ( \int_compare_p:nNn {#3} = \c_two   ||
                  \int_compare_p:nNn {#3} = \c_three ||
                  \int_compare_p:nNn {#3} = \c_four )
              }
              { \zhnum_digit_map:n { #3 0 } \use_none:n }
              { \zhnum_digit_map:n {#3} }
          }
        \c__zhnum_ten_tl
      }
    \int_compare:nNnF {#4} = \c_zero { \zhnum_digit_map:n {#4} }
  }
```

(*End definition for* `\zhnum_process_number:NNn` *and* `\zhnum_process_number:NNNNNN`.)

\zhdigits 将输入的数字输出为中文数字串输出。

\zhdigitswithoptions

```
\DeclareExpandableDocumentCommand \zhdigits { s o m }
  {
    \IfNoValueTF {#2}
      { \zhnum_digits:Nn #1 {#3} }
      {
        \IfBooleanTF #1
          { \zhdigitswithoptions * }
          { \zhdigitswithoptions }
        {#2} {#3}
      }
  }
```

```
\NewDocumentCommand \zhdigitswithoptions { s m m }
  {
    \group_begin:
      \keys_set:nn { zhnum / options } {#2}
      \zhnum_digits:Nn #1 {#3}
    \group_end:
  }
```

(*End definition for* `\zhdigits` *and* `\zhdigitswithoptions`*. These functions are documented on page 3.*)

`\zhnum_digits_zero:n` `\zhnum_digits_null:n` 快捷方式。

```
\cs_new_nopar:Npn \zhnum_digits_zero:n
  { \zhnum_digits:Nn \BooleanTrue }
\cs_new_nopar:Npn \zhnum_digits_null:n
  { \zhnum_digits:Nn \BooleanFalse }
\cs_generate_variant:Nn \zhnum_digits_null:n { V }
```

(*End definition for* `\zhnum_digits_zero:n` *and* `\zhnum_digits_null:n`*.*)

`\zhnum_digits:Nn` 与 `\zhnum_integer:n` 类似，但不用去掉多余的零。

```
\cs_new:Npn \zhnum_digits:Nn #1#2
  {
    \exp_after:wN \__zhnum_read_digits:w
    \tex_number:D
      \exp_after:wN \__zhnum_read_sign_loop:NN \exp_after:wN #1
      \tex_romannumeral:D -`0 \use:n
      #2 \exp_stop_f: \q_recursion_tail \q_recursion_stop
  }
\cs_new:Npn \__zhnum_read_sign_loop:NN #1#2
  {
    \if:w + \if:w - \exp_not:N #2 + \fi: \exp_not:N #2
      \exp_after:wN \__zhnum_read_sign_loop:NN \exp_after:wN #1
      \tex_romannumeral:D -`0 \exp_after:wN \use:n
    \else:
      1 \exp_after:wN ;
        \exp_after:wN \__zhnum_read_digits_loop:NN
        \exp_after:wN #1
        \exp_after:wN #2
    \fi:
  }
\cs_new:Npn \__zhnum_read_digits_loop:NN #1#2
  {
    \zhnum_if_digit:NTF #2
      { \__zhnum_output_digits:NN #1#2 }
      {
        \quark_if_recursion_tail_stop:N #2
        \if:w .\exp_not:N #2 \exp_after:wN \c__zhnum_dot_tl \fi:
      }
    \exp_after:wN \__zhnum_read_digits_loop:NN \exp_after:wN #1
        \tex_romannumeral:D -`0 \use:n
  }
\cs_new:Npn \__zhnum_read_digits:w #1 ;
  {
    \int_compare:nNnF {#1} = \c_one
      { \c__zhnum_minus_tl }
  }
\cs_new:Npn \__zhnum_output_digits:NN #1#2
  {
    \cs:w
      c__zhnum_
        \if_int_compare:w #2 = \c_zero
          \IfBooleanTF #1 { zero } { null }
        \else:
          #2
        \fi:
      _tl
    \cs_end:
  }
```

(*End definition for* `\zhnum_digits:Nn`.)

`\zhdate` 输出中文日期。

```
\DeclareExpandableDocumentCommand \zhdate { s m }
  {
    \__zhnum_date:www #2 \q_stop
    \IfBooleanT {#1}
      {  \__zhnum_week_day:www #2 \q_stop }
  }
\cs_new_nopar:Npn \__zhnum_date:www #1/#2/#3 \q_stop
  {
    \zhnum_check_time:Nn \zhnum_digits_null:n {#1} \c__zhnum_year_tl
    \zhnum_check_time:Nn \zhnum_integer:n {#2} \c__zhnum_month_tl
    \zhnum_check_time:Nn \zhnum_integer:n {#3} \c__zhnum_day_tl
  }
```

(*End definition for* `\zhdate`. *This function is documented on page* 2.)

`\zhtoday` 输出当天日期。

