The zahl2string package*

Jonathan Sauer
jonathan.sauer@gmx.de

2004/11/25

Abstract

This file describes the zahl2string package that provides macros for formatting numbers as german words, i.e. '1' gets formatted as 'eins'.

Contents

1 Introduction

Sometimes, for example when formatting a chapter title, you do not want to say 'Kapitel 1' ('Chapter 1') but 'Kapitel eins' ('Chapter one') or 'Erstes Kapitel'

*This document corresponds to zahl2string.sty v1.2.1, dated 2004/11/25.

('First chapter'). This package provides macros to format a LaTeX-counter or more generally speaking a number as a german word.

2 Description of the macros

All macros format numbers in the range 0 to 999,999,999. Larger numbers are formatted as an arabic number, smaller numbers are formatted as zero. This can be changed using the package options described on page 4.

2.1 LaTeX macros

The four macros `\numstring`, `\Numstring`, `\ordstring` and `\Ordstring` are macros taking a LaTeX-counter as their only argument.

`\numstring` Usage: `\numstring {`⟨*LaTeX counter*⟩`}`.
Formats a LaTeX counter in the range of 0 to 999,999,999 as a word.

`\Numstring` Usage: `\Numstring {`⟨*LaTeX counter*⟩`}`.
Formats a LaTeX counter in the range of 0 to 999,999,999 as words, where the first letter is a capital letter.

`\ordstring` Usage: `\ordstring {`⟨*LaTeX counter*⟩`}`.
Formats a LaTeX counter in the range of 0 to 999,999,999 as an ordinal word.

`\Ordstring` Usage: `\Ordstring {`⟨*LaTeX counter*⟩`}`.
Formats a LaTeX counter in the range of 0 to 999,999,999 as an ordinal word, where the first letter is a capital letter.

2.2 General macros

The four macros `\@numstring`, `\@Numstring`, `\@ordstring` and `\@Ordstring` as well as their aliases `\numstr`, `\Numstr`, `\ordstr` and `\Ordstr` are macros taking a number or a TeX count register as their only argument.

`\@numstring` `\numstr` Usage: `\@numstring {`⟨*number*⟩`}` / `\numstr {`⟨*number*⟩`}`.
Formats a number or a TeX count register in the range of 0 to 999,999,999 as words.

`\@Numstring` `\Numstr` Usage: `\@Numstring {`⟨*number*⟩`}` / `\Numstr {`⟨*number*⟩`}`.
Formats a number or a TeX count register in the range of 0 to 999,999,999 as words, where the first letter is a capital letter.

`\@ordstring` `\ordstr` Usage: `\@ordstring {`⟨*number*⟩`}` / `\ordstr {`⟨*number*⟩`}`.
Formats a number or a TeX count register in the range of 0 to 999,999,999 as an ordinal word.

`\@Ordstring` `\Ordstr` Usage: `\@Ordstring {`⟨*number*⟩`}` / `\Ordstr {`⟨*number*⟩`}`.
Formats a number or a TeX count register in the range of 0 to 999,999,999 as an ordinal word, where the first letter is a capital letter.

3 Examples

Some examples using `\numstring`, `\numstr` and `\@numstring`:

0 $\Longrightarrow$ null
7 $\Longrightarrow$ sieben
13 $\Longrightarrow$ dreizehn
23 $\Longrightarrow$ dreiundzwanzig
42 $\Longrightarrow$ zweiundvierzig
99 $\Longrightarrow$ neunundneunzig
127 $\Longrightarrow$ hundertsiebenundzwanzig
999 $\Longrightarrow$ neunhundertneunundneunzig
1000 $\Longrightarrow$ tausend
1001 $\Longrightarrow$ tausendeins
2004 $\Longrightarrow$ zweitausendvier
2017 $\Longrightarrow$ zweitausendsiebzehn
2029 $\Longrightarrow$ zweitausendneunundzwanzig
9999 $\Longrightarrow$ neuntausendneunhundertneunundneunzig
10000 $\Longrightarrow$ zehntausend
101101 $\Longrightarrow$ hunderteintausendeinhunderteins
999999 $\Longrightarrow$ neunhundertneunundneunzigtausendneunhundertneunundneunzig
1000000 $\Longrightarrow$ eine Million
1234567 $\Longrightarrow$ eine Million zweihundertvierunddreißigtausendfünfhundertsiebenundsechzig
123456789 $\Longrightarrow$ hundertdreiundzwanzig Millionen vierhundertsechsundfünfzigtausendsiebenhundertneunundachtzig
101101101 $\Longrightarrow$ hunderteins Millionen einhunderteintausendeinhunderteins
999999999 $\Longrightarrow$ neunhundertneunundneunzig Millionen neunhundertneunundneunzigtausendneunhundertneunundneunzig

