

# fmtcount.sty: Displaying the Values of L<sup>A</sup>T<sub>E</sub>X Counters

Nicola L.C. Talbot

Vincent Belaïche

[www.dickimaw-books.com](http://www.dickimaw-books.com)

2017-12-26 (version 3.05)

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Available Commands</b>	<b>2</b>
<b>3</b>	<b>Package Options</b>	<b>8</b>
<b>4</b>	<b>Multilingual Support</b>	<b>8</b>
4.1	Options for setting ordinal ending position raise/level . . . . .	9
4.2	Options for French . . . . .	10
4.3	Prefixes . . . . .	14
<b>5</b>	<b>Configuration File <code>fmtcount.cfg</code></b>	<b>14</b>
<b>6</b>	<b>LaTeX2HTML style</b>	<b>15</b>
<b>7</b>	<b>Acknowledgements</b>	<b>15</b>
<b>8</b>	<b>Troubleshooting</b>	<b>15</b>
<b>9</b>	<b>The Code</b>	<b>15</b>
9.1	Language definition files . . . . .	15
9.1.1	<code>fc-american.def</code> . . . . .	15
9.1.2	<code>fc-brazilian.def</code> . . . . .	16
9.1.3	<code>fc-british.def</code> . . . . .	18
9.1.4	<code>fc-english.def</code> . . . . .	19
9.1.5	<code>fc-francais.def</code> . . . . .	29
9.1.6	<code>fc-french.def</code> . . . . .	29
9.1.7	<code>fc-frenchb.def</code> . . . . .	61
9.1.8	<code>fc-german.def</code> . . . . .	61
9.1.9	<code>fc-germanb.def</code> . . . . .	71

9.1.10 <code>fc-italian</code>	72
9.1.11 <code>fc-ngerman.def</code>	73
9.1.12 <code>fc-ngermanb.def</code>	73
9.1.13 <code>fc-portuges.def</code>	74
9.1.14 <code>fc-portuguese.def</code>	89
9.1.15 <code>fc-spanish.def</code>	89
9.1.16 <code>fc-UKenglish.def</code>	106
9.1.17 <code>fc-USenglish.def</code>	107
9.2 <code>fcnumparser.sty</code>	107
9.3 <code>fcprefix.sty</code>	118
9.4 <code>fmtcount.sty</code>	128
9.4.1 Multilingual Definitions	153

## 1 Introduction

The `fmtcount` package provides commands to display the values of  $\LaTeX$  counters in a variety of formats. It also provides equivalent commands for actual numbers rather than counter names. Limited multilingual support is available. Currently, there is only support for English, French (including Belgian and Swiss variations), Spanish, Portuguese, German and Italian.

## 2 Available Commands

The commands can be divided into two categories: those that take the name of a counter as the argument, and those that take a number as the argument.

`\ordinal` `\ordinal{<counter>}[<gender>]`

This will print the value of a  $\LaTeX$  counter `<counter>` as an ordinal, where the macro

`\fmtord` `\fmtord{<text>}`

is used to format the st, nd, rd, th bit. By default the ordinal is formatted as a superscript, if the package option `level` is used, it is level with the text. For example, if the current section is 3, then `\ordinal{section}` will produce the output: 3<sup>rd</sup>. Note that the optional argument `<gender>` occurs *at the end*. This argument may only take one of the following values: `m` (masculine), `f` (feminine) or `n` (neuter.) If `<gender>` is omitted, or if the given gender has no meaning in the current language, `m` is assumed.

### Notes:

1. the memoir class also defines a command called `\ordinal` which takes a number as an argument instead of a counter. In order to overcome this incompatibility, if you want to use the `fmtcount` package with the memoir class you should use

`\FCordinal` `\FCordinal`

to access `fmtcount`'s version of `\ordinal`, and use `\ordinal` to use `memoir`'s version of that command.

2. As with all commands which have an optional argument as the last argument, if the optional argument is omitted, any spaces following the final argument will be ignored. Whereas, if the optional argument is present, any spaces following the optional argument won't be ignored. so `\ordinal{section} !` will produce: 3<sup>rd</sup>! whereas `\ordinal{section}[m] !` will produce: 3<sup>rd</sup>!

The commands below only work for numbers in the range 0 to 99999.

`\ordinalnum` `\ordinalnum{<n>}[<gender>]`

This is like `\ordinal` but takes an actual number rather than a counter as the argument. For example: `\ordinalnum{3}` will produce: 3<sup>rd</sup>.

`\numberstring` `\numberstring{<counter>}[<gender>]`

This will print the value of `<counter>` as text. E.g. `\numberstring{section}` will produce: three. The optional argument is the same as that for `\ordinal`.

`\Numberstring` `\Numberstring{<counter>}[<gender>]`

This does the same as `\numberstring`, but with initial letters in uppercase. For example, `\Numberstring{section}` will produce: Three.

`\NUMBERstring` `\NUMBERstring{<counter>}[<gender>]`

This does the same as `\numberstring`, but converts the string to upper case. Note that `\MakeUppercase{\NUMBERstring{<counter>}}` doesn't work, due to the way that `\MakeUppercase` expands its argument<sup>1</sup>.

`\numberstringnum` `\numberstringnum{<n>}[<gender>]`

`\Numberstringnum` `\Numberstringnum{<n>}[<gender>]`

`\NUMBERstringnum` `\NUMBERstringnum{<n>}[<gender>]`

These macros work like `\numberstring`, `\Numberstring` and `\NUMBERstring`, respectively, but take an actual number rather than a counter as the argument. For example: `\Numberstringnum{105}` will produce: One Hundred and Five.

---

<sup>1</sup>See all the various postings to `comp.text.tex` about `\MakeUppercase`

`\ordinalstring` `\ordinalstring{<counter>}[<gender>]`

This will print the value of `<counter>` as a textual ordinal. E.g. `\ordinalstring{section}` will produce: third. The optional argument is the same as that for `\ordinal`.

`\Ordinalstring` `\Ordinalstring{<counter>}[<gender>]`

This does the same as `\ordinalstring`, but with initial letters in uppercase. For example, `\Ordinalstring{section}` will produce: Third.

`\ORDINALstring` `\ORDINALstring{<counter>}[<gender>]`

This does the same as `\ordinalstring`, but with all words in upper case (see previous note about `\MakeUppercase`).

`\ordinalstringnum` `\ordinalstringnum{<n>}[<gender>]`

`\Ordinalstringnum` `\Ordinalstringnum{<n>}[<gender>]`

`\ORDINALstringnum` `\ORDINALstringnum{<n>}[<gender>]`

These macros work like `\ordinalstring`, `\Ordinalstring` and `\ORDINALstring`, respectively, but take an actual number rather than a counter as the argument. For example, `\ordinalstringnum{3}` will produce: third.

As from version 1.09, textual representations can be stored for later use. This overcomes the problems encountered when you attempt to use one of the above commands in `\edef`.

Each of the following commands takes a label as the first argument, the other arguments are as the analogous commands above. These commands do not display anything, but store the textual representation. This can later be retrieved using

`\FMCuse` `\FMCuse{<label>}`

Note: with `\storeordinal` and `\storeordinalnum`, the only bit that doesn't get expanded is `\fmtord`. So, for example, `\storeordinalnum{mylabel}{3}` will be stored as `3\relax \fmtord{rd}`.

`\storeordinal` `\storeordinal{<label>}{<counter>}[<gender>]`

reordinalstring `\storeordinalstring{<label>}{<counter>}[<gender>]`

reOrdinalstring `\storeOrdinalstring{<label>}{<counter>}[<gender>]`

reORDINALstring `\storeORDINALstring{<label>}{<counter>}[<gender>]`

renumberstring `\storenumberstring{<label>}{<counter>}[<gender>]`

reNumberstring `\storeNumberstring{<label>}{<counter>}[<gender>]`

reNUMBERstring `\storeNUMBERstring{<label>}{<counter>}[<gender>]`

storeordinalnum `\storeordinalnum{<label>}{<number>}[<gender>]`

reordinalstringnum `\storeordinalstring{<label>}{<number>}[<gender>]`

reOrdinalstringnum `\storeOrdinalstringnum{<label>}{<number>}[<gender>]`

reORDINALstringnum `\storeORDINALstringnum{<label>}{<number>}[<gender>]`

renumberstringnum `\storenumberstring{<label>}{<number>}[<gender>]`

reNumberstringnum `\storeNumberstring{<label>}{<number>}[<gender>]`

reNUMBERstringnum `\storeNUMBERstring{<label>}{<number>}[<gender>]`

`\binary` `\binary{<counter>}`

This will print the value of *<counter>* as a binary number. E.g. `\binary{section}` will produce: 11. The declaration

`\padzeroes`

```
\padzeroes[n]
```

will ensure numbers are written to *n* digits, padding with zeroes if necessary. E.g. `\padzeroes[8]\binary{sect}` will produce: 00000011. The default value for *n* is 17.

`\binarynum`

```
\binary{n}
```

This is like `\binary` but takes an actual number rather than a counter as the argument. For example: `\binarynum{5}` will produce: 101.

The octal commands only work for values in the range 0 to 32768.

`\octal`

```
\octal{counter}
```

This will print the value of *counter* as an octal number. For example, if you have a counter called, say `mycounter`, and you set the value to 125, then `\octal{mycounter}` will produce: 177. Again, the number will be padded with zeroes if necessary, depending on whether `\padzeroes` has been used.

`\octalnum`

```
\octalnum{n}
```

This is like `\octal` but takes an actual number rather than a counter as the argument. For example: `\octalnum{125}` will produce: 177.

`\hexadecimal`

```
\hexadecimal{counter}
```

This will print the value of *counter* as a hexadecimal number. Going back to the counter used in the previous example, `\hexadecimal{mycounter}` will produce: 7d. Again, the number will be padded with zeroes if necessary, depending on whether `\padzeroes` has been used.

`\Hexadecimal`

```
\Hexadecimal{counter}
```

This does the same thing, but uses uppercase characters, e.g. `\Hexadecimal{mycounter}` will produce: 7D.

`\hexadecimalnum`

```
\hexadecimalnum{n}
```

`\Hexadecimalnum`

```
\Hexadecimalnum{n}
```

These are like `\hexadecimal` and `\Hexadecimal` but take an actual number rather than a counter as the argument. For example: `\hexadecimalnum{125}` will produce: 7d, and `\Hexadecimalnum{125}` will produce: 7D.

`\decimal`

```
\decimal{counter}
```

This is similar to `\arabic` but the number can be padded with zeroes depending on whether `\padzeroes` has been used. For example: `\padzeroes[8]\decimal{section}` will produce: 00000005.

`\decimalnum`

```
\decimalnum{<n>}
```

This is like `\decimal` but takes an actual number rather than a counter as the argument. For example: `\padzeroes[8]\decimalnum{5}` will produce: 00000005.

`\aaalph`

```
\aaalph{<counter>}
```

This will print the value of `<counter>` as: a b ... z aa bb ... zz etc. For example, `\aaalpha{mycounter}` will produce: uuuuu if `mycounter` is set to 125.

`\AAAAlph`

```
\AAAAlph{<counter>}
```

This does the same thing, but uses uppercase characters, e.g. `\AAAAlph{mycounter}` will produce: UUUUU.

`\aaalphnum`

```
\aaalphnum{<n>}
```

`\AAAAlphnum`

```
\AAAAlphnum{<n>}
```

These macros are like `\aaalph` and `\AAAAlph` but take an actual number rather than a counter as the argument. For example: `\aaalphnum{125}` will produce: uuuuu, and `\AAAAlphnum{125}` will produce: UUUUU.

The `abalph` commands described below only work for values in the range 0 to 17576.

`\abalph`

```
\abalph{<counter>}
```

This will print the value of `<counter>` as: a b ... z aa ab ... az etc. For example, `\abalpha{mycounter}` will produce: du if `mycounter` is set to 125.