```
\cs_new_nopar:Npn \zhtoday
  {
    \zhnum_check_time:Nn \zhnum_digits_null:V \tex_year:D \c__zhnum_year_tl
    \zhnum_check_time:Nn \zhnum_int:n \tex_month:D \c__zhnum_month_tl
    \zhnum_check_time:Nn \zhnum_int:n \tex_day:D   \c__zhnum_day_tl
  }
```

(*End definition for* `\zhtoday`. *This function is documented on page* 2.)

`\zhnum_check_time:Nn` 判断是用中文数字还是用阿拉伯数组。

```
\cs_new_nopar:Npn \zhnum_check_time:Nn #1
  { \bool_if:NTF \l__zhnum_time_bool {#1} { \int_eval:n } }
```

(*End definition for* `\zhnum_check_time:Nn`.)

`\zhweekday` 输出星期

```
\cs_new_nopar:Npn \zhweekday #1
  { \__zhnum_week_day:www #1 \q_stop }
```

(*End definition for* `\zhweekday`. *This function is documented on page* 2.)

`\__zhnum_week_day:www` 用 Zeller 公式计算的结果 h 与实际星期的关系是 $d = h + 5 \pmod 7 + 1$。

```
\cs_new_nopar:Npn \__zhnum_week_day:www #1/#2/#3 \q_stop
  {
    \if_case:w \etex_numexpr:D \zhnum_Zeller:nnn {#1} {#2} {#3} \scan_stop:
           \c__zhnum_sat_tl
      \or: \c__zhnum_sun_tl
      \or: \c__zhnum_mon_tl
      \or: \c__zhnum_tue_tl
      \or: \c__zhnum_wed_tl
      \or: \c__zhnum_thu_tl
      \or: \c__zhnum_fri_tl
    \fi:
  }
```

(*End definition for* `\__zhnum_week_day:www`.)

`\zhnum_Zeller:nnn` 用 Zeller 公式[1] 计算星期几。

`\zhnum_Zeller_aux:Nnnn`

`\zhnum_two_digits:n`

```
\cs_new_nopar:Npn \zhnum_Zeller:nnn #1#2#3
  {
    \int_compare:nNnTF
      { #1 \zhnum_two_digits:n {#2} \zhnum_two_digits:n {#3} } > { 1582 10 04 }
      { \__zhnum_Zeller_aux:Nnnn \zhnum_Zeller_Gregorian:nnn }
      { \__zhnum_Zeller_aux:Nnnn \zhnum_Zeller_Julian:nnn }
    {#1} {#2} {#3}
  }
\cs_new_nopar:Npn \__zhnum_Zeller_aux:Nnnn  #1#2#3#4
  {
```

[1] http://en.wikipedia.org/wiki/Zeller's_congruence

```
    \int_compare:nNnTF {#3} < \c_three
      { #1 { #2 - \c_one } { #3 + \c_twelve } {#4} }
      { #1 {#2} {#3} {#4} }
  }
\cs_new_nopar:Npn \zhnum_two_digits:n #1
  {
    \int_compare:nNnT {#1} < \c_ten { 0 }
    \int_eval:n {#1}
  }
```

(*End definition for* `\zhnum_Zeller:nnn`, `\zhnum_Zeller_aux:Nnnn`, *and* `\zhnum_two_digits:n`.)

`\zhnum_Zeller_Gregorian:nnn` 格里历(1582 年 10 月 15 日及以后)的计算公式

$$h = \left(q + \left\lfloor \frac{26(m+1)}{10} \right\rfloor + Y + \left\lfloor \frac{Y}{4} \right\rfloor + 6\left\lfloor \frac{Y}{100} \right\rfloor + \left\lfloor \frac{Y}{400} \right\rfloor\right) \pmod 7$$

其中 Y 为年, m 为月, q 为日;若 $m = 1, 2$, 则令 $m \mathrel{+}= 12$, 同时 $Y --$。

```
\cs_new_nopar:Npn \zhnum_Zeller_Gregorian:nnn #1#2#3
  {
    \int_mod:nn
      {
          (#3)
        + \int_div_truncate:nn { 26 * ( #2 + \c_one ) } \c_ten
        + (#1)
        + \int_div_truncate:nn {#1} \c_four
        + \c_six * \int_div_truncate:nn {#1} \c_one_hundred
        + \int_div_truncate:nn {#1} { 400 }
      }
      { \c_seven }
  }
```

(*End definition for* `\zhnum_Zeller_Gregorian:nnn`.)

`\zhnum_Zeller_Julian:nnn` 儒略历(1582 年 10 月 4 日及以前)的计算公式

$$h = \left(q + \left\lfloor \frac{26(m+1)}{10} \right\rfloor + Y + \left\lfloor \frac{Y}{4} \right\rfloor + 5\right) \pmod 7$$