Some examples using `\ordstring`, `\ordstr` and `\@ordstring`:
0 $\Longrightarrow$ nullte
7 $\Longrightarrow$ siebte
13 $\Longrightarrow$ dreizehnte
23 $\Longrightarrow$ dreiundzwanzigste
42 $\Longrightarrow$ zweiundvierzigste
99 $\Longrightarrow$ neunundneunzigste
127 $\Longrightarrow$ hundertsiebenundzwanzigste
999 $\Longrightarrow$ neunhundertneunundneunzigste
1000 $\Longrightarrow$ tausendste
1001 $\Longrightarrow$ tausenderste
2004 $\Longrightarrow$ zweitausendvierte
2017 $\Longrightarrow$ zweitausendsiebzehnte
2029 $\Longrightarrow$ zweitausendneunundzwanzigste
9999 $\Longrightarrow$ neuntausendneunhundertneunundneunzigste
10000 $\Longrightarrow$ zehntausendste
101101 $\Longrightarrow$ hunderteintausendeinhunderterste
999999 $\Longrightarrow$ neunhundertneunundneunzigtausendneunhundertneunundneunzigste
1000000 $\Longrightarrow$ eine Millionste

1234567 $\Longrightarrow$ eine Million zweihundertvierunddreißigtausendfünfhundertsiebenundsechzigste

123456789 $\Longrightarrow$ hundertdreiundzwanzig Millionen vierhundertsechsundfünfzigtausendsiebenhundertneunundachtzigste

101101101 $\Longrightarrow$ hunderteins Millionen einhunderteintausendeinhunderterste

999999999 $\Longrightarrow$ neunhundertneunundneunzig Millionen neunhundertneunundneunzigtausendneunhundertneunundneunzigste

Formatting the current page number (a LaTeX counter) results in: Dies ist Seite vier (`\numstring{page}`). Dies ist die vierte Seite (`\ordstring{page}`). Seite: Vier (`\Numstring{page}`). Vierte Seite (`\Ordstring{page}`).

4 Options

The package has the following option:

showrangeerrors If a number larger than 999,999,999 is to be formatted, normally the number is not formatted as words, but using arabic digits. This option changes this behaviour to display an error instead, thus notifying you when you format too large a number

5 Creating your own formatting

You can modify output of the zahl2string macros in a limited way by providing your own macros for formatting the numbers between '1' and '19. *But note* that if you simply want to add something to the suffix, , i.e. if you want to format numbers with the suffix 'tens' ('erstens', 'zweitens' ...), then you can simply say `\ordstr{`⟨*number*⟩`}ns`, resulting in i.e. 'zweiundvierzigstens'.

If on the other hand you want to create a more complicated formatting, then you have to do the following:

- Create a macro for formatting the numbers '1' to '19', i.e. `\my@neunzehnte`. See the predefined macros in section 7.3.2 on page 12, `\ns@neunzehn`, `\ns@neunzehns`, `\ns@neunzehne` and `\ns@neunzehnord` for examples and notes.

  This macro has one parameter, the number (up to two digits, in the range of '0' to '19'. '0' must expand to the generic suffix, i.e. 'stens'; '19' to '19' simply format the number.

  Do not forget to insert discretionary hyphens using `\-`, or hyphenation will not be perfect!

- Create another macro you want to call, i.e. `\mynumstring`. This macro takes one parameter, the number, and calls `\ns@numstr` using the following parameters:

1. The number to be formatted (the parameter to `\mynumstring`).
2. The macro for formatting numbers '1' to '19', i.e. `\my@neunzehnte`.
3. The value that represents the number zero, i.e. 'nulltens'.
4. The suffix for numbers larger than 999,999,999, i.e. 'tens'.