`\ABAlph`

```
\ABAlph{<counter>}
```

This does the same thing, but uses uppercase characters, e.g. `\ABAlph{mycounter}` will produce: DU.

`\abalphnum`

```
\abalphnum{<n>}
```

`\ABAlphnum`

```
\ABAlphnum{<n>}
```

These macros are like `\abalph` and `\ABAlph` but take an actual number rather than a counter

as the argument. For example: `\abalphnum{125}` will produce: du, and `\ABAlphnum{125}` will produce: DU.

### 3 Package Options

The following options can be passed to this package:

`<dialect>` load language `<dialect>`, supported `<dialect>` are the same as passed to `\FCloadlang`, see 4

`raise` make ordinal st,nd,rd,th appear as superscript

`level` make ordinal st,nd,rd,th appear level with rest of text

Options `raise` and `level` can also be set using the command:

`\fmtcountsetoptions`

```
\fmtcountsetoptions{fmtord=<type>}
```

where `<type>` is either `level` or `raise`. Since version 3.01 of `fmtcount`, it is also possible to set `<type>` on a language by language basis, see § 4.

### 4 Multilingual Support

Version 1.02 of the `fmtcount` package now has limited multilingual support. The following languages are implemented: English, Spanish, Portuguese, French, French (Swiss) and French (Belgian). German support was added in version 1.1.<sup>2</sup> Italian support was added in version 1.31.<sup>3</sup>

Actually, `fmtcount` has two modes:

- a multilingual mode, in which the commands `\numberstring`, `\ordinalstring`, `\ordinal`, and their variants will be formatted in the currently selected language, as per the `\languagename` macro set by `babel`, `polyglossia` or `suchlikes`, and
- a default mode for backward compatibility in which these commands are formatted in English irrespective of `\languagename`, and to which `fmtcount` falls back when it cannot detect packages such as `babel` or `polyglossia` are loaded.

For multilingual mode, `fmtcount` needs to load correctly the language definition for document dialects. To do this use

`\FCloadlang`

```
\FCloadlang{<dialect>}
```

in the preamble — this will both switch on multilingual mode, and load the `<dialect>` definition. The `<dialect>` should match the options passed to `babel` or `polyglossia`. `fmtcount`

<sup>2</sup>Thanks to K. H. Fricke for supplying the information.

<sup>3</sup>Thanks to Edoardo Pasca for supplying the information.

currently supports the following *<dialect>*'s: english, UKenglish, brazilian, british, USenglish, american, spanish, portugues, portuguese, french, frenchb, francais, german, germanb, ngerman, ngermanb, and italian.

If you don't use `\FCloadlang`, `fmtcount` will attempt to detect the required dialects and call `\FCloadlang` for you, but this isn't guaranteed to work. Notably, when `\FCloadlang` is not used and `fmtcount` has switched on multilingual mode, but without detecting the needed dialects in the preamble, and `fmtcount` has to format a number for a dialect for which definition has not been loaded (via `\FCloadlang` above), then if `fmtcount` detects a definition file for this dialect it will attempt to load it, and cause an error otherwise. This loading in body has not been tested extensively, and may may cause problems such as spurious spaces insertion before the first formatted number, so it's best to use `\FCloadlang` explicitly in the preamble.

If the French language is selected, the `french` option let you configure the dialect and other aspects. The `abbr` also has some influence with French. Please refer to § 4.2.

The male gender for all languages is used by default, however the feminine or neuter forms can be obtained by passing `f` or `n` as an optional argument to `\ordinal`, `\ordinalnum` etc. For example: `\numberstring{section}[f]`. Note that the optional argument comes *after* the compulsory argument. If a gender is not defined in a given language, the masculine version will be used instead.

Let me know if you find any spelling mistakes (has been known to happen in English, let alone other languages with which I'm not so familiar.) If you want to add support for another language, you will need to let me know how to form the numbers and ordinals from 0 to 99999 in that language for each gender.

#### 4.1 Options for setting ordinal ending position raise/level

countsetoptions

```
\fmtcountsetoptions{<language>={fmtord=<type>}}
```

where *<language>* is one of the supported language *<type>* is either `level` or `raise` or `undefine`. If the value is `level` or `raise`, then that will set the `fmtord` option accordingly<sup>4</sup> only for that language *<language>*. If the value is `undefine`, then the non-language specific behaviour is followed.

Some *<language>* are synonyms, here is a table:

language	alias(es)
english	british
french	frenchb
german	germanb ngerman ngermanb
USenglish	american

---

<sup>4</sup>see § 3

## 4.2 Options for French

This section is in French, as it is most useful to French speaking people.

Il est possible de configurer plusieurs aspects de la numérotation en français avec les options `french` et `abbr`. Ces options n'ont d'effet que si le langage `french` est chargé.

countsetoptions

```
\fmtcountsetoptions{french={french options}}
```

L'argument *french options* est une liste entre accolades et séparée par des virgules de réglages de la forme "*clef*=*valeur*", chacun de ces réglages est ci-après désigné par "option française" pour le distinguer des "options générales" telles que `french`.

Le dialecte peut être sélectionné avec l'option française `dialect` dont la valeur *dialect* peut être `france`, `belgian` ou `swiss`.

dialect

```
\fmtcountsetoptions{french={dialect={dialect}}}
```

french

```
\fmtcountsetoptions{french=dialect}
```

Pour alléger la notation et par souci de rétro-compatibilité `france`, `belgian` ou `swiss` sont également des *clef*s pour *french options* à utiliser sans *valeur*.

L'effet de l'option `dialect` est illustré ainsi :

`france` soixante-dix pour 70, quatre-vingts pour 80, et quatre-vingts-dix pour 90,

`belgian` septante pour 70, quatre-vingts pour 80, et nonante pour 90,

`swiss` septante pour 70, huitante<sup>5</sup> pour 80, et nonante pour 90

Il est à noter que la variante `belgian` est parfaitement correcte pour les francophones français<sup>6</sup>, et qu'elle est également utilisée en Suisse Romande hormis dans les cantons de Vaud, du Valais et de Fribourg. En ce qui concerne le mot "octante", il n'est actuellement pas pris en charge et n'est guère plus utilisé, ce qui est sans doute dommage car il est sans doute plus acceptable que le "huitante" de certains de nos amis suisses.

abbr

```
\fmtcountsetoptions{abbr=boolean}
```

L'option générale `abbr` permet de changer l'effet de `\ordinal`. Selon *boolean* on a :

`true` pour produire des ordinaux de la forme 2<sup>e</sup> (par défaut), ou

`false` pour produire des ordinaux de la forme 2<sup>ème</sup>

vingt plural

```
\fmtcountsetoptions{french={vingt plural=french plural control}}
```

cent plural

```
\fmtcountsetoptions{french={cent plural=french plural control}}
```

<sup>5</sup>voir [Octante et huitante](#) sur le site d'Alain Lassine

<sup>6</sup>je précise que l'auteur de ces lignes est français

mil plural `\fmtcountsetoptions{french={mil plural=<french plural control>}}`

n-illion plural `\fmtcountsetoptions{french={n-illion plural=<french plural control>}}`

n-illiard plural `\fmtcountsetoptions{french={n-illiard plural=<french plural control>}}`

all plural `\fmtcountsetoptions{french={all plural=<french plural control>}}`

Les options `vingt plural`, `cent plural`, `mil plural`, `n-illion plural`, et `n-illiard plural`, permettent de contrôler très finement l'accord en nombre des mots respectivement vingt, cent, mil, et des mots de la forme  $\langle n \rangle$ illion et  $\langle n \rangle$ illiard, où  $\langle n \rangle$  désigne 'm' pour 1, 'b' pour 2, 'tr' pour 3, etc. L'option `all plural` est un raccourci permettant de contrôler de concert l'accord en nombre de tous ces mots. Tous ces paramètres valent `reformed` par défaut.

Attention, comme on va l'expliquer, seules quelques combinaisons de configurations de ces options donnent un orthographe correcte vis à vis des règles en vigueur. La raison d'être de ces options est la suivante :

- la règle de l'accord en nombre des noms de nombre dans un numéral cardinal dépend de savoir s'il a vraiment une valeur cardinale ou bien une valeur ordinale, ainsi on écrit « aller à la page deux-cent (sans s) d'un livre de deux-cents (avec s) pages », il faut donc pouvoir changer la configuration pour sélectionner le cas considéré,
- un autre cas demandant quelque configurabilité est celui de « mil » et « mille ». Pour rappel « mille » est le pluriel irrégulier de « mil », mais l'alternance mil/mille est rare, voire pédante, car aujourd'hui « mille » n'est utilisé que comme un mot invariable, en effet le sort des pluriels étrangers est systématiquement de finir par disparaître comme par exemple « scénarii » aujourd'hui supplanté par « scénarios ». Pour continuer à pouvoir écrire « mil », il aurait fallu former le pluriel comme « mils », ce qui n'est pas l'usage. Certaines personnes utilisent toutefois encore « mil » dans les dates, par exemple « mil neuf cent quatre-vingt quatre » au lieu de « mille neuf cent quatre-vingt quatre »,
- finalement les règles du français quoique bien définies ne sont pas très cohérentes et il est donc inévitable qu'un jour ou l'autre on les simplifie. Le paquetage `fmtcount` est déjà prêt à cette éventualité.

Le paramètre  $\langle french plural control \rangle$  peut prendre les valeurs suivantes :

`traditional` pour sélectionner la règle en usage chez les adultes à la date de parution de ce document, et dans le cas des numéraux cardinaux, lorsqu'ils ont une valeur cardinale,

reformed	pour suivre toute nouvelle recommandation à la date de parution de ce document, , et dans le cas des numéraux cardinaux, lorsqu'ils ont une valeur cardinale, l'idée des options <code>traditional</code> et <code>reformed</code> est donc de pouvoir contenter à la fois les anciens et les modernes, mais à dire vrai à la date où ce document est écrit elles ont exactement le même effet,
<code>traditional o</code>	pareil que <code>traditional</code> mais dans le cas des numéraux cardinaux, lorsqu'ils ont une valeur ordinale,
<code>reformed o</code>	pareil que <code>reformed</code> mais dans le cas des numéraux cardinaux, lorsqu'ils ont une valeur ordinale, de même que précédemment <code>reformed o</code> et <code>traditional o</code> ont exactement le même effet,
<code>always</code>	pour marquer toujours le pluriel, ceci n'est correct que pour « mil » vis à vis des règles en vigueur,
<code>never</code>	pour ne jamais marquer le pluriel, ceci est incorrect vis à vis des règles d'orthographe en vigueur,
<code>multiple</code>	pour marquer le pluriel lorsque le nombre considéré est multiplié par au moins 2, ceci est la règle en vigueur pour les nombres de la forme $\langle n \rangle$ illion et $\langle n \rangle$ illiard lorsque le nombre a une valeur cardinale,
<code>multiple g-last</code>	pour marquer le pluriel lorsque le nombre considéré est multiplié par au moins 2 est est <i>globalement</i> en dernière position, où "globalement" signifie qu'on considère le nombre formaté en entier, ceci est incorrect vis à vis des règles d'orthographe en vigueur,
<code>multiple l-last</code>	pour marquer le pluriel lorsque le nombre considéré est multiplié par au moins 2 et est <i>localement</i> en dernière position, où "localement" signifie qu'on considère seulement la portion du nombre qui multiplie soit l'unité, soit un $\langle n \rangle$ illion ou un $\langle n \rangle$ illiard; ceci est la convention en vigueur pour le pluriel de "vingt" et de "cent" lorsque le nombre formaté a une valeur cardinale,
<code>multiple lng-last</code>	pour marquer le pluriel lorsque le nombre considéré est multiplié par au moins 2 et est <i>localement</i> mais <i>non globalement</i> en dernière position, où "localement" et <i>globalement</i> on la même signification que pour les options <code>multiple g-last</code> et <code>multiple l-last</code> ; ceci est la convention en vigueur pour le pluriel de "vingt" et de "cent" lorsque le nombre formaté a une valeur ordinale,
<code>multiple ng-last</code>	pour marquer le pluriel lorsque le nombre considéré est multiplié par au moins 2, et <i>n'</i> est pas <i>globalement</i> en dernière position, où "globalement" a la même signification que pour l'option <code>multiple g-last</code> ; ceci est la règle que j'infère être en vigueur pour les nombres de la forme $\langle n \rangle$ illion et $\langle n \rangle$ illiard lorsque le nombre a une valeur ordinale, mais à dire vrai pour des nombres aussi grands, par exemple « deux millions », je pense qu'il n'est tout simplement pas d'usage de dire « l'exemplaire deux million(s?) » pour « le deux millionième exemplaire ».