```
\cs_new_nopar:Npn \zhnum_Zeller_Julian:nnn #1#2#3
  {
    \int_mod:nn
      {
          (#3)
        + \int_div_truncate:nn { 26 * ( #2 + \c_one ) } \c_ten
        + (#1)
        + \int_div_truncate:nn {#1} \c_four
        + \c_five
      }
      { \c_seven }
  }
```

(*End definition for* `\zhnum_Zeller_Julian:nnn`.)

`\zhtime` 输出时间。

```
\cs_new_nopar:Npn \zhtime #1
  { \__zhnum_time:ww #1 \q_stop }
\group_begin:
\char_set_lccode:nn { `\; } { `\: }
\tl_to_lowercase:n
  {
    \group_end:
    \cs_new_nopar:Npn \__zhnum_time:ww #1 ; #2 \q_stop
      {
        \zhnum_check_time:Nn \zhnum_integer:n {#1} \c__zhnum_hour_tl
        \zhnum_check_time:Nn \zhnum_integer:n {#2} \c__zhnum_minute_tl
      }
  }
```

(*End definition for* `\zhtime`. *This function is documented on page* 2.)

`\zhcurrtime` 输出当前时间。

```
\cs_new_nopar:Npn \zhcurrtime
  {
    \zhnum_check_time:Nn \zhnum_int:n
      { \int_div_truncate:nn \tex_time:D { 60 } } \c__zhnum_hour_tl
    \zhnum_check_time:Nn \zhnum_int:n
      { \int_mod:nn \tex_time:D { 60 } } \c__zhnum_minute_tl
  }
```

(*End definition for* `\zhcurrtime`. *This function is documented on page* 2.)

`\zhnum_digit_map:n` 阿拉伯数字与中文数字的映射。

```
\cs_new_nopar:Npn \zhnum_digit_map:n #1
  { \use:c { c__zhnum_ #1 _tl } }
```

(*End definition for* `\zhnum_digit_map:n`.)

`\zhnum_scale_map:n` 大数系统的映射。

`\zhnum_scale_map_loop:n`

```
\cs_new_nopar:Npn \zhnum_scale_map:n #1
  {
    \cs_if_exist_use:cF { c__zhnum_s #1 _tl }
      { \zhnum_scale_map_hook:n {#1} }
  }
\cs_new_nopar:Npn \zhnum_scale_map_loop:n #1
  { \zhnum_scale_map:n { \int_mod:nn {#1} \g__zhnum_scale_int } }
\int_new:N \g__zhnum_scale_int
\int_set_eq:NN \g__zhnum_scale_int \c_eleven
\cs_new_eq:NN \zhnum_scale_map_hook:n \zhnum_scale_map_loop:n
```

(*End definition for* `\zhnum_scale_map:n` *and* `\zhnum_scale_map_loop:n`.)

`\zhnumExtendScaleMap`

```
\NewDocumentCommand \zhnumExtendScaleMap { > { \TrimSpaces } o m }
  {
    \int_zero:N \l_tmpa_int
    \clist_map_inline:nn {#2}
      {
        \int_incr:N \l_tmpa_int
        \tl_set:Nx \l_tmpa_tl
          { c__zhnum_s \int_eval:n { \l_tmpa_int + \c_eleven } _tl }
        \tl_if_exist:cF { \l_tmpa_tl }
          { \int_incr:N \g__zhnum_scale_int }
        \tl_set:cn { \l_tmpa_tl } {##1}
      }
    \IfNoValueF {#1}
      { \cs_set:Npn \zhnum_scale_map_hook:n ##1 {#1} }
  }
```

(*End definition for* `\zhnumExtendScaleMap`. *This function is documented on page* 2.)

根据需要设置中文阿拉伯数字。

```
\group_begin:
  \tl_set:Nn \l_tmpa_tl
    {
      -   .tl_set:N = \l__zhnum_minus_tl ,
      -0  .tl_set:N = \l__zhnum_null_tl ,
    }
  \tl_put_right:Nx \l_tmpa_tl
    {
      E2  .tl_set:N = \exp_not:c { l__zhnum_ 100 _tl } ,
      E3  .tl_set:N = \exp_not:c { l__zhnum_ 1000 _tl } ,
      FE2 .tl_set:N = \exp_not:c { l__zhnum_financial_ 100 _tl } ,
      FE3 .tl_set:N = \exp_not:c { l__zhnum_financial_ 1000 _tl } ,
    }
  \clist_map_inline:nn
    { 0 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 100 , 1000 }
    {
```