- Call the macro `\mynumstring` with the number to format.

If you want to create a macro to format a LaTeX-counter, create an additional macro, i.e. `\myLnumstring`, that calls `\mynumstring` by saying:

```
\newcommand{\myLnumstring}[1]{%
 \expandafter\mynumstring\csname c@#1\endcsname%
}
```

See also the notes and predefined macros in section 7.3.2 on page 12.

6 Notes/Limitations

- Ordinal numbers larger than 999999 do not look that good, as i.e. 1000000 gets formatted as 'eine Millionste' instead of 'einmillionste'.

7 Implementation

7.1 Main macros

`\numstring` Usage: `\numstring {`⟨*LaTeX counter*⟩`}`.
Formats a LaTeX counter in the range of 0 to 999,999,999 as words.

```
\newcommand{\numstring}[1]{%
  \expandafter\@numstring\csname c@#1\endcsname%
}
```

`\Numstring` Usage: `\Numstring {`⟨*LaTeX counter*⟩`}`.
Formats a LaTeX counter in the range of 0 to 999,999,999 as words. The first letter is uppercase.

```
\newcommand{\Numstring}[1]{%
  \expandafter\@Numstring\csname c@#1\endcsname%
}
```

`\ordstring` Usage: `\ordstring {`⟨*LaTeX counter*⟩`}`.
Formats a LaTeX counter in the range of 0 to 999,999,999 as an ordinal word.

```
\newcommand{\ordstring}[1]{%
  \expandafter\@ordstring\csname c@#1\endcsname%
}
```

`\Ordstring` Usage: `\Ordstring {`⟨*LaTeX counter*⟩`}`.

Formats a LaTeX counter in the range of 0 to 999,999,999 as an ordinal word. The first letter is uppercase.

```
\newcommand{\Ordstring}[1]{%
  \expandafter\@Ordstring\csname c@#1\endcsname%
}
```

`\@numstring` Usage: `\@numstring {`⟨*number or TeX count register*⟩`}`.

Formats a number or a TeX count register in the range of 0 to 999,999,999 as words.

```
\newcommand{\@numstring}[1]{%
  \ns@numstr{#1}\ns@neunzehns{null}{}%
}
```

`\@Numstring` Usage: `\@Numstring {`⟨*number or TeX count register*⟩`}`.

Formats a number or a TeX count register in the range of 0 to 999,999,999 as words. The first letter is uppercase.

```
\newcommand\@Numstring[1]{%
  \expandafter\@@Numstring\expandafter{\number#1}%
}
```

`\@@Numstring` Support macro for `\@Numstring` to make `\@Numstring` robust.

```
\DeclareRobustCommand{\@@Numstring}[1]{%
  \protected@edef\@tempa{\@numstring{#1}}%
  \expandafter\MakeUppercase\@tempa%
}
```

`\@ordstring` Usage: `\@ordstring {`⟨*number or TeX count register*⟩`}`.

Formats a number or a TeX count register in the range of 0 to 999,999,999 as an ordinalword, i.e. ‘erste’, ‘zweite’ et cetera.

```
\newcommand{\@ordstring}[1]{%
  \ns@numstr{#1}\ns@neunzehnord{null\-te}{te}%
}
```

`\@Ordstring` Usage: `\@Ordstring {`⟨*number or TeX count register*⟩`}`.

Formats a number or a TeX count register in the range of 0 to 999,999,999 as an ordinalword, i.e. ‘Erste’, ‘Zweite’ et cetera. The first letter is uppercase.

```
\newcommand{\@Ordstring}[1]{%
  \expandafter\@@Ordstring\expandafter{\number#1}%
}
```

`\@@Ordstring` Support macro for `\@Ordstring` to make `\@Ordstring` robust.

```
\DeclareRobustCommand{\@@Ordstring}[1]{%
  \protected@edef\@tempa{\@ordstring{#1}}%
  \expandafter\MakeUppercase\@tempa%
}
```

We provide public aliases for the macros. The macros beginning with `@` are still necessary in order to be able to format the page number as a string. (see ltpageno.dtx)

```
\let\numstr\@numstring%
\let\Numstr\@Numstring%
\let\ordstr\@ordstring%
\let\Ordstr\@Ordstring%
```

7.2 Option processing

```
\DeclareOption{publicnumstr}{%
  \PackageWarning{zahl2string}{Option 'publicnumstr' is %
    deprecated and will be removed in version 1.3}%
}
```

`\ns@numoutofrange` Formats a number that is too large to be formatted as words.