L'effet des paramètres `traditional`, `traditional o`, `reformed`, et `reformed o`, est le suivant :

$\langle x \rangle$ dans “ $\langle x \rangle$ plural”	<code>traditional</code>	<code>reformed</code>	<code>traditional o</code>	<code>reformed o</code>
<code>vingt</code>	<code>multiple l-last</code>		<code>multiple lng-last</code>	
<code>cent</code>	<code>multiple l-last</code>		<code>multiple lng-last</code>	
<code>mil</code>	<code>always</code>			
<code>n-illion</code>	<code>multiple</code>		<code>multiple ng-last</code>	
<code>n-illiard</code>	<code>multiple</code>		<code>multiple ng-last</code>	

Les configurations qui respectent les règles d'orthographe sont les suivantes :

- `\fmtcountsetoptions{french={all plural=reformed o}}` pour formater les numéraux cardinaux à valeur ordinale,
- `\fmtcountsetoptions{french={mil plural=multiple}}` pour activer l'alternance mil/mille.
- `\fmtcountsetoptions{french={all plural=reformed}}` pour revenir dans la configuration par défaut.

dash or space

```
\fmtcountsetoptions{french={dash or space=dash or space}}
```

Avant la réforme de l'orthographe de 1990, on ne met des traits d'union qu'entre les dizaines et les unités, et encore sauf quand le nombre  $n$  considéré est tel que  $n \bmod 10 = 1$ , dans ce cas on écrit “et un” sans trait d'union. Après la réforme de 1990, on recommande de mettre des traits d'union de partout sauf autour de “mille”, “million” et “milliard”, et les mots analogues comme “billion”, “billiard”. Cette exception a toutefois été contestée par de nombreux auteurs, et on peut aussi mettre des traits d'union de partout. Mettre l'option  $\langle dash or space \rangle$  à :

`traditional` pour sélectionner la règle d'avant la réforme de 1990,  
`1990` pour suivre la recommandation de la réforme de 1990,  
`reformed` pour suivre la recommandation de la dernière réforme prise en charge, actuellement l'effet est le même que 1990, ou à  
`always` pour mettre systématiquement des traits d'union de partout.

Par défaut, l'option vaut `reformed`.

scale

```
\fmtcountsetoptions{french={scale=scale}}
```

L'option `scale` permet de configurer l'écriture des grands nombres. Mettre  $\langle scale \rangle$  à :

- recursive dans ce cas  $10^{30}$  donne mille milliards de milliards de milliards, pour  $10^n$ , on écrit  $10^{n-9 \times \max\{(n \div 9) - 1, 0\}}$  suivi de la répétition  $\max\{(n \div 9) - 1, 0\}$  fois de “de milliards”
- long  $10^{6 \times n}$  donne un  $\langle n \rangle$ illion où  $\langle n \rangle$  est remplacé par “bi” pour 2, “tri” pour 3, etc. et  $10^{6 \times n + 3}$  donne un  $\langle n \rangle$ illiard avec la même convention pour  $\langle n \rangle$ . L’option long est correcte en Europe, par contre j’ignore l’usage au Québec.
- short  $10^{6 \times n}$  donne un  $\langle n \rangle$ illion où  $\langle n \rangle$  est remplacé par “bi” pour 2, “tri” pour 3, etc. L’option short est incorrecte en Europe.

Par défaut, l’option vaut recursive.

n-illiard upto

```
\fmtcountsetoptions{french={n-illiard upto=\langle n-illiard upto \rangle}}
```

Cette option n’a de sens que si scale vaut long. Certaines personnes préfèrent dire “mille  $\langle n \rangle$ illions” qu’un “ $\langle n \rangle$ illiard”. Mettre l’option n-illiard upto à :

- infinity pour que  $10^{6 \times n + 3}$  donne  $\langle n \rangle$ illiards pour tout  $n > 0$ ,
- infty même effet que infinity,
- k où  $k$  est un entier quelconque strictement positif, dans ce cas  $10^{6 \times n + 3}$  donne “mille  $\langle n \rangle$ illions” lorsque  $n > k$ , et donne “ $\langle n \rangle$ illiard” sinon

mil plural mark

```
\fmtcountsetoptions{french={mil plural mark=\langle any text \rangle}}
```

La valeur par défaut de cette option est « le ». Il s’agit de la terminaison ajoutée à « mil » pour former le pluriel, c’est à dire « mille », cette option ne sert pas à grand chose sauf dans l’éventualité où ce pluriel serait francisé un jour — à dire vrai si cela se produisait une alternance mille/milles est plus vraisemblable, car « mille » est plus fréquent que « mille » et que les pluriels francisés sont formés en ajoutant « s » à la forme la plus fréquente, par exemple « blini/blinis », alors que « blini » veut dire « crêpes » (au pluriel).

### 4.3 Prefixes

latinnumeralstring

```
\latinnumeralstring{\langle counter \rangle}[\langle prefix options \rangle]
```

latinnumeralstringnum

```
\latinnumeralstringnum{\langle number \rangle}[\langle prefix options \rangle]
```

## 5 Configuration File `fmtcount.cfg`

You can save your preferred default settings to a file called `fmtcount.cfg`, and place it on the TeX path. These settings will then be loaded by the `fmtcount` package.

Note that if you are using the `datetime` package, the `datetime.cfg` configuration file will override the `fmtcount.cfg` configuration file. For example, if `datetime.cfg` has the line:

```
\renewcommand{\fmtord}[1]{\textsuperscript{\underline{\#1}}}
```

and if `fmtcount.cfg` has the line:

```
\fmtcountsetoptions{fmtord=level}
```

then the former definition of `\fmtord` will take precedence.

## 6 LaTeX2HTML style

The LaTeX2HTML style file `fmtcount.perl` is provided. The following limitations apply:

- `\padzeroes` only has an effect in the preamble.
- The configuration file `fmtcount.cfg` is currently ignored. (This is because I can't work out the correct code to do this. If you know how to do this, please let me know.) You can however do:

```
\usepackage{fmtcount}  
\html{\input{fmtcount.cfg}}
```

This, I agree, is an unpleasant cludge.

## 7 Acknowledgements

I would like to thank all the people who have provided translations.

## 8 Troubleshooting

There is a FAQ available at: <http://theoval.cmp.uea.ac.uk/~nlct/latex/packages/faq/>.

Bug reporting should be done via the Github issue manager at: <https://github.com/nlct/fmtcount/issues/>.

Local Variables: coding: utf-8 End:

## 9 The Code

### 9.1 Language definition files

#### 9.1.1 `fc-american.def`

American English definitions

```
1 \ProvidesFCLanguage{american}[2016/01/12]%
```

Loaded `fc-USenglish.def` if not already loaded

```
2 \FCloadlang{USenglish}%
```

These are all just synonyms for the commands provided by fc-USenglish.def.

```
3 \global\let\@ordinalMamerican\@ordinalMUSenglish
4 \global\let\@ordinalFamerican\@ordinalMUSenglish
5 \global\let\@ordinalNamerican\@ordinalMUSenglish
6 \global\let\@numberstringMamerican\@numberstringMUSenglish
7 \global\let\@numberstringFamerican\@numberstringMUSenglish
8 \global\let\@numberstringNamerican\@numberstringMUSenglish
9 \global\let\@NumberstringMamerican\@NumberstringMUSenglish
10 \global\let\@NumberstringFamerican\@NumberstringMUSenglish
11 \global\let\@NumberstringNamerican\@NumberstringMUSenglish
12 \global\let\@ordinalstringMamerican\@ordinalstringMUSenglish
13 \global\let\@ordinalstringFamerican\@ordinalstringMUSenglish
14 \global\let\@ordinalstringNamerican\@ordinalstringMUSenglish
15 \global\let\@OrdinalstringMamerican\@OrdinalstringMUSenglish
16 \global\let\@OrdinalstringFamerican\@OrdinalstringMUSenglish
17 \global\let\@OrdinalstringNamerican\@OrdinalstringMUSenglish
```

### 9.1.2 fc-brazilian.def

Brazilian definitions.

```
18 \ProvidesFCLanguage{brazilian}[2017/12/26]%
```

Load fc-portuges.def if not already loaded.

```
19 \FCloadlang{portuges}%
```

Set brazilian to be equivalent to portuges for all the numeral ordinals, and string ordinals.

```
20 \global\let\@ordinalMbrazilian=\@ordinalMportuges
21 \global\let\@ordinalFbrazilian=\@ordinalFportuges
22 \global\let\@ordinalNbrazilian=\@ordinalNportuges
23 \global\let\@ordinalstringFbrazilian\@ordinalstringFportuges
24 \global\let\@ordinalstringMbrazilian\@ordinalstringMportuges
25 \global\let\@ordinalstringNbrazilian\@ordinalstringMportuges
26 \global\let\@OrdinalstringMbrazilian\@OrdinalstringMportuges
27 \global\let\@OrdinalstringFbrazilian\@OrdinalstringFportuges
28 \global\let\@OrdinalstringNbrazilian\@OrdinalstringMportuges
```

Convert a number to a textual representation. To make it easier, split it up into units, tens, teens and hundreds. Units, tens, and hundreds are the same as for portuges and are not redefined, only the teens are Brazilian specific.