```
      \tl_put_right:Nx \l_tmpa_tl
        {
           #1 .tl_set:N = \exp_not:c { l__zhnum_ #1 _tl } ,
          F#1 .tl_set:N = \exp_not:c { l__zhnum_financial_ #1 _tl } ,
        }
    }
  \clist_map_inline:nn
    { 20 , 30 , 40 , 200 }
    {
      \tl_put_right:Nx \l_tmpa_tl
        { #1 .tl_set:N = \exp_not:c { l__zhnum_ #1 _tl } , }
    }
  \clist_map_inline:nn
    { 4 , 8 , 12 , 16 , 20 , 24 , 28 , 32 , 36 , 40 , 44 }
    {
      \tl_put_right:Nx \l_tmpa_tl
        { E#1 .tl_set:N = \exp_not:c { l__zhnum_ s \int_eval:n { #1 / 4 } _tl } , }
    }
  \clist_map_inline:nn
    {
      dot , and , parts , year , month , day , weekday , hour , minute
      mon , tue , wed , thu , fri , sat , sun
    }
    {
      \tl_put_right:Nx \l_tmpa_tl
        { #1 .tl_set:N = \exp_not:c { l__zhnum_ #1 _tl } , }
    }
\use:x
  {
    \group_end:
    \keys_define:nn { zhnum / options } { \exp_not:o \l_tmpa_tl }
  }
```

\zhnum_set_digits_map:nn
\zhnum_set_digits_map:nnn
\zhnum_set_financial_map:nn
\zhnum_set_financial_map:nnn
\l__zhnum_cfg_map_prop
\l__zhnum_cfg_map_var_prop
\l__zhnum_cfg_map_finan_prop

将配置文件中的中文数字保存到 prop 变量中。

```
\cs_new_protected_nopar:Npn \zhnum_set_digits_map:nn #1#2
  { \prop_put:Nnn \l__zhnum_cfg_map_prop {#1} {#2} }
\cs_new_protected_nopar:Npn \zhnum_set_digits_map:nnn #1#2#3
  {
    \prop_put_if_new:Nnn \l__zhnum_cfg_map_prop {#1} {#3}
    \prop_put:Nnn \l__zhnum_cfg_map_var_prop {#1_#2} {#3}
  }
\cs_new_protected_nopar:Npn \zhnum_set_financial_map:nn #1#2
  { \prop_put:Nnn \l__zhnum_cfg_map_finan_prop {#1} {#2} }
\cs_new_protected_nopar:Npn \zhnum_set_financial_map:nnn #1#2#3
  {
    \prop_put_if_new:Nnn \l__zhnum_cfg_map_finan_prop {#1} {#3}
    \prop_put:Nnn \l__zhnum_cfg_map_var_prop { financial_#1_#2 } {#3}
  }
\prop_new:N \l__zhnum_cfg_map_prop
\prop_new:N \l__zhnum_cfg_map_var_prop
\prop_new:N \l__zhnum_cfg_map_finan_prop
```

(*End definition for* `\zhnum_set_digits_map:nn` *and others.*)

\zhnum_parse_config:
\zhnum_check_simp:nn
\zhnum_check_financial:nn
\zhnum_set_zero:
\zhnum_set_week_day:

将 prop 表转化到单独的 tl 变量。

```
\cs_new_protected_nopar:Npn \zhnum_parse_config:
  {
    \prop_map_function:NN \l__zhnum_cfg_map_prop \zhnum_check_simp:nn
    \bool_if:NF \l__zhnum_reset_bool
      { \prop_map_function:NN \l__zhnum_cfg_map_prop \zhnum_check_financial:nn }
    \zhnum_set_zero:
    \zhnum_set_week_day:
  }
\cs_new_protected_nopar:Npn \zhnum_check_simp:nn #1#2
  {
    \__zhnum_check_simp_aux:nn {#1} {#2}
    \prop_get:NnNT \l__zhnum_cfg_map_finan_prop {#1} \l_tmpa_tl
      { \exp_args:Nno \__zhnum_check_simp_aux:nn { financial_ #1 } \l_tmpa_tl }
  }
```