Usage: `\ns@numoutofrange {`⟨*number*⟩`} {`⟨*suffix*⟩`}`.

This macro is redefined to show an error message using the package option `showrangeerrors`.

```
\def\ns@numoutofrange#1#2{%
  \number#1#2%
}

\DeclareOption{showrangeerrors}{%
  \def\ns@numoutofrange#1#2{%
    \PackageError{zahl2string}{The number '#1' is too large %
      to be formatted using zahl2string}{The largest possible %
      number is 999,999,999.}
  }%
}

\ProcessOptions\relax
```

7.3 Internal macros

7.3.1 Splitting and formatting a number

`\ns@numstr` Base macro for formatting a number.

Usage: `\ns@numstr {`⟨*number*⟩`} {`⟨*macro*⟩`} {`⟨*nullvalue*⟩`} {`⟨*suffix*⟩`}`, where ⟨*macro*⟩ is the macro to use for the numbers between `1` and `19`, as these require some special treatment, ⟨*nullvalue*⟩ is the value this macro expands to when ⟨*number*⟩ is zero, and ⟨*suffix*⟩ is text added as a suffix to a number larger than 999,999,999.

```
\def\ns@numstr#1#2#3#4{%
  \ifnum\number#1<\@ne%
    #3%
  \else\ifnum\number#1<1000000000 %
    \expandafter\ns@numstring\expandafter{\number#1}#2%
  \else%
```

```
    \ns@numoutofrange{#1}{#4}%
  \fi\fi%
}
```

`\ns@numstring` Formats a number as words.

Usage: `\ns@numstring {`⟨*number*⟩`} {`⟨*macro*⟩`}`, where ⟨*macro*⟩ is the macro to use for the numbers between `1` and `19`, as these require some special treatment.

Note: ⟨number⟩ must be a real number consisting of digits in the range 0 to 9! It must not be a TEX count register!

How does this work? Modulo operations are not trivial in TEX as in order to achieve a mod b you have to calculate $a - (a \div b) \times b$. This is complicated and also not expandable, so another solution has to be found.

TEX's capabilities of parsing text using macro arguments are fairly strong, so why not use them? It would be much easier if it would be possible of defining a macro with, say, six parameters, where each parameter is one digit of the number to be formatted. Then it would be possible to directly access each digit (or several digits combined by grouping several parameters) without having to perform lengthy modulo calculations.

Adding leading zeros to a number is easily done by comparing it using `\ifnum` and adding zeros if the number is too small. However, the macro must not receive the `\ifnum` et.al. tokens as a parameter, but the result of the expansion, that is the number with leading zeros.

`\expandafter` would not suffice, as it expands a macro only once, not fully. `\edef` would accomplish the task at hand, however `\edef` is not fully expandable.

So what do we do? We take advantage of the fact that when expanding an `\ifcase`, TEX goes on expanding until it has made sure that the number for the `\ifcase` is complete. So immediately after the `\ifcase` we launch into several nested `\ifnum`s, which TEX expands dutifully in order to determine the number to use.

What these nested `\ifnum`s do is the following: Depending on the length of ⟨*number*⟩, they expand to a digit between one and nine, one being the digit if ⟨*number*⟩ is less than 10 and nine being the digit if ⟨*number*⟩ is larger than 99,999,999.