Teens (argument must be a number from 0 to 9):

```
29 \newcommand*{\@teenstringbrazilian[1]}{%
30   \ifcase#1\relax
31     dez%
32     \or onze%
33     \or doze%
34     \or treze%
35     \or quatorze%
36     \or quinze%
37     \or dezesseis%
38     \or dezessete%
```

```

39   \or dezoito%
40   \or dezenove%
41   \fi
42 }%
43 \global\let\@@teenstringbrazilian\@@teenstringbrazilian

```

Teens (with initial letter in upper case):

```

44 \newcommand*\@@Teenstringbrazilian[1]{%
45   \ifcase#1\relax
46     Dez%
47     \or Onze%
48     \or Doze%
49     \or Treze%
50     \or Quatorze%
51     \or Quinze%
52     \or Dezesesseis%
53     \or Dezessete%
54     \or Dezoito%
55     \or Dezenove%
56   \fi
57 }%
58 \global\let\@@Teenstringbrazilian\@@Teenstringbrazilian

```

This has changed in version 1.08, so that it now stores the result in the second argument, but doesn't display anything. Since it only affects internal macros, it shouldn't affect documents created with older versions. (These internal macros are not meant for use in documents.)

```

59 \newcommand*\@@numberstringMbrazilian}[2]{%
60   \let\@unitstring=\@unitstringportuges
61   \let\@teenstring=\@teenstringbrazilian
62   \let\@tenstring=\@tenstringportuges
63   \let\@hundredstring=\@hundredstringportuges
64   \def\@hundred{cem}\def\@thousand{mil}%
65   \def\@andname{e}%
66   \@numberstringportuges{#1}{#2}%
67 }%
68 \global\let\@numberstringMbrazilian\@numberstringMbrazilian

```

As above, but feminine form:

```

69 \newcommand*\@@numberstringFbrazilian}[2]{%
70   \let\@unitstring=\@unitstringFportuges
71   \let\@teenstring=\@teenstringbrazilian
72   \let\@tenstring=\@tenstringportuges
73   \let\@hundredstring=\@hundredstringFportuges
74   \def\@hundred{cem}\def\@thousand{mil}%
75   \def\@andname{e}%
76   \@numberstringportuges{#1}{#2}%
77 }%
78 \global\let\@numberstringFbrazilian\@numberstringFbrazilian

```

Make neuter same as masculine:

```

79 \global\let\@numberstringNbrazilian\@numberstringMbrazilian

```

As above, but initial letters in upper case:

```
80 \newcommand*{\@NumberstringMbrazilian}[2]{%
81   \let\@unitstring=\@@unitstringportuges
82   \let\@teenstring=\@@Teenstringbrazilian
83   \let\@tenstring=\@@Tenstringportuges
84   \let\@hundredstring=\@@hundredstringportuges
85   \def\@hundred{Cem}\def\@thousand{Mil}%
86   \def\@andname{e}%
87   \@numberstringportuges{#1}{#2}%
88 }%
89 \global\let\@NumberstringMbrazilian\@NumberstringMbrazilian
```

As above, but feminine form:

```
90 \newcommand*{\@NumberstringFbrazilian}[2]{%
91   \let\@unitstring=\@@UnitstringFportuges
92   \let\@teenstring=\@@Teenstringbrazilian
93   \let\@tenstring=\@@Tenstringportuges
94   \let\@hundredstring=\@@HundredstringFportuges
95   \def\@hundred{Cem}\def\@thousand{Mil}%
96   \def\@andname{e}%
97   \@numberstringportuges{#1}{#2}%
98 }%
99 \global\let\@NumberstringFbrazilian\@NumberstringFbrazilian
```

Make neuter same as masculine:

```
100 \global\let\@NumberstringNbrazilian\@NumberstringMbrazilian
```

### 9.1.3 fc-british.def

British definitions

```
101 \ProvidesFCLanguage{british}[2013/08/17]%
```

Load fc-english.def, if not already loaded

```
102 \FCloadlang{english}%
```

These are all just synonyms for the commands provided by fc-english.def.

```
103 \global\let\@ordinalMbritish\@ordinalMenglish
104 \global\let\@ordinalFbritish\@ordinalMenglish
105 \global\let\@ordinalNbritish\@ordinalMenglish
106 \global\let\@numberstringMbritish\@numberstringMenglish
107 \global\let\@numberstringFbritish\@numberstringMenglish
108 \global\let\@numberstringNbritish\@numberstringMenglish
109 \global\let\@NumberstringMbritish\@NumberstringMenglish
110 \global\let\@NumberstringFbritish\@NumberstringMenglish
111 \global\let\@NumberstringNbritish\@NumberstringMenglish
112 \global\let\@ordinalstringMbritish\@ordinalstringMenglish
113 \global\let\@ordinalstringFbritish\@ordinalstringMenglish
114 \global\let\@ordinalstringNbritish\@ordinalstringMenglish
115 \global\let\@OrdinalstringMbritish\@OrdinalstringMenglish
116 \global\let\@OrdinalstringFbritish\@OrdinalstringMenglish
117 \global\let\@OrdinalstringNbritish\@OrdinalstringMenglish
```

#### 9.1.4 fc-english.def

English definitions

```
118 \ProvidesFCLanguage{english}[2016/01/12]%
```

Define macro that converts a number or count register (first argument) to an ordinal, and stores the result in the second argument, which should be a control sequence.

```
119 \newcommand*\@ordinalMenglish[2]{%
120 \def\@fc@ord{}%
121 \@orgargctr=#1\relax
122 \@ordinalctr=#1%
123 \@FCmodulo{\@ordinalctr}{100}%
124 \ifnum\@ordinalctr=11\relax
125 \def\@fc@ord{th}%
126 \else
127 \ifnum\@ordinalctr=12\relax
128 \def\@fc@ord{th}%
129 \else
130 \ifnum\@ordinalctr=13\relax
131 \def\@fc@ord{th}%
132 \else
133 \@FCmodulo{\@ordinalctr}{10}%
134 \ifcase\@ordinalctr
135 \def\@fc@ord{th}% case 0
136 \or \def\@fc@ord{st}% case 1
137 \or \def\@fc@ord{nd}% case 2
138 \or \def\@fc@ord{rd}% case 3
139 \else
140 \def\@fc@ord{th}% default case
141 \fi
142 \fi
143 \fi
144 \fi
145 \edef#2{\number#1\relax\noexpand\fmtord{\@fc@ord}}%
146 }%
147 \global\let\@ordinalMenglish\@ordinalMenglish
```

There is no gender difference in English, so make feminine and neuter the same as the masculine.

```
148 \global\let\@ordinalFenglish=\@ordinalMenglish
149 \global\let\@ordinalNenglish=\@ordinalMenglish
```

Define the macro that prints the value of a TeX count register as text. To make it easier, break it up into units, teens and tens. First, the units: the argument should be between 0 and 9 inclusive.

```
150 \newcommand*\@@unitstringenglish[1]{%
151 \ifcase#1\relax
152 zero%
153 \or one%
154 \or two%
```

```

155 \or three%
156 \or four%
157 \or five%
158 \or six%
159 \or seven%
160 \or eight%
161 \or nine%
162 \fi
163 }%
164 \global\let\@@unitstringenglish\@@unitstringenglish

```

Next the tens, again the argument should be between 0 and 9 inclusive.

```

165 \newcommand*\@@tenstringenglish[1]{%
166 \ifcase#1\relax
167 \or ten%
168 \or twenty%
169 \or thirty%
170 \or forty%
171 \or fifty%
172 \or sixty%
173 \or seventy%
174 \or eighty%
175 \or ninety%
176 \fi
177 }%
178 \global\let\@@tenstringenglish\@@tenstringenglish

```

Finally the teens, again the argument should be between 0 and 9 inclusive.

```

179 \newcommand*\@@teenstringenglish[1]{%
180 \ifcase#1\relax
181 ten%
182 \or eleven%
183 \or twelve%
184 \or thirteen%
185 \or fourteen%
186 \or fifteen%
187 \or sixteen%
188 \or seventeen%
189 \or eighteen%
190 \or nineteen%
191 \fi
192 }%
193 \global\let\@@teenstringenglish\@@teenstringenglish

```

As above, but with the initial letter in uppercase. The units:

```

194 \newcommand*\@@Unitstringenglish[1]{%
195 \ifcase#1\relax
196 Zero%
197 \or One%
198 \or Two%
199 \or Three%

```

```

200 \or Four%
201 \or Five%
202 \or Six%
203 \or Seven%
204 \or Eight%
205 \or Nine%
206 \fi
207 }%
208 \global\let\@@Unitstringenglish\@@Unitstringenglish

```

The tens:

```

209 \newcommand*\@@Tenstringenglish[1]{%
210 \ifcase#1\relax
211 \or Ten%
212 \or Twenty%
213 \or Thirty%
214 \or Forty%
215 \or Fifty%
216 \or Sixty%
217 \or Seventy%
218 \or Eighty%
219 \or Ninety%
220 \fi
221 }%
222 \global\let\@@Tenstringenglish\@@Tenstringenglish

```

The teens:

```

223 \newcommand*\@@Teenstringenglish[1]{%
224 \ifcase#1\relax
225 Ten%
226 \or Eleven%
227 \or Twelve%
228 \or Thirteen%
229 \or Fourteen%
230 \or Fifteen%
231 \or Sixteen%
232 \or Seventeen%
233 \or Eighteen%
234 \or Nineteen%
235 \fi
236 }%
237 \global\let\@@Teenstringenglish\@@Teenstringenglish

```

This has changed in version 1.09, so that it now stores the result in the second argument, but doesn't display anything. Since it only affects internal macros, it shouldn't affect documents created with older versions. (These internal macros are not meant for use in documents.)

```

238 \newcommand*\@@numberstringenglish[2]{%
239 \ifnum#1>99999
240 \PackageError{fmtcount}{Out of range}%
241 {This macro only works for values less than 100000}%

```

```

242 \else
243 \ifnum#1<0
244 \PackageError{fmtcount}{Negative numbers not permitted}%
245 {This macro does not work for negative numbers, however
246 you can try typing "minus" first, and then pass the modulus of
247 this number}%
248 \fi
249 \fi
250 \def#2{}%
251 \@strctr=#1\relax \divide\@strctr by 1000\relax
252 \ifnum\@strctr>9
253 \divide\@strctr by 10
254 \ifnum\@strctr>1\relax
255 \let\@fc@numstr#2\relax
256 \edef#2{\@fc@numstr\@tenstring{\@strctr}}%
257 \@strctr=#1 \divide\@strctr by 1000\relax
258 \@FCmodulo{\@strctr}{10}%
259 \ifnum\@strctr>0\relax
260 \let\@fc@numstr#2\relax
261 \edef#2{\@fc@numstr-\@unitstring{\@strctr}}%
262 \fi
263 \else
264 \@strctr=#1\relax
265 \divide\@strctr by 1000\relax
266 \@FCmodulo{\@strctr}{10}%
267 \let\@fc@numstr#2\relax
268 \edef#2{\@fc@numstr\@teenstring{\@strctr}}%
269 \fi
270 \let\@fc@numstr#2\relax
271 \edef#2{\@fc@numstr\ \@thousand}%
272 \else
273 \ifnum\@strctr>0\relax
274 \let\@fc@numstr#2\relax
275 \edef#2{\@fc@numstr\@unitstring{\@strctr}\ \@thousand}%
276 \fi
277 \fi
278 \@strctr=#1\relax \@FCmodulo{\@strctr}{1000}%
279 \divide\@strctr by 100
280 \ifnum\@strctr>0\relax
281 \ifnum#1>1000\relax
282 \let\@fc@numstr#2\relax
283 \edef#2{\@fc@numstr\ }%
284 \fi
285 \let\@fc@numstr#2\relax
286 \edef#2{\@fc@numstr\@unitstring{\@strctr}\ \@hundred}%
287 \fi
288 \@strctr=#1\relax \@FCmodulo{\@strctr}{100}%
289 \ifnum#1>100\relax
290 \ifnum\@strctr>0\relax

```

```

291 \let\@fc@numstr#2\relax
292 \edef#2{\@fc@numstr\ \@andname\ }%
293 \fi
294 \fi
295 \ifnum\@strctr>19\relax
296 \divide\@strctr by 10\relax
297 \let\@fc@numstr#2\relax
298 \edef#2{\@fc@numstr\@tenstring{\@strctr}}%
299 \@strctr=#1\relax \@FCmodulo{\@strctr}{10}%
300 \ifnum\@strctr>0\relax
301 \let\@fc@numstr#2\relax
302 \edef#2{\@fc@numstr-\@unitstring{\@strctr}}%
303 \fi
304 \else
305 \ifnum\@strctr<10\relax
306 \ifnum\@strctr=0\relax
307 \ifnum#1<100\relax
308 \let\@fc@numstr#2\relax
309 \edef#2{\@fc@numstr\@unitstring{\@strctr}}%
310 \fi
311 \else
312 \let\@fc@numstr#2\relax
313 \edef#2{\@fc@numstr\@unitstring{\@strctr}}%
314 \fi
315 \else
316 \@FCmodulo{\@strctr}{10}%
317 \let\@fc@numstr#2\relax
318 \edef#2{\@fc@numstr\@teenstring{\@strctr}}%
319 \fi
320 \fi
321 }%
322 \global\let\@numberstringenglish\@numberstringenglish