```
\cs_new_protected_nopar:Npn \__zhnum_check_simp_aux:nn #1#2
  {
    \prop_get:NnNTF \l__zhnum_cfg_map_var_prop { #1 _trad } \l_tmpa_tl
      {
        \prop_get:NnNTF \l__zhnum_cfg_map_var_prop { #1 _simp } \l_tmpb_tl
          {
            \tl_set:cx { l__zhnum_ #1 _tl }
              {
                \exp_not:n { \bool_if:NTF \l__zhnum_simp_bool }
                  { \exp_not:o \l_tmpb_tl } { \exp_not:o \l_tmpa_tl }
              }
          }
          {
            \tl_set:cx { l__zhnum_ #1 _tl }
              {
                \exp_not:n { \bool_if:NTF \l__zhnum_simp_bool }
                  { \exp_not:n {#2} } { \exp_not:o \l_tmpa_tl }
              }
          }
      }
      { \tl_set:cn { l__zhnum_ #1 _tl } {#2} }
  }
\cs_new_protected_nopar:Npn \zhnum_check_financial:nn #1#2
  {
    \prop_get:NnNTF \l__zhnum_cfg_map_finan_prop {#1} \l_tmpa_tl
      {
        \zhnum_assgin_const_tl:cx { c__zhnum_ #1 _tl }
          {
            \exp_not:n { \bool_if:NTF \l__zhnum_normal_bool }
              { \exp_not:c { l__zhnum_ #1 _tl } }
              { \exp_not:c { l__zhnum_financial_ #1 _tl } }
          }
      }
      {
        \zhnum_assgin_const_tl:cx
          { c__zhnum_ #1 _tl } { \exp_not:c { l__zhnum_ #1 _tl } }
      }
  }
\cs_new_protected_nopar:Npn \zhnum_set_zero:
  {
    \tl_set:cx { l__zhnum_0_tl }
      {
        \exp_not:n { \bool_if:nTF \l__zhnum_null_bool }
          { \exp_not:o \l__zhnum_null_tl } { \exp_not:v { l__zhnum_0_tl } }
      }
  }
\cs_new_protected_nopar:Npn \zhnum_set_week_day:
  {
    \tl_set:Nx \l__zhnum_mon_tl
      { \exp_not:N \c__zhnum_weekday_tl \exp_not:v { l__zhnum_1_tl } }
    \tl_set:Nx \l__zhnum_tue_tl
      { \exp_not:N \c__zhnum_weekday_tl \exp_not:v { l__zhnum_2_tl } }
    \tl_set:Nx \l__zhnum_wed_tl
      { \exp_not:N \c__zhnum_weekday_tl \exp_not:v { l__zhnum_3_tl } }
    \tl_set:Nx \l__zhnum_thu_tl
      { \exp_not:N \c__zhnum_weekday_tl \exp_not:v { l__zhnum_4_tl } }
    \tl_set:Nx \l__zhnum_fri_tl
      { \exp_not:N \c__zhnum_weekday_tl \exp_not:v { l__zhnum_5_tl } }
    \tl_set:Nx \l__zhnum_sat_tl
      { \exp_not:N \c__zhnum_weekday_tl \exp_not:v { l__zhnum_6_tl } }
    \tl_set:Nx \l__zhnum_sun_tl
      { \exp_not:N \c__zhnum_weekday_tl \exp_not:o \l__zhnum_day_tl }
    \bool_if:NF \l__zhnum_reset_bool
      {
        \clist_map_inline:nn { mon , tue , wed , thu , fri , sat , sun }
          {
            \zhnum_assgin_const_tl:cx
              { c__zhnum_ ##1 _tl } { \exp_not:c { l__zhnum_ ##1 _tl } }
          }
```

```
        \bool_set_true:N \l__zhnum_reset_bool
      }
  }
\cs_new_eq:NN \zhnum_assgin_const_tl:cx \tl_const:cx
\AtEndOfPackage
  { \cs_set_eq:NN \zhnum_assgin_const_tl:cx \tl_set:cx }
```

(*End definition for* `\zhnum_parse_config:` *and others.*)

`\zhnum_set_alias:` 一些易于使用的别名。

```
\cs_new_protected_nopar:Npn \zhnum_set_alias:
  {
    \tl_const:Nx \c__zhnum_zero_tl     { \exp_not:c { c__zhnum_ 0 _tl } }
    \tl_const:Nx \c__zhnum_ten_tl      { \exp_not:c { c__zhnum_ 10 _tl } }
    \tl_const:Nx \c__zhnum_hundred_tl  { \exp_not:c { c__zhnum_ 100 _tl } }
    \tl_const:Nx \c__zhnum_thousand_tl { \exp_not:c { c__zhnum_ 1000 _tl } }
  }
\AtEndOfPackage { \zhnum_set_alias: }
```

(*End definition for* `\zhnum_set_alias:`.)

`\zhnum_load_cfg:n` 根据选定编码载入配置文件。

```
\cs_new_protected_nopar:Npn \zhnum_load_cfg:n #1
  {
    \zhnum_set_cfg_name:Nn \l__zhnum_cfg_tl {#1}
    \tl_if_eq:NNF \l__zhnum_cfg_tl \l__zhnum_last_cfg_tl
      { \zhnum_update_cfg:n {#1} }
    \zhnum_parse_config:
  }
\cs_generate_variant:Nn \zhnum_load_cfg:n { o }
\cs_new_protected_nopar:Npn \zhnum_update_cfg:n #1
  {
    \prop_if_exist:cTF { g__zhnum_cfg_ \l__zhnum_cfg_tl _prop }
      { \tl_set_eq:NN \l__zhnum_last_cfg_tl \l__zhnum_cfg_tl }
      { \zhnum_input_cfg:n  {#1} }
    \__zhnum_update_cfg_prop:N \prop_set_eq:Nc
  }
\cs_new_protected_nopar:Npn \zhnum_input_cfg:n #1
  {
    \file_if_exist_input:nTF { zhnumber - #1 .cfg }
      {
        \bool_set_false:N \l__zhnum_reset_bool
        \__zhnum_update_cfg_prop:N \__zhnum_prop_initial:Nn
        \group_begin:
          \zhnum_set_catcode:
      }
      {
        \msg_error:nnx { zhnumber } { file-not-found } {#1}
        \use_none:nnn
      }
    \__zhnum_update_cfg_prop:N \__zhnum_prop_gset_eq:Nn
    \group_end:
  }
\cs_new_protected_nopar:Npn \__zhnum_update_cfg_prop:N #1
  {
    #1 \l__zhnum_cfg_map_prop       { g__zhnum_cfg_ \l__zhnum_cfg_tl _prop }
    #1 \l__zhnum_cfg_map_var_prop   { g__zhnum_cfg_var_ \l__zhnum_cfg_tl _prop }
    #1 \l__zhnum_cfg_map_finan_prop { g__zhnum_cfg_finan_ \l__zhnum_cfg_tl _prop }
  }
\cs_new_protected:Npn \__zhnum_prop_initial:Nn #1#2
  {
    \prop_clear:N #1
    \prop_new:c {#2}
  }
\cs_new_protected:Npn \__zhnum_prop_gset_eq:Nn #1#2
  { \prop_gset_eq:cN {#2} #1 }
\tl_new:N \l__zhnum_cfg_tl
\tl_new:N \l__zhnum_last_cfg_tl
\bool_new:N \l__zhnum_reset_bool
```