Then TEX uses this digit between one and nine to jump to the appropriate part of the `\ifcase`-clause. There ⟨*number*⟩ is prefixed with the necessary amount of zeros to result in a number exactly nine digits long: For the digit one (resulting from the nested `\ifnum`s), eight leading zeros have to be prefixed, as ⟨*number*⟩ is only one digit long, for the digit two, seven zeros are prefixed, as ⟨*number*⟩ is less than 100 (but more than 9) et cetera, until for digit nine no zeros have to be prefixed, as ⟨*number*⟩ is already nine digits long.

Let's have an example: Suppose ⟨*number*⟩ is 42. Then the result of the nested `\ifnum`s is 2, as ⟨*number*⟩ is not less than 10 but less than 100 (the second `\ifnum` is true). This 2 leads to jumping to part after the second `\or` (before the first `\or` is the part for the number 0, in this case left blank, and after the first `\or` for the number 1), which is `0000000#1`. `#1` is 42. So the result is: `000000042`, a number prefixed with leading zeros and exactly nine digits long.

Another example: Suppose ⟨*number*⟩ is 12,345,678. Then the result of the nested `\ifnum`s is 8, as the eigths `\ifnum` is true (less than 100,000,000 but not less than 10,000,000). Then only one zero is prefixed, resulting in `012345678`.

However, we are not finished yet, as TEX does not expand further. So we are left (picking up the second example above) with this: `012345678\or12345678\or` (the second incarnation of `12345678` is due to the ninth part of the `\ifcase` clause, `\or#1`).

We do not need all this `\or` baggage, we only want the number. But TEX is good at matching text using macros with delimited parameters, so we just define `\ns@@numstring` in a way that gobbles up the first `\or` and whatever follows.

And we are done: We have a number padded perfectly with leading zeros to a length of nine digits!

Implementation note In Version 1.0, we used `\csname` ... `\endcsname` to add the leading zeros, as `\csname` ... `\endcsname` expands everything inbetween until only unexpandable tokens remain, in this case digits (the `\ifnum`s are expanded). Afterwards, the resulting control sequence was converted into separate tokens using `\string` and finally, the backslash at the beginning was gobbled by `\ns@@numstring` as its first (and unused) argument.

Unfortunately, this had its price: For every number we formatted using `\ns@numstring` and that had not been formatted before, a new entry was inserted into the hash table TEX uses to store all control sequences. So if you had a document where you formatted a lot of numbers this way, you would run out of hash table space, and TEX would complain. (The number of hash table entries used is indicated in the `log`-file as 'multiletter control sequences'.)

This was less than optimal, so we changed the implementation to this `\ifcase`-`\ifnum` construct.

```
\def\ns@numstring#1#2{%
  \expandafter\ns@@numstring%
    \ifcase%
      \ifnum#1<10 1%
```

Why are the constants predefined by the LATEX-kernel used instead of numbers? Because they save tokens: `1000` are four tokens, `\@m` is only one.

```
      \else\ifnum#1<100 2%
      \else\ifnum#1<\@m 3%
      \else\ifnum#1<\@M 4%
      \else\ifnum#1<100000 5%
      \else\ifnum#1<1000000 6%
      \else\ifnum#1<10000000 7%
      \else\ifnum#1<100000000 8%
      \else9%
      \fi\fi\fi\fi\fi\fi\fi\fi %
    \or00000000#1% case 1: Add 8 leading zeros
    \or0000000#1%  case 2: Add 7 leading zeros
    \or000000#1%   case 3: Add 6 leading zeros
    \or00000#1%    case 4: Add 5 leading zeros
```

```
    \or0000#1%      case 5: Add 4 leading zeros
    \or000#1%       case 6: Add 3 leading zeros
    \or00#1%        case 7: Add 2 leading zeros
    \or0#1%         case 8: Add 1 leading zero
    \or#1%          case 9: Add no leading zeros
```

The next `\or` is only necessary because `\ns@@numstring` needs an `\or` as a delimiter of the number:

```
    \or%
```

The last parameter to `\ns@@numstring` is ⟨*macro*⟩; we delimit it using `\@nil`:

```
  \@nil#2%
```

Finally we end the `\ifcase` (note that this is *after* the number has been formatted):

```
  \fi%
}
```

`\ns@@numstring` Expands to a number in words between `1` and `999,999,999`.