```

All lower case version, the second argument must be a control sequence.

```

323 \newcommand*{\@numberstringMenglish}[2]{%
324 \let\@unitstring=\@unitstringenglish
325 \let\@teenstring=\@teenstringenglish
326 \let\@tenstring=\@tenstringenglish
327 \def\@hundred{hundred}\def\@thousand{thousand}%
328 \def\@andname{and}%
329 \@numberstringenglish{#1}{#2}%
330 }%
331 \global\let\@numberstringMenglish\@numberstringMenglish

```

There is no gender in English, so make feminine and neuter the same as the masculine.

```

332 \global\let\@numberstringFenglish=\@numberstringMenglish
333 \global\let\@numberstringNenglish=\@numberstringMenglish

```

This version makes the first letter of each word an uppercase character (except “and”). The second argument must be a control sequence.

```

334 \newcommand*\@NumberstringMenglish[2]{%
335   \let\@unitstring=\@Unitstringenglish
336   \let\@teenstring=\@Teenstringenglish
337   \let\@tenstring=\@Tenstringenglish
338   \def\@hundred{Hundred}\def\@thousand{Thousand}%
339   \def\@andname{and}%
340   \@numberstringenglish{#1}{#2}%
341 }%
342 \global\let\@NumberstringMenglish\@NumberstringMenglish

```

There is no gender in English, so make feminine and neuter the same as the masculine.

```

343 \global\let\@NumberstringFenglish=\@NumberstringMenglish
344 \global\let\@NumberstringNenglish=\@NumberstringMenglish

```

Define a macro that produces an ordinal as a string. Again, break it up into units, teens and tens. First the units:

```

345 \newcommand*\@unitthstringenglish[1]{%
346   \ifcase#1\relax
347     zeroth%
348   \or first%
349   \or second%
350   \or third%
351   \or fourth%
352   \or fifth%
353   \or sixth%
354   \or seventh%
355   \or eighth%
356   \or ninth%
357   \fi
358 }%
359 \global\let\@unitthstringenglish\@unitthstringenglish

```

Next the tens:

```

360 \newcommand*\@tenthstringenglish[1]{%
361   \ifcase#1\relax
362     \or tenth%
363     \or twentieth%
364     \or thirtieth%
365     \or fortieth%
366     \or fiftieth%
367     \or sixtieth%
368     \or seventieth%
369     \or eightieth%
370     \or ninetieth%
371   \fi
372 }%
373 \global\let\@tenthstringenglish\@tenthstringenglish

```

The teens:

```

374 \newcommand*\@teenthstringenglish[1]{%
375   \ifcase#1\relax

```

```

376 tenth%
377 \or eleventh%
378 \or twelfth%
379 \or thirteenth%
380 \or fourteenth%
381 \or fifteenth%
382 \or sixteenth%
383 \or seventeenth%
384 \or eighteenth%
385 \or nineteenth%
386 \fi
387 }%
388 \global\let\@@teenthstringenglish\@@teenthstringenglish

```

As before, but with the first letter in upper case. The units:

```

389 \newcommand*\@@Unitthstringenglish[1]{%
390 \ifcase#1\relax
391 Zeroth%
392 \or First%
393 \or Second%
394 \or Third%
395 \or Fourth%
396 \or Fifth%
397 \or Sixth%
398 \or Seventh%
399 \or Eighth%
400 \or Ninth%
401 \fi
402 }%
403 \global\let\@@Unitthstringenglish\@@Unitthstringenglish

```

The tens:

```

404 \newcommand*\@@Tenthstringenglish[1]{%
405 \ifcase#1\relax
406 \or Tenth%
407 \or Twentieth%
408 \or Thirtieth%
409 \or Fortieth%
410 \or Fiftieth%
411 \or Sixtieth%
412 \or Seventieth%
413 \or Eightieth%
414 \or Ninetieth%
415 \fi
416 }%
417 \global\let\@@Tenthstringenglish\@@Tenthstringenglish

```

The tens:

```

418 \newcommand*\@@Teenthstringenglish[1]{%
419 \ifcase#1\relax
420 Tenth%

```

```

421 \or Eleventh%
422 \or Twelfth%
423 \or Thirteenth%
424 \or Fourteenth%
425 \or Fifteenth%
426 \or Sixteenth%
427 \or Seventeenth%
428 \or Eighteenth%
429 \or Nineteenth%
430 \fi
431 }%
432 \global\let\@@Teenthstringenglish\@@Teenthstringenglish

Again, as from version 1.09, this has been changed to take two arguments, where the second
argument is a control sequence. The resulting text is stored in the control sequence, and
nothing is displayed.

433 \newcommand*\@@ordinalstringenglish[2]{%
434 \@strctr=#1\relax
435 \ifnum#1>99999
436 \PackageError{fmtcount}{Out of range}%
437 {This macro only works for values less than 100000 (value given: \number\@strctr)}%
438 \else
439 \ifnum#1<0
440 \PackageError{fmtcount}{Negative numbers not permitted}%
441 {This macro does not work for negative numbers, however
442 you can try typing "minus" first, and then pass the modulus of
443 this number}%
444 \fi
445 \def#2{}%
446 \fi
447 \@strctr=#1\relax \divide\@strctr by 1000\relax
448 \ifnum\@strctr>9\relax

#1 is greater or equal to 10000

449 \divide\@strctr by 10
450 \ifnum\@strctr>1\relax
451 \let\@@fc@ordstr#2\relax
452 \edef#2{\@@fc@ordstr\@tenstring{\@strctr}}%
453 \@strctr=#1\relax
454 \divide\@strctr by 1000\relax
455 \@FCmodulo{\@strctr}{10}%
456 \ifnum\@strctr>0\relax
457 \let\@@fc@ordstr#2\relax
458 \edef#2{\@@fc@ordstr-\@unitstring{\@strctr}}%
459 \fi
460 \else
461 \@strctr=#1\relax \divide\@strctr by 1000\relax
462 \@FCmodulo{\@strctr}{10}%
463 \let\@@fc@ordstr#2\relax
464 \edef#2{\@@fc@ordstr\@teenstring{\@strctr}}%

```

```

465 \fi
466 \@strctr=#1\relax \@FCmodulo{\@strctr}{1000}%
467 \ifnum\@strctr=0\relax
468   \let\@@fc@ordstr#2\relax
469   \edef#2{\@@fc@ordstr\ \@thousandth}%
470 \else
471   \let\@@fc@ordstr#2\relax
472   \edef#2{\@@fc@ordstr\ \@thousand}%
473 \fi
474 \else
475 \ifnum\@strctr>0\relax
476   \let\@@fc@ordstr#2\relax
477   \edef#2{\@@fc@ordstr\@unitstring{\@strctr}}%
478   \@strctr=#1\relax \@FCmodulo{\@strctr}{1000}%
479   \let\@@fc@ordstr#2\relax
480   \ifnum\@strctr=0\relax
481     \edef#2{\@@fc@ordstr\ \@thousandth}%
482   \else
483     \edef#2{\@@fc@ordstr\ \@thousand}%
484   \fi
485 \fi
486 \fi
487 \@strctr=#1\relax \@FCmodulo{\@strctr}{1000}%
488 \divide\@strctr by 100
489 \ifnum\@strctr>0\relax
490 \ifnum#1>1000\relax
491   \let\@@fc@ordstr#2\relax
492   \edef#2{\@@fc@ordstr\ }%
493 \fi
494 \let\@@fc@ordstr#2\relax
495 \edef#2{\@@fc@ordstr\@unitstring{\@strctr}}%
496 \@strctr=#1\relax \@FCmodulo{\@strctr}{100}%
497 \let\@@fc@ordstr#2\relax
498 \ifnum\@strctr=0\relax
499   \edef#2{\@@fc@ordstr\ \@hundredth}%
500 \else
501   \edef#2{\@@fc@ordstr\ \@hundred}%
502 \fi
503 \fi
504 \@strctr=#1\relax \@FCmodulo{\@strctr}{100}%
505 \ifnum#1>100\relax
506 \ifnum\@strctr>0\relax
507   \let\@@fc@ordstr#2\relax
508   \edef#2{\@@fc@ordstr\ \@andname\ }%
509 \fi
510 \fi
511 \ifnum\@strctr>19\relax
512 \@tmpstrctr=\@strctr
513 \divide\@strctr by 10\relax

```

```

514 \@FCmodulo{\@tmpstrctr}{10}%
515 \let\@fc@ordstr#2\relax
516 \ifnum\@tmpstrctr=0\relax
517   \edef#2{\@fc@ordstr\@tenthstring{\@strctr}}%
518 \else
519   \edef#2{\@fc@ordstr\@tenstring{\@strctr}}%
520 \fi
521 \@strctr=#1\relax \@FCmodulo{\@strctr}{10}%
522 \ifnum\@strctr>0\relax
523   \let\@fc@ordstr#2\relax
524   \edef#2{\@fc@ordstr-\@unitthstring{\@strctr}}%
525 \fi
526 \else
527   \ifnum\@strctr<10\relax
528     \ifnum\@strctr=0\relax
529       \ifnum#1<100\relax
530         \let\@fc@ordstr#2\relax
531         \edef#2{\@fc@ordstr\@unitthstring{\@strctr}}%
532       \fi
533     \else
534       \let\@fc@ordstr#2\relax
535       \edef#2{\@fc@ordstr\@unitthstring{\@strctr}}%
536     \fi
537   \else
538     \@FCmodulo{\@strctr}{10}%
539     \let\@fc@ordstr#2\relax
540     \edef#2{\@fc@ordstr\@teenthstring{\@strctr}}%
541   \fi
542 \fi
543 }%
544 \global\let\@ordinalstringenglish\@ordinalstringenglish

```

All lower case version. Again, the second argument must be a control sequence in which the resulting text is stored.

```

545 \newcommand*{\@ordinalstringMenglish}[2]{%
546   \let\@unitthstring=\@unitthstringenglish
547   \let\@teenthstring=\@teenthstringenglish
548   \let\@tenthstring=\@tenthstringenglish
549   \let\@unitstring=\@unitstringenglish
550   \let\@teenstring=\@teenstringenglish
551   \let\@tenstring=\@tenstringenglish
552   \def\@andname{and}%
553   \def\@hundred{hundred}\def\@thousand{thousand}%
554   \def\@hundredth{hundredth}\def\@thousandth{thousandth}%
555   \@ordinalstringenglish{#1}{#2}%
556 }%
557 \global\let\@ordinalstringMenglish\@ordinalstringMenglish

```

No gender in English, so make feminine and neuter same as masculine:

```

558 \global\let\@ordinalstringFenglish=\@ordinalstringMenglish

```

```
559 \global\let\@OrdinalstringNenglish=\@OrdinalstringMenglish
```

First letter of each word in upper case:

```
560 \newcommand*{\@OrdinalstringMenglish}[2]{%
561   \let\@unitthstring=\@Unitthstringenglish
562   \let\@teenthstring=\@Teenthstringenglish
563   \let\@tenthstring=\@Tenthstringenglish
564   \let\@unitstring=\@Unitstringenglish
565   \let\@teenstring=\@Teenstringenglish
566   \let\@tenstring=\@Tenstringenglish
567   \def\@andname{and}%
568   \def\@hundred{Hundred}\def\@thousand{Thousand}%
569   \def\@hundredth{Hundredth}\def\@thousandth{Thousandth}%
570   \@ordinalstringenglish{#1}{#2}%
571 }%
572 \global\let\@OrdinalstringMenglish\@OrdinalstringMenglish
```

No gender in English, so make feminine and neuter same as masculine:

```
573 \global\let\@OrdinalstringFenglish=\@OrdinalstringMenglish
574 \global\let\@OrdinalstringNenglish=\@OrdinalstringMenglish
```

### 9.1.5 fc-francais.def

```
575 \ProvidesFCLanguage{francais}[2013/08/17]%
576 \FCloadlang{french}%
```

Set francais to be equivalent to french.

```
577 \global\let\@ordinalMfrancais=\@ordinalMfrench
578 \global\let\@ordinalFfrancais=\@ordinalFfrench
579 \global\let\@ordinalNfrancais=\@ordinalNfrench
580 \global\let\@numberstringMfrancais=\@numberstringMfrench
581 \global\let\@numberstringFfrancais=\@numberstringFfrench
582 \global\let\@numberstringNfrancais=\@numberstringNfrench
583 \global\let\@NumberstringMfrancais=\@NumberstringMfrench
584 \global\let\@NumberstringFfrancais=\@NumberstringFfrench
585 \global\let\@NumberstringNfrancais=\@NumberstringNfrench
586 \global\let\@ordinalstringMfrancais=\@ordinalstringMfrench
587 \global\let\@ordinalstringFfrancais=\@ordinalstringFfrench
588 \global\let\@ordinalstringNfrancais=\@ordinalstringNfrench
589 \global\let\@OrdinalstringMfrancais=\@OrdinalstringMfrench
590 \global\let\@OrdinalstringFfrancais=\@OrdinalstringFfrench
591 \global\let\@OrdinalstringNfrancais=\@OrdinalstringNfrench
```

### 9.1.6 fc-french.def

Definitions for French.

```
592 \ProvidesFCLanguage{french}[2017/06/15]%
```

Package fcprefix is needed to format the prefix  $\langle n \rangle$  in  $\langle n \rangle$ illion or  $\langle n \rangle$ illiard. Big numbers were developed based on reference: [http://www.alain.be/boece/noms\\_de\\_nombre.html](http://www.alain.be/boece/noms_de_nombre.html). Package fcprefix is now loaded by fmtcount.