```
\msg_new:nnnn  { zhnumber } { file-not-found }
  { File~`#1'~not~found. }
  {
    The~requested~file~could~not~be~found~in~the~current~directory,~
    in~the~TeX~search~path~or~in~the~LaTeX~search~path.
  }
```

(*End definition for* `\zhnum_load_cfg:n`.)

`\zhnum_if_unicode_engine_p:`
`\zhnum_if_unicode_engine:TF`

使用 upTEX 的时候，也不必将汉字的首字符设置为活动字符。判断 `^^^^0021` 是否为单个记号的办法对 upTEX 不适用。因此，参考 ifuptex 宏包，通过 `\kchar` 是否为 primitive 来判断。

```
\pdftex_if_engine:TF
  {
    \str_if_eq_x:nnTF
      { \token_to_str:N \kchar }
      { \token_to_meaning:N \kchar }
  }
  { \use_i:nn }
  {
    \cs_new_eq:NN \zhnum_if_unicode_engine_p: \c_true_bool
    \cs_new_eq:NN \zhnum_if_unicode_engine:TF \use_i:nn
  }
  {
    \cs_new_eq:NN \zhnum_if_unicode_engine_p: \c_false_bool
    \cs_new_eq:NN \zhnum_if_unicode_engine:TF \use_ii:nn
  }
```

(*End definition for* `\zhnum_if_unicode_engine:TF`.)

`\zhnum_set_catcode:`
`\zhnum_set_cfg_name:Nn`
`\zhnum_reset_config:`

设置与恢复配置文件前后的 catcode。pdfLATEX 需要将汉字的首字节设置为活动字符。

```
\if_predicate:w \zhnum_if_unicode_engine_p:
  \cs_new_eq:NN \zhnum_set_catcode: \prg_do_nothing:
  \cs_new_protected_nopar:Npn \zhnum_set_cfg_name:Nn #1#2
    {
      \tl_set:Nx \l__zhnum_encoding_tl {#2}
      \tl_set:Nx #1 { \tl_to_str:N \l__zhnum_encoding_tl }
    }
  \cs_new_eq:NN \zhnum_reset_config: \zhnum_parse_config:
\else:
  \cs_new_protected_nopar:Npn \zhnum_set_catcode:
    { \bool_if:NT \l__zhnum_active_char_bool { \zhnum_set_active: } }
  \cs_new_protected_nopar:Npn \zhnum_set_active:
    {
      \str_case:onTF { \l__zhnum_encoding_tl }
        {
          { gbk }  { \int_set:Nn \l__zhnum_byte_min_int { "81 } }
          { big5 } { \int_set:Nn \l__zhnum_byte_min_int { "A1 } }
        }
        { \int_set:Nn \l__zhnum_byte_max_int { "FE } }
        {
          \int_set:Nn \l__zhnum_byte_min_int { "E0 }
          \int_set:Nn \l__zhnum_byte_max_int { "EF }
        }
      \int_step_function:nnnN
        { \l__zhnum_byte_min_int } { \c_one }
        { \l__zhnum_byte_max_int } \char_set_catcode_active:n
    }
  \int_new:N \l__zhnum_byte_min_int
  \int_new:N \l__zhnum_byte_max_int
  \cs_new_protected_nopar:Npn \zhnum_set_cfg_name:Nn #1#2
    {
      \tl_set:Nx \l__zhnum_encoding_tl {#2}
      \tl_set:Nx #1
        {
          \tl_to_str:N \l__zhnum_encoding_tl
          \bool_if:NT \l__zhnum_active_char_bool
            { \tl_to_str:n { _active } }
        }
    }
```

```
  \cs_new_protected_nopar:Npn \zhnum_reset_config:
    { \zhnum_load_cfg:o { \l__zhnum_encoding_tl } }
  \bool_new:N \l__zhnum_active_char_bool
  \bool_set_true:N \l__zhnum_active_char_bool
\fi:
```

(*End definition for* `\zhnum_set_catcode:`, `\zhnum_set_cfg_name:Nn`, *and* `\zhnum_reset_config:`.)

encoding
style
null
reset

宏包设置选项。