Usage: `\ns@@numstring {`⟨*9. digit*⟩`} {`⟨*8. digit*⟩`} {`⟨*7. digit*⟩`} {`⟨*6. digit*⟩`} {`⟨*5. digit*⟩`} {`⟨*4. digit*⟩`} {`⟨*3rd - 1st digit*⟩`} {`⟨*(ignored)*⟩`} {`⟨*macro*⟩`}`, where ⟨*macro*⟩ is, as is the case with `\ns@numstring`, the macro to use for the numbers between `1` and `19`, as these require some special treatment.

`\or` and the following `#8` (⟨*ignored*⟩) gobble up whatever was left from the expansion of `\ifcase` in `\ns@numstring`. `\@nil` acts as a delimiter for the last parameter, ⟨*macro*⟩.

```
\def\ns@@numstring#1#2#3#4#5#6#7\or#8\@nil#9{%
  \ifnum#1#2#3>\z@
    \ns@million#1#2#3%
```

We insert a space if a number follows:

```
    \ifnum#4#5#6>\z@\space\fi%
  \fi%
  \ifnum#4#5#6>\z@%
    \ns@hundred#4#5#6{#1#2#3}{#4#5}\ns@neunzehn%
```

If there has been a number larger than one before the ‘tausend’, insert a discretionary hyphen before:

```
    \ifnum#4#5#6>\@ne\-\fi%
    tau\-send%
```

If there will be a number after the ‘tausend’, insert a discretionary hyphen after:

```
    \ifnum#7>\z@\-\fi%
  \fi%
  \ns@hundred#7{#4#5#6}1#9%
}
```

`\ns@million` Expands to millions.

Usage: `\ns@million {`⟨*third digit*⟩`} {`⟨*second digit*⟩`} {`⟨*first digit*⟩`}`.

```
\def\ns@million#1#2#3{%
  \ifnum#1#2#3=\@ne%
    \ns@hundred#1#2#301\ns@neunzehne%
    \space%
    Mil\-lion%
  \else%
    \ns@hundred#1#2#301\ns@neunzehns%
    \space%
    Mil\-lio\-nen%
  \fi%
}
```

`\ns@hundred` Expands to a number in words between `1` and `100`.

Usage: `\ns@hundred {`⟨*third digit*⟩`} {`⟨*second digit*⟩`} {`⟨*first digit*⟩`} {`⟨*shownumber*⟩`} {`⟨*showone*⟩`} {`⟨*macro*⟩`}`.

⟨*shownumber*⟩ defines if the number before the `hundert` ('hundred') should be shown, i.e. `einhundert` ('onehundred') instead of `hundert`. `0` is `false`, everything else `true`. Can contain more than one digit.

⟨*showone*⟩ defines if ⟨*macro*⟩ should be called for the number `1`. `0` is `false`, everything else `true`. Can contain more than one digit.

What does all this code do? First the third digit ('hundred'), contained in `#1`, is expanded – if it is not zero. However, there is a catch: If the third digit is one, this digit is only included in the result of this macro if `#4` says so. The reason is that normally you would say `hunderteins` ('101') instead of `einhunderteins` – but not if there is a fourth digit. Then the digit has to be included in the output of the macro, i.e. `tausendeinhunderteins`

So we want to include the digit in the output of the macro, if `#3` is larger than one *or* `#4` is larger than zero. We could use two `\ifnum`s to accomplish this, but it can be combined into a single `\ifnum`, saving tokens and time:

We check if `#4#1` is larger than one. That means that if `#1`, the third digit, is larger than one, it is included in the output. But that also means that if `#4` is not zero, `#4#1` is always at least `10`, which is also larger than one and exactly what we want.

After successfully processing the third digit, the remaining last two digits (`#2` and `#3`) are not really complicated anymore. We perform some special treatment of the numbers between `1` and `19` as these numbers are not constructed systematically. Here we have to perform a check similar to the one performed for the third digit: We have to check if we have to output the number if it is `1`, using the same trick for a logical `or` as before, only this time with `#5`, `#2` and `#3`.

The macro to output `1` to `19` is parametrized as `#6` in order to be able to use different macros for normal numbers and ordinal numbers.