First of all we define two macros `\fc@gl@let` and `\fc@gl@def` used in place of `\let` and `\def` within options setting macros. This way we can control from outside these macros whether the respective `\let` or `\def` is group-local or global. By default they are defined to be group-local.

```
593 \ifcsundef{fc@gl@let}{\global\let\fc@gl@let\let}{\PackageError{fmtcount}{Command already defined}}
594 \protect\fc@gl@let\space already defined.}}
595 \ifcsundef{fc@gl@def}{\global\let\fc@gl@def\def}{\PackageError{fmtcount}{Command already defined}}
596 \protect\fc@gl@def\space already defined.}}
```

Options for controlling plural mark. First of all we define some temporary macro `\fc@french@set@plural` in order to factorize code that defines an plural mark option:

```
#1 key name,
#2 key value,
#3 configuration index for 'reformed',
#4 configuration index for 'traditional',
#5 configuration index for 'reformed o', and
#6 configuration index for 'traditional o'.
597 \gdef\fc@french@set@plural#1#2#3#4#5#6{%
598   \ifthenelse{\equal{#2}{reformed}}{%
599     \expandafter\fc@gl@def\csname fc@frenchoptions@#1@plural\endcsname{#3}%
600   }{%
601     \ifthenelse{\equal{#2}{traditional}}{%
602       \expandafter\fc@gl@def\csname fc@frenchoptions@#1@plural\endcsname{#4}%
603     }{%
604       \ifthenelse{\equal{#2}{reformed o}}{%
605         \expandafter\fc@gl@def\csname fc@frenchoptions@#1@plural\endcsname{#5}%
606       }{%
607         \ifthenelse{\equal{#2}{traditional o}}{%
608           \expandafter\fc@gl@def\csname fc@frenchoptions@#1@plural\endcsname{#6}%
609         }{%
610           \ifthenelse{\equal{#2}{always}}{%
611             \expandafter\fc@gl@def\csname fc@frenchoptions@#1@plural\endcsname{0}%
612           }{%
613             \ifthenelse{\equal{#2}{never}}{%
614               \expandafter\fc@gl@def\csname fc@frenchoptions@#1@plural\endcsname{1}%
615             }{%
616               \ifthenelse{\equal{#2}{multiple}}{%
617                 \expandafter\fc@gl@def\csname fc@frenchoptions@#1@plural\endcsname{2}%
618               }{%
619                 \ifthenelse{\equal{#2}{multiple g-last}}{%
620                   \expandafter\fc@gl@def\csname fc@frenchoptions@#1@plural\endcsname{3}%
621                 }{%
622                   \ifthenelse{\equal{#2}{multiple l-last}}{%
623                     \expandafter\fc@gl@def\csname fc@frenchoptions@#1@plural\endcsname{4}%
624                   }{%
625                     \ifthenelse{\equal{#2}{multiple lng-last}}{%
626                       \expandafter\fc@gl@def\csname fc@frenchoptions@#1@plural\endcsname{5}%
627                     }{%
```



```

660 \let\fc@frenchoptions@supermillion@dos\space%
661 \let\fc@frenchoptions@submillion@dos\space
662 }{%
663 \ifthenelse{\equal{#1}{reformed}}\or\equal{#1}{1990}}{%
664 \let\fc@frenchoptions@supermillion@dos\space
665 \def\fc@frenchoptions@submillion@dos{-}%
666 }{%
667 \ifthenelse{\equal{#1}{always}}{%
668 \def\fc@frenchoptions@supermillion@dos{-}%
669 \def\fc@frenchoptions@submillion@dos{-}%
670 }{%
671 \PackageError{fmtcount}{Unexpected argument}{%
672 French option ‘dash or space’ expects ‘always’, ‘reformed’ or ‘traditional’
673 }
674 }%
675 }%
676 }%
677 }%

```

Option ‘scale’, can take 3 possible values:

- long for which  $\langle n \rangle$ illions &  $\langle n \rangle$ illiards are used with  $10^{6 \times n} = 1 \langle n \rangle$ illion, and  $10^{6 \times n + 3} = 1 \langle n \rangle$ illiard
- short for which  $\langle n \rangle$ illions only are used with  $10^{3 \times n + 3} = 1 \langle n \rangle$ illion
- recursive for which  $10^{18} =$  un milliard de milliards

```

678 \define@key{fcfrench}{scale}[recursive]{%
679 \ifthenelse{\equal{#1}{long}}{%
680 \let\fc@poweroften\fc@@pot@longscalefrench
681 }{%
682 \ifthenelse{\equal{#1}{recursive}}{%
683 \let\fc@poweroften\fc@@pot@recursivefrench
684 }{%
685 \ifthenelse{\equal{#1}{short}}{%
686 \let\fc@poweroften\fc@@pot@shortscalefrench
687 }{%
688 \PackageError{fmtcount}{Unexpected argument}{%
689 French option ‘scale’ expects ‘long’, ‘recursive’ or ‘short’
690 }
691 }%
692 }%
693 }%
694 }%

```

Option ‘n-illiard upto’ is ignored if ‘scale’ is different from ‘long’. It can take the following values:

- infinity in that case  $\langle n \rangle$ illard are never disabled,
- infty this is just a shorthand for ‘infinity’, and
- $n$  any integer that is such that  $n > 0$ , and that  $\forall k \in \mathbb{N}, k \geq n$ , number  $10^{6 \times k + 3}$  will be formatted as “mille  $\langle n \rangle$ illions”

```

695 \define@key{fcfrench}{n-illiard upto}[infinity]{%

```

```

696 \ifthenelse{\equal{#1}{infinity}}{%
697   \def\fc@longscale@milliard@upto{0}%
698 }{%
699   \ifthenelse{\equal{#1}{infy}}{%
700     \def\fc@longscale@milliard@upto{0}%
701   }{%
702     \if Q\ifnum9<1#1Q\fi\else
703     \PackageError{fmtcount}{Unexpected argument}{%
704       French option ‘milliard threshold’ expects ‘infinity’, or equivalently ‘infy’, or a no
705       integer.}%
706     \fi
707     \def\fc@longscale@milliard@upto{#1}%
708   }}%
709 }%

```

Now, the options ‘france’, ‘swiss’ and ‘belgian’ are defined to select the dialect to use.

Macro \@tempa is just a local shorthand to define each one of this option.

```

710 \def\@tempa#1{%
711   \define@key{fcfrench}{#1}[]{%
712     \PackageError{fmtcount}{Unexpected argument}{French option with key ‘#1’ does not take
713     any value}}%
714   \csgdef{KV@fcfrench@#1@default}{%
715     \fc@gl@def\fmtcount@french{#1}}%
716 }%
717 \@tempa{france}\@tempa{swiss}\@tempa{belgian}%

```

Make ‘france’ the default dialect for ‘french’ language

```

718 \gdef\fmtcount@french{france}%

```

Now, option ‘dialect’ is now defined so that ‘france’, ‘swiss’ and ‘belgian’ can also be used as key values, which is more conventional although less concise.

```

719 \define@key{fcfrench}{dialect}[france]{%
720   \ifthenelse{\equal{#1}{france}
721     \or\equal{#1}{swiss}
722     \or\equal{#1}{belgian}}{%
723     \def\fmtcount@french{#1}}{%
724     \PackageError{fmtcount}{Invalid value ‘#1’ to french option dialect key}
725     {Option ‘french’ can only take the values ‘france’,
726     ‘belgian’ or ‘swiss’}}}%
727 \expandafter\@tempb\csname KV@fcfrench@dialect\endcsname

```

The option `mil plural mark` allows to make the plural of `mil` to be regular, i.e. `mils`, instead of `mille`. By default it is ‘le’.

```

728 \define@key{fcfrench}{mil plural mark}[le]{%
729   \def\fc@frenchoptions@mil@plural@mark{#1}}
730 \expandafter\@tempb\csname KV@fcfrench@mil plural mark\endcsname

```

Definition of case handling macros. This should be moved somewhere else to be commonalized between all languages.

The macro `\fc@UpperCaseFirstLetter` is such that `\fc@UpperCaseFirstLetter<word>\@nil` expands to `\word` with first letter capitalized and remainder unchanged.

```

731 \gdef\fc@UpperCaseFirstLetter#1#2\@nil{%
732   \uppercase{#1}#2}

\fc@CaseIden  The macro \fc@CaseIden is such that \fc@CaseIden<word>\@nil expands to \word un-
              changed.
733 \gdef\fc@CaseIden#1\@nil{%
734   #1%
735 }%

\fc@UpperCaseAll  The macro \fc@UpperCaseAll is such that \fc@UpperCaseAll<word>\@nil expands to
                  \word all capitalized.
736 \gdef\fc@UpperCaseAll#1\@nil{%
737   \uppercase{#1}%
738 }%

\fc@wcase  The macro \fc@wcase is the capitalizing macro for word-by-word capitalization. By default
           we set it to identity, ie. no capitalization.
739 \global\let\fc@wcase\fc@CaseIden

\fc@gcase  The macro \fc@gcase is the capitalizing macro for global (the completed number) capital-
           ization. By default we set it to identity, ie. no capitalization.
740 \global\let\fc@gcase\fc@CaseIden

\fc@apply@gcase  The macro \fc@apply@gcase simply applies \fc@gcase to \@tempa, knowing that \@tempa
                 is the macro containing the result of formatting.
741 \gdef\fc@apply@gcase{%
           First of all we expand whatever \fc@wcase...\@nil found within \@tempa.
742   \protected@edef\@tempa{\@tempa}%
743   \protected@edef\@tempa{\expandafter\fc@gcase\@tempa\@nil}%
744 }

\ordinalMfrench
745 \newcommand*{\@ordinalMfrench}[2]{%
746   \iffmtord@abbrv
747   \ifnum#1=1 %
748     \edef#2{\number#1\relax\noexpand\fmtord{er}}%
749   \else
750     \edef#2{\number#1\relax\noexpand\fmtord{e}}%
751   \fi
752 \else
753   \PackageWarning{fmtcount}{Non abbreviated ordinal finals ('eme) are
754     considered incorrect in French.}%
755   \ifnum#1=1 %
756     \edef#2{\number#1\relax\noexpand\fmtord{er}}%
757   \else
758     \protected@edef#2{\number#1\relax\noexpand\fmtord{\protect\'eme}}%
759   \fi
760 \fi}
761 \global\let\@ordinalMfrench\@ordinalMfrench