```
\keys_define:nn { zhnum / options }
  {
    encoding         .choices:nn =
      { UTF8 , GBK , Big5 }
      {
        \tl_set:Nx \l__zhnum_encoding_tl
          { \exp_args:No \tl_expandable_lowercase:n { \l_keys_choice_tl } }
        \zhnum_load_cfg:o { \l__zhnum_encoding_tl }
      } ,
    encoding          .default:n = { GBK } ,
    encoding / Bg5       .meta:n = { encoding = Big5 } ,
    encoding / unknown   .code:n =
      { \msg_error:nnn { zhnumber } { encoding-invalid } {#1} } ,
    style .multichoice: ,
    style / Normal       .code:n =
      {
        \bool_set_false:N \l__zhnum_ancient_bool
        \bool_set_true:N  \l__zhnum_normal_bool
      } ,
    style / Financial    .code:n =
      {
        \bool_set_false:N \l__zhnum_ancient_bool
        \bool_set_false:N \l__zhnum_normal_bool
      } ,
    style / Ancient      .code:n =
      {
        \bool_set_true:N \l__zhnum_ancient_bool
        \bool_set_true:N \l__zhnum_normal_bool
      } ,
    style / Simplified   .code:n = { \bool_set_true:N  \l__zhnum_simp_bool } ,
    style / Traditional  .code:n = { \bool_set_false:N \l__zhnum_simp_bool } ,
    style             .default:n = { Normal , Simplified } ,
    null              .bool_set:N = \l__zhnum_null_bool ,
    time .choice: ,
    time / Chinese       .code:n = { \bool_set_true:N \l__zhnum_time_bool } ,
    time / Arabic        .code:n = { \bool_set_false:N  \l__zhnum_time_bool } ,
    time              .default:n = { Arabic } ,
    reset                .code:n = { \zhnum_reset_config: } ,
    activechar        .bool_set:N = \l__zhnum_active_char_bool ,
  }
\tl_new:N \l__zhnum_encoding_tl
\msg_new:nnnn { zhnumber } { encoding-invalid }
  { The~encoding~`#1'~is~invalid. }
  { Available~encoding~are~`UTF8',~`GBK'~and~`Big5'. }
```

(*End definition for* `encoding` *and others. These functions are documented on page 1.*)

\zhnumsetup

在文档中设置 zhnumber 的接口。

```
\NewDocumentCommand \zhnumsetup { m }
  {
    \keys_set:nn { zhnum / options } {#1}
    \tex_ignorespaces:D
  }
```

(*End definition for* `\zhnumsetup`. *This function is documented on page 2.*)

初始化设置和执行宏包选项。

```
\keys_set:nn { zhnum / options } { style , time }
\ProcessKeysOptions { zhnum / options }
```

如果没有选定编码,则根据引擎自动设置编码。

```
\tl_if_empty:NT \l__zhnum_encoding_tl
  {
    \zhnum_if_unicode_engine:TF
      { \keys_set:nn { zhnum / options } { encoding = UTF8 } }
      { \keys_set:nn { zhnum / options } { encoding = GBK } }
  }
```

⟨/package⟩

4 中文数字配置文件

⟨*config⟩