If the last two digits are larger than `19`, we first output the third digit using `\ns@neunzehn`, followed by `und` ('and') and then the second digit. Here we don't have to use the parametrized macro as before, as the ordinal suffix is appended at the end of the number, not inbetween.

But where to get the suffix from? We could pass it as a macro, but that would be tedious. So we simply define the macro to output the numbers `1` to `19` (`\ns@neunzehns` for normal numbers and `\ns@neunzehnord` for ordinal numbers) to output the suffix if called with `0` as its parameter.

And we are done!

```
\def\ns@hundred#1#2#3#4#5#6{%
```

We expand the third digit:

```
  \ifnum#1>\z@%
```

Logical `OR` hidden in `#4#1` (see above):

```
    \ifnum#4#1>\@ne\ns@neunzehn#1\-\fi%
    hun\-dert%
```

We insert a discretionary hyphen, if a number follows:

```
    \ifnum#2#3>\z@\-\fi%
  \fi%
```

We expand the first an second digit:

```
  \ifnum#2#3<20 %
```

Again: Logical `OR` in `#5#2#3`:

```
    \ifnum#5#2#3>\@ne#6{#2#3}\fi%
  \else%
    \ifnum#3>\z@\ns@neunzehn#3\-und\-\fi%
    \ns@neunzig#2%
    #60%
  \fi%
}
```

7.3.2 Macros for formatting '1' to '19'

Very important: The position for `0` must expand to the suffix of the number (*any number*), see the explanations for `\ns@hundred` above. If it expands to a text (as opposed to `\@empty`), this text must be prefixed by a discretionary hyphen!

`\ns@neunzehn` Expands to `ein` ('1') to `neunzehn` ('19').

```
\def\ns@neunzehn#1{%
  \ifcase#1\@empty\or ein\or zwei\or drei\or vier\or f\"unf\or sechs\or%
    sie\-ben\or acht\or neun\or zehn\or elf\or zw\"olf\or drei\-zehn\or%
    vier\-zehn\or f\"unf\-zehn\or sech\-zehn\or sieb\-zehn\or%
    acht\-zehn\or neun\-zehn\fi%
}
```

`\ns@neunzehns` Expands to `eins` ('1') to `neunzehn` ('19'). `0` expands to `\@empty`.

Why `\@empty` instead of nothing? Because then TEX would insert a `\relax` before the first `\or` in order to finish expansion of the number. This `\relax` would remain in the output and would stay there even when `\edef`ed, whereas `\@empty` expands to nothing.

This is necessary for `\Numstring` and `\Ordstring`, as they convert the first token of the `\edef`ed result of `\numstring` and `\ordstring` to uppercase, and they need this token to be the first letter of the number, not `\relax`.

```
\def\ns@neunzehns#1{%
  \ifcase#1\@empty\or eins\else\ns@neunzehn{#1}\fi%
}
```

`\ns@neunzehnord` Expands to `erste` (‘1st’) to `neunzehnte` (‘19th’). `0` expands to `ste`.

```
\def\ns@neunzehnord#1{%
  \ifcase#1\-ste\or er\-ste\or zwei\-te\or drit\-te\or vier\-te\or%
    f\"unf\-te\or sech\-ste\or sieb\-te\or ach\-te\or neun\-te\or%
    zehn\-te\or elf\-te\or zw\"olf\-te\or drei\-zehn\-te\or%
    vier\-zehn\-te\or f\"unf\-zehn\-te\or sech\-zehn\-te\or%
    sieb\-zehn\-te\or acht\-zehn\-te\or neun\-zehn\-te\fi%
}
```

`\ns@neunzig` Expands to `zwanzig` (twenty) to `neunzig` (ninety) in steps of ten.

```
\def\ns@neunzig#1{%
  \ifcase#1\or\or zwan\-zig\or drei\ss ig\or vier\-zig\or%
    f\"unf\-zig\or sech\-zig\or sieb\-zig\or acht\-zig\or%
    neun\-zig\fi%
}
```

Change History

Index

Numbers written in italic refer to the page where the corresponding entry is described, the ones underlined to the code line of the definition, the rest to the code lines where the entry is used.

N

O