```

@ordinalFfrench

```
762 \newcommand*{\@ordinalFfrench}[2]{%
763 \iffmtord@abbrv
764 \ifnum#1=1 %
765 \edef#2{\number#1\relax\noexpand\fmtord{re}}%
766 \else
767 \edef#2{\number#1\relax\noexpand\fmtord{e}}%
768 \fi
769 \else
770 \PackageWarning{fmtcount}{Non abbreviated ordinal finals ('eme) are
771 considered incorrect in French.}%
772 \ifnum#1=1 %
773 \protected@edef#2{\number#1\relax\noexpand\fmtord{\protect\'ere}}%
774 \else
775 \protected@edef#2{\number#1\relax\noexpand\fmtord{\protect\'eme}}%
776 \fi
777 \fi}
778 \global\let\@ordinalFfrench\@ordinalFfrench
In French neutral gender and masculine gender are formally identical.
779 \global\let\@ordinalNfrench\@ordinalMfrench
```

@unitstringfrench

```
780 \newcommand*{\@unitstringfrench}[1]{%
781 \noexpand\fc@wcase
782 \ifcase#1 %
783 z\'ero%
784 \or un%
785 \or deux%
786 \or trois%
787 \or quatre%
788 \or cinq%
789 \or six%
790 \or sept%
791 \or huit%
792 \or neuf%
793 \fi
794 \noexpand\@nil
795 }%
796 \global\let\@unitstringfrench\@unitstringfrench
```

@tenstringfrench

```
797 \newcommand*{\@tenstringfrench}[1]{%
798 \noexpand\fc@wcase
799 \ifcase#1 %
800 \or dix%
801 \or vingt%
802 \or trente%
803 \or quarante%
804 \or cinquante%
```

```

805 \or soixante%
806 \or septante%
807 \or huitante%
808 \or nonante%
809 \or cent%
810 \fi
811 \noexpand\@nil
812 }%
813 \global\let\@@tenstringfrench\@tenstringfrench

```

teenstringfrench

```

814 \newcommand*\@@teenstringfrench}[1]{%
815 \noexpand\fc@wcase
816 \ifcase#1 %
817     dix%
818 \or onze%
819 \or douze%
820 \or treize%
821 \or quatorze%
822 \or quinze%
823 \or seize%
824 \or dix\noexpand\@nil-\noexpand\fc@wcase sept%
825 \or dix\noexpand\@nil-\noexpand\fc@wcase huit%
826 \or dix\noexpand\@nil-\noexpand\fc@wcase neuf%
827 \fi
828 \noexpand\@nil
829 }%
830 \global\let\@@teenstringfrench\@teenstringfrench

```

seventiesfrench

```

831 \newcommand*\@@seventiesfrench}[1]{%
832 \@tenstring{6}%
833 \ifnum#1=1 %
834 \fc@frenchoptions@submillion@dos\@andname\fc@frenchoptions@submillion@dos
835 \else
836 -%
837 \fi
838 \@teenstring{#1}%
839 }%
840 \global\let\@@seventiesfrench\@seventiesfrench

```

@eightiesfrench

Macro \@@eightiesfrench is used to format numbers in the interval [80..89]. Argument as follows:

#1 digit  $d_w$  such that the number to be formatted is  $80 + d_w$

Implicit arguments as:

`\count0` weight  $w$  of the number  $d_{w+1}d_w$  to be formatted  
`\count1` same as `\#1`  
`\count6` input, counter giving the least weight of non zero digits in top level formatted number integral part, with rounding down to a multiple of 3,  
`\count9` input, counter giving the power type of the power of ten following the eighties to be formatted; that is '1' for "mil" and '2' for "<n>illion|<n>illiard".

```

841 \newcommand*{\@@eightiesfrench[1]}{
842 \fc@wcase quatre\@nil-\noexpand\fc@wcase vingt%
843 \ifnum#1>0 %
844 \ifnum\fc@frenchoptions@vingt@plural=0 % vingt plural=always
845 s%
846 \fi
847 \noexpand\@nil
848 -\@unitstring{#1}%
849 \else
850 \ifcase\fc@frenchoptions@vingt@plural\space
851 s% 0: always
852 \or
853 % 1: never
854 \or
855 s% 2: multiple
856 \or
857 % 3: multiple g-last
858 \ifnum\count0=\count6\ifnum\count9=0 s\fi\fi
859 \or
860 % 4: multiple l-last
861 \ifnum\count9=1 %
862 \else
863 s%
864 \fi
865 \or
866 % 5: multiple lng-last
867 \ifnum\count9=1 %
868 \else
869 \ifnum\count0>0 %
870 s%
871 \fi
872 \fi
873 \or
874 % or 6: multiple ng-last
875 \ifnum\count0>0 %
876 s%
877 \fi
878 \fi
879 \noexpand\@nil
880 \fi
881 }%
882 \global\let\@@eightiesfrench\@@eightiesfrench
883 \newcommand*{\@@ninetiesfrench}[1]{

```

```

884 \fc@wcase quatre\@nil-\noexpand\fc@wcase vingt%
885 \ifnum\fc@frenchoptions@vingt@plural=0 % vingt plural=always
886 s%
887 \fi
888 \noexpand\@nil
889 -\@teenstring{#1}%
890 }%
891 \global\let\@@ninetiesfrench\@@ninetiesfrench
892 \newcommand*\@@seventiesfrenchswiss}[1]{%
893 \@tenstring{7}%
894 \ifnum#1=1\ \@andname\ \fi
895 \ifnum#1>1-\fi
896 \ifnum#1>0 \@unitstring{#1}\fi
897 }%
898 \global\let\@@seventiesfrenchswiss\@@seventiesfrenchswiss
899 \newcommand*\@@eightiesfrenchswiss}[1]{%
900 \@tenstring{8}%
901 \ifnum#1=1\ \@andname\ \fi
902 \ifnum#1>1-\fi
903 \ifnum#1>0 \@unitstring{#1}\fi
904 }%
905 \global\let\@@eightiesfrenchswiss\@@eightiesfrenchswiss
906 \newcommand*\@@ninetiesfrenchswiss}[1]{%
907 \@tenstring{9}%
908 \ifnum#1=1\ \@andname\ \fi
909 \ifnum#1>1-\fi
910 \ifnum#1>0 \@unitstring{#1}\fi
911 }%
912 \global\let\@@ninetiesfrenchswiss\@@ninetiesfrenchswiss

```

c@french@common Macro \fc@french@common does all the preliminary settings common to all French dialects & formatting options.

```

913 \newcommand*\fc@french@common{%
914 \let\fc@wcase\fc@CaseIden
915 \let\@unitstring=\@unitstringfrench
916 \let\@teenstring=\@teenstringfrench
917 \let\@tenstring=\@tenstringfrench
918 \def\@hundred{cent}%
919 \def\@andname{et}%
920 }%
921 \global\let\fc@french@common\fc@french@common
922 \newcommand*\@numberstringMfrenchswiss}[2]{%
923 \fc@french@common
924 \let\fc@gcase\fc@CaseIden
925 \let\@seventies=\@seventiesfrenchswiss
926 \let\@eighties=\@eightiesfrenchswiss
927 \let\@nineties=\@ninetiesfrenchswiss
928 \let\fc@nbrstr@preamble\@empty
929 \let\fc@nbrstr@postamble\@empty

```

```

930 \@numberstringfrench{#1}{#2}}
931 \global\let\@numberstringMfrenchswiss\@numberstringMfrenchswiss
932 \newcommand*{\@numberstringMfrenchfrance}[2]{%
933 \fc@french@common
934 \let\fc@gcase\fc@CaseIden
935 \let\@seventies=\@@seventiesfrench
936 \let\@eighties=\@@eightiesfrench
937 \let\@nineties=\@@ninetiesfrench
938 \let\fc@nbrstr@preamble\@empty
939 \let\fc@nbrstr@postamble\@empty
940 \@numberstringfrench{#1}{#2}}
941 \global\let\@numberstringMfrenchfrance\@numberstringMfrenchfrance
942 \newcommand*{\@numberstringMfrenchbelgian}[2]{%
943 \fc@french@common
944 \let\fc@gcase\fc@CaseIden
945 \let\@seventies=\@@seventiesfrenchswiss
946 \let\@eighties=\@@eightiesfrench
947 \let\@nineties=\@@ninetiesfrench
948 \let\fc@nbrstr@preamble\@empty
949 \let\fc@nbrstr@postamble\@empty
950 \@numberstringfrench{#1}{#2}}
951 \global\let\@numberstringMfrenchbelgian\@numberstringMfrenchbelgian
952 \let\@numberstringMfrench=\@numberstringMfrenchfrance
953 \newcommand*{\@numberstringFfrenchswiss}[2]{%
954 \fc@french@common
955 \let\fc@gcase\fc@CaseIden
956 \let\@seventies=\@@seventiesfrenchswiss
957 \let\@eighties=\@@eightiesfrenchswiss
958 \let\@nineties=\@@ninetiesfrenchswiss
959 \let\fc@nbrstr@preamble\fc@@nbrstr@Fpreamble
960 \let\fc@nbrstr@postamble\@empty
961 \@numberstringfrench{#1}{#2}}
962 \global\let\@numberstringFfrenchswiss\@numberstringFfrenchswiss
963 \newcommand*{\@numberstringFfrenchfrance}[2]{%
964 \fc@french@common
965 \let\fc@gcase\fc@CaseIden
966 \let\@seventies=\@@seventiesfrench
967 \let\@eighties=\@@eightiesfrench
968 \let\@nineties=\@@ninetiesfrench
969 \let\fc@nbrstr@preamble\fc@@nbrstr@Fpreamble
970 \let\fc@nbrstr@postamble\@empty
971 \@numberstringfrench{#1}{#2}}
972 \global\let\@numberstringFfrenchfrance\@numberstringFfrenchfrance
973 \newcommand*{\@numberstringFfrenchbelgian}[2]{%
974 \fc@french@common
975 \let\fc@gcase\fc@CaseIden
976 \let\@seventies=\@@seventiesfrenchswiss
977 \let\@eighties=\@@eightiesfrench
978 \let\@nineties=\@@ninetiesfrench