```
⟨*!big5⟩
\zhnum_set_digits_map:nnn { minus } { simp } { 负 }
\zhnum_set_digits_map:nnn { minus } { trad } { 負 }
⟨/!big5⟩
⟨*big5⟩
\zhnum_set_digits_map:nn { minus } { 負 }
⟨/big5⟩
\zhnum_set_digits_map:nn { 0 }     { 零 }
⟨*!big5⟩
\zhnum_set_digits_map:nn { null }  { 〇 }
⟨/!big5⟩
⟨*big5⟩
\zhnum_set_digits_map:nn { null }  { ○ }
⟨/big5⟩
\zhnum_set_digits_map:nn { 1 }     { 一 }
\zhnum_set_digits_map:nn { 2 }     { 二 }
\zhnum_set_digits_map:nn { 3 }     { 三 }
\zhnum_set_digits_map:nn { 4 }     { 四 }
\zhnum_set_digits_map:nn { 5 }     { 五 }
\zhnum_set_digits_map:nn { 6 }     { 六 }
\zhnum_set_digits_map:nn { 7 }     { 七 }
\zhnum_set_digits_map:nn { 8 }     { 八 }
\zhnum_set_digits_map:nn { 9 }     { 九 }
\zhnum_set_digits_map:nn { 10 }    { 十 }
\zhnum_set_digits_map:nn { 100 }   { 百 }
\zhnum_set_digits_map:nn { 1000 }  { 千 }
\zhnum_set_digits_map:nn { 20 }    { 廿 }
\zhnum_set_digits_map:nn { 30 }    { 卅 }
\zhnum_set_digits_map:nn { 40 }    { 卌 }
\zhnum_set_digits_map:nn { 200 }   { 皕 }
⟨*!big5⟩
\zhnum_set_digits_map:nnn { dot } { simp } { 点 }
\zhnum_set_digits_map:nnn { dot } { trad } { 點 }
⟨/!big5⟩
⟨*big5⟩
\zhnum_set_digits_map:nn { dot }   { 點 }
⟨/big5⟩
\zhnum_set_digits_map:nn { and }   { 又 }
\zhnum_set_digits_map:nn { parts } { 分之 }
\zhnum_set_digits_map:nn { s0 }    { }
⟨*!big5⟩
\zhnum_set_digits_map:nnn { s1 }    { simp } { 万 }
\zhnum_set_digits_map:nnn { s1 }    { trad } { 萬 }
\zhnum_set_digits_map:nnn { s2 }    { simp } { 亿 }
\zhnum_set_digits_map:nnn { s2 }    { trad } { 億 }
⟨/!big5⟩
⟨*big5⟩
\zhnum_set_digits_map:nn { s1 }    { 萬 }
\zhnum_set_digits_map:nn { s2 }    { 億 }
⟨/big5⟩
\zhnum_set_digits_map:nn { s3 }    { 兆 }
\zhnum_set_digits_map:nn { s4 }    { 京 }
\zhnum_set_digits_map:nn { s5 }    { 垓 }
\zhnum_set_digits_map:nn { s6 }    { 秭 }
\zhnum_set_digits_map:nn { s7 }    { 穰 }
```

```
⟨*!big5⟩
\zhnum_set_digits_map:nnn { s8 }    { simp } { 沟 }
\zhnum_set_digits_map:nnn { s8 }    { trad } { 溝 }
\zhnum_set_digits_map:nnn { s9 }    { simp } { 涧 }
\zhnum_set_digits_map:nnn { s9 }    { trad } { 澗 }
⟨/!big5⟩
⟨*big5⟩
\zhnum_set_digits_map:nn { s8 }    { 溝 }
\zhnum_set_digits_map:nn { s9 }    { 澗 }
⟨/big5⟩
\zhnum_set_digits_map:nn { s10 }   { 正 }
⟨*!big5⟩
\zhnum_set_digits_map:nnn { s11 }   { simp } { 载 }
\zhnum_set_digits_map:nnn { s11 }   { trad } { 載 }
⟨/!big5⟩
⟨*big5⟩
\zhnum_set_digits_map:nn { s11 }   { 載 }
⟨/big5⟩
\zhnum_set_digits_map:nn { year }  { 年 }
\zhnum_set_digits_map:nn { month } { 月 }
\zhnum_set_digits_map:nn { day }   { 日 }
⟨*!big5⟩
\zhnum_set_digits_map:nnn { hour }  { simp } { 时 }
\zhnum_set_digits_map:nnn { hour }  { trad } { 時 }
⟨/!big5⟩
⟨*big5⟩
\zhnum_set_digits_map:nn { hour }  { 時 }
⟨/big5⟩
\zhnum_set_digits_map:nn { minute }  { 分 }
\zhnum_set_digits_map:nn { weekday } { 星期 }
\zhnum_set_financial_map:nn { null } { 零 }
\zhnum_set_financial_map:nn { 0 }    { 零 }
\zhnum_set_financial_map:nn { 1 }    { 壹 }
\zhnum_set_financial_map:nn { 2 }    { 貳 }
⟨*!big5⟩
\zhnum_set_financial_map:nnn { 3 }  { simp } { 叁 }
\zhnum_set_financial_map:nnn { 3 }  { trad } { 叁 }
⟨/!big5⟩
⟨*big5⟩
\zhnum_set_financial_map:nn { 3 }    { 參 }
⟨/big5⟩
\zhnum_set_financial_map:nn { 4 }    { 肆 }
\zhnum_set_financial_map:nn { 5 }    { 伍 }
⟨*!big5⟩
\zhnum_set_financial_map:nnn { 6 }  { simp } { 陆 }
\zhnum_set_financial_map:nnn { 6 }  { trad } { 陸 }
⟨/!big5⟩
⟨*big5⟩
\zhnum_set_financial_map:nn { 6 }    { 陸 }
⟨/big5⟩
\zhnum_set_financial_map:nn { 7 }    { 柒 }
\zhnum_set_financial_map:nn { 8 }    { 捌 }
\zhnum_set_financial_map:nn { 9 }    { 玖 }
\zhnum_set_financial_map:nn { 10 }   { 拾 }
\zhnum_set_financial_map:nn { 100 }  { 佰 }
\zhnum_set_financial_map:nn { 1000 } { 仟 }
⟨/config⟩
```

5 代码索引

斜体的数字表示对应项说明所在的页码，下划线的数字表示定义所在的代码行号，而直立体的数字表示对应项使用时所在的行号。

U

Z