```

```

979 \let\fc@nbrstr@preamble\fc@@nbrstr@Fpreamble
980 \let\fc@nbrstr@postamble\@empty
981 \@@numberstringfrench{#1}{#2}}
982 \global\let\@numberstringFfrenchbelgian\@numberstringFfrenchbelgian
983 \global\let\@numberstringFfrench=\@numberstringFfrenchfrance
984 \global\let\@ordinalstringNfrench\@ordinalstringMfrench
985 \newcommand*\@NumberstringMfrenchswiss}[2]{%
986 \fc@french@common
987 \let\fc@gcase\fc@UpperCaseFirstLetter
988 \let\@seventies=\@@seventiesfrenchswiss
989 \let\@eighties=\@@eightiesfrenchswiss
990 \let\@nineties=\@@ninetiesfrenchswiss
991 \let\fc@nbrstr@preamble\@empty
992 \let\fc@nbrstr@postamble\fc@apply@gcase
993 \@@numberstringfrench{#1}{#2}}
994 \global\let\@NumberstringMfrenchswiss\@NumberstringMfrenchswiss
995 \newcommand*\@NumberstringMfrenchfrance}[2]{%
996 \fc@french@common
997 \let\fc@gcase\fc@UpperCaseFirstLetter
998 \let\@seventies=\@@seventiesfrench
999 \let\@eighties=\@@eightiesfrench
1000 \let\@nineties=\@@ninetiesfrench
1001 \let\fc@nbrstr@preamble\@empty
1002 \let\fc@nbrstr@postamble\fc@apply@gcase
1003 \@@numberstringfrench{#1}{#2}}
1004 \global\let\@NumberstringMfrenchfrance\@NumberstringMfrenchfrance
1005 \newcommand*\@NumberstringMfrenchbelgian}[2]{%
1006 \fc@french@common
1007 \let\fc@gcase\fc@UpperCaseFirstLetter
1008 \let\@seventies=\@@seventiesfrenchswiss
1009 \let\@eighties=\@@eightiesfrench
1010 \let\@nineties=\@@ninetiesfrench
1011 \let\fc@nbrstr@preamble\@empty
1012 \let\fc@nbrstr@postamble\fc@apply@gcase
1013 \@@numberstringfrench{#1}{#2}}
1014 \global\let\@NumberstringMfrenchbelgian\@NumberstringMfrenchbelgian
1015 \global\let\@NumberstringMfrench=\@NumberstringMfrenchfrance
1016 \newcommand*\@NumberstringFfrenchswiss}[2]{%
1017 \fc@french@common
1018 \let\fc@gcase\fc@UpperCaseFirstLetter
1019 \let\@seventies=\@@seventiesfrenchswiss
1020 \let\@eighties=\@@eightiesfrenchswiss
1021 \let\@nineties=\@@ninetiesfrenchswiss
1022 \let\fc@nbrstr@preamble\fc@@nbrstr@Fpreamble
1023 \let\fc@nbrstr@postamble\fc@apply@gcase
1024 \@@numberstringfrench{#1}{#2}}
1025 \global\let\@NumberstringFfrenchswiss\@NumberstringFfrenchswiss
1026 \newcommand*\@NumberstringFfrenchfrance}[2]{%
1027 \fc@french@common

```

```

1028 \let\fc@gcase\fc@UpperCaseFirstLetter
1029 \let\@seventies=\@seventiesfrench
1030 \let\@eighties=\@eightiesfrench
1031 \let\@nineties=\@ninetiesfrench
1032 \let\fc@nbrstr@preamble\fc@nbrstr@Fpreamble
1033 \let\fc@nbrstr@postamble\fc@apply@gcase
1034 \@@numberstringfrench{#1}{#2}}
1035 \global\let\@NumberstringFfrenchfrance\@NumberstringFfrenchfrance
1036 \newcommand*\@NumberstringFfrenchbelgian}[2]{%
1037 \fc@french@common
1038 \let\fc@gcase\fc@UpperCaseFirstLetter
1039 \let\@seventies=\@seventiesfrenchswiss
1040 \let\@eighties=\@eightiesfrench
1041 \let\@nineties=\@ninetiesfrench
1042 \let\fc@nbrstr@preamble\fc@nbrstr@Fpreamble
1043 \let\fc@nbrstr@postamble\fc@apply@gcase
1044 \@@numberstringfrench{#1}{#2}}
1045 \global\let\@NumberstringFfrenchbelgian\@NumberstringFfrenchbelgian
1046 \global\let\@NumberstringFfrench=\@NumberstringFfrenchfrance
1047 \global\let\@NumberstringNfrench\@NumberstringMfrench
1048 \newcommand*\@ordinalstringMfrenchswiss}[2]{%
1049 \fc@french@common
1050 \let\fc@gcase\fc@CaseIden
1051 \let\fc@first\fc@@firstfrench
1052 \let\@seventies=\@seventiesfrenchswiss
1053 \let\@eighties=\@eightiesfrenchswiss
1054 \let\@nineties=\@ninetiesfrenchswiss
1055 \@@ordinalstringfrench{#1}{#2}%
1056 }%
1057 \global\let\@ordinalstringMfrenchswiss\@ordinalstringMfrenchswiss
1058 \newcommand*\fc@@firstfrench{premier}
1059 \global\let\fc@@firstfrench\fc@@firstfrench

1060 \newcommand*\fc@@firstFfrench{premi\protect\`ere}
1061 \global\let\fc@@firstFfrench\fc@@firstFfrench
1062 \newcommand*\@ordinalstringMfrenchfrance}[2]{%
1063 \fc@french@common
1064 \let\fc@gcase\fc@CaseIden
1065 \let\fc@first=\fc@@firstfrench
1066 \let\@seventies=\@seventiesfrench
1067 \let\@eighties=\@eightiesfrench
1068 \let\@nineties=\@ninetiesfrench
1069 \@@ordinalstringfrench{#1}{#2}}
1070 \global\let\@ordinalstringMfrenchfrance\@ordinalstringMfrenchfrance
1071 \newcommand*\@ordinalstringMfrenchbelgian}[2]{%
1072 \fc@french@common
1073 \let\fc@gcase\fc@CaseIden
1074 \let\fc@first=\fc@@firstfrench
1075 \let\@seventies=\@seventiesfrench
1076 \let\@eighties=\@eightiesfrench

```

```

1077 \let\@nineties=\@ninetiesfrench
1078 \@ordinalstringfrench{#1}{#2}%
1079 }%
1080 \global\let\@ordinalstringMfrenchbelgian\@ordinalstringMfrenchbelgian
1081 \global\let\@ordinalstringMfrench=\@ordinalstringMfrenchfrance
1082 \newcommand*\@ordinalstringFfrenchswiss}[2]{%
1083 \fc@french@common
1084 \let\fc@gcase\fc@CaseIden
1085 \let\fc@first\fc@@firstFfrench
1086 \let\@seventies=\@seventiesfrenchswiss
1087 \let\@eighties=\@eightiesfrenchswiss
1088 \let\@nineties=\@ninetiesfrenchswiss
1089 \@ordinalstringfrench{#1}{#2}%
1090 }%
1091 \global\let\@ordinalstringFfrenchswiss\@ordinalstringFfrenchswiss
1092 \newcommand*\@ordinalstringFfrenchfrance}[2]{%
1093 \fc@french@common
1094 \let\fc@gcase\fc@CaseIden
1095 \let\fc@first=\fc@@firstFfrench
1096 \let\@seventies=\@seventiesfrench
1097 \let\@eighties=\@eightiesfrench
1098 \let\@nineties=\@ninetiesfrench
1099 \@ordinalstringfrench{#1}{#2}%
1100 }%
1101 \global\let\@ordinalstringFfrenchfrance\@ordinalstringFfrenchfrance
1102 \newcommand*\@ordinalstringFfrenchbelgian}[2]{%
1103 \fc@french@common
1104 \let\fc@gcase\fc@CaseIden
1105 \let\fc@first=\fc@@firstFfrench
1106 \let\@seventies=\@seventiesfrench
1107 \let\@eighties=\@eightiesfrench
1108 \let\@nineties=\@ninetiesfrench
1109 \@ordinalstringfrench{#1}{#2}%
1110 }%
1111 \global\let\@ordinalstringFfrenchbelgian\@ordinalstringFfrenchbelgian
1112 \global\let\@ordinalstringFfrench=\@ordinalstringFfrenchfrance
1113 \global\let\@ordinalstringNfrench\@ordinalstringMfrench
1114 \newcommand*\@OrdinalstringMfrenchswiss}[2]{%
1115 \fc@french@common
1116 \let\fc@gcase\fc@UpperCaseFirstLetter
1117 \let\fc@first=\fc@@firstfrench
1118 \let\@seventies=\@seventiesfrenchswiss
1119 \let\@eighties=\@eightiesfrenchswiss
1120 \let\@nineties=\@ninetiesfrenchswiss
1121 \@ordinalstringfrench{#1}{#2}%
1122 }%
1123 \global\let\@OrdinalstringMfrenchswiss\@OrdinalstringMfrenchswiss
1124 \newcommand*\@OrdinalstringMfrenchfrance}[2]{%
1125 \fc@french@common

```

```

1126 \let\fc@gcase\fc@UpperCaseFirstLetter
1127 \let\fc@first\fc@@firstfrench
1128 \let\@seventies=\@seventiesfrench
1129 \let\@eighties=\@eightiesfrench
1130 \let\@nineties=\@ninetiesfrench
1131 \@@ordinalstringfrench{#1}{#2}%
1132 }%
1133 \global\let\@OrdinalstringMfrenchfrance\@OrdinalstringMfrenchfrance
1134 \newcommand*{\@OrdinalstringMfrenchbelgian}[2]{%
1135 \fc@french@common
1136 \let\fc@gcase\fc@UpperCaseFirstLetter
1137 \let\fc@first\fc@@firstfrench
1138 \let\@seventies=\@seventiesfrench
1139 \let\@eighties=\@eightiesfrench
1140 \let\@nineties=\@ninetiesfrench
1141 \@@ordinalstringfrench{#1}{#2}%
1142 }%
1143 \global\let\@OrdinalstringMfrenchbelgian\@OrdinalstringMfrenchbelgian
1144 \global\let\@OrdinalstringMfrench=\@OrdinalstringMfrenchfrance
1145 \newcommand*{\@OrdinalstringFfrenchswiss}[2]{%
1146 \fc@french@common
1147 \let\fc@gcase\fc@UpperCaseFirstLetter
1148 \let\fc@first\fc@@firstfrench
1149 \let\@seventies=\@seventiesfrenchswiss
1150 \let\@eighties=\@eightiesfrenchswiss
1151 \let\@nineties=\@ninetiesfrenchswiss
1152 \@@ordinalstringfrench{#1}{#2}%
1153 }%
1154 \global\let\@OrdinalstringFfrenchswiss\@OrdinalstringFfrenchswiss
1155 \newcommand*{\@OrdinalstringFfrenchfrance}[2]{%
1156 \fc@french@common
1157 \let\fc@gcase\fc@UpperCaseFirstLetter
1158 \let\fc@first\fc@@firstFfrench
1159 \let\@seventies=\@seventiesfrench
1160 \let\@eighties=\@eightiesfrench
1161 \let\@nineties=\@ninetiesfrench
1162 \@@ordinalstringfrench{#1}{#2}%
1163 }%
1164 \global\let\@OrdinalstringFfrenchfrance\@OrdinalstringFfrenchfrance
1165 \newcommand*{\@OrdinalstringFfrenchbelgian}[2]{%
1166 \fc@french@common
1167 \let\fc@gcase\fc@UpperCaseFirstLetter
1168 \let\fc@first\fc@@firstFfrench
1169 \let\@seventies=\@seventiesfrench
1170 \let\@eighties=\@eightiesfrench
1171 \let\@nineties=\@ninetiesfrench
1172 \@@ordinalstringfrench{#1}{#2}%
1173 }%
1174 \global\let\@OrdinalstringFfrenchbelgian\@OrdinalstringFfrenchbelgian

```

```

1175 \global\let\@OrdinalstringFfrench=\@OrdinalstringFfrenchfrance
1176 \global\let\@OrdinalstringNfrench\@OrdinalstringMfrench

```

`\fc@@do@plural@mark` Macro `\fc@@do@plural@mark` will expand to the plural mark of  $\langle n \rangle$ illiard,  $\langle n \rangle$ illion, mil, cent or vingt, whichever is applicable. First check that the macro is not yet defined.

```

1177 \ifcsundef{fc@@do@plural@mark}{}%
1178 {\PackageError{fmtcount}{Duplicate definition}{Redefinition of macro
1179   'fc@@do@plural@mark'}}

```

Arguments as follows:

#1 plural mark, 's' in general, but for mil it is `\fc@frenchoptions@mil@plural@mark`

Implicit arguments as follows:

- `\count0` input, counter giving the weight  $w$ , this is expected to be multiple of 3,
- `\count1` input, counter giving the plural value of multiplied object  $\langle n \rangle$ illiard,  $\langle n \rangle$ illion, mil, cent or vingt, whichever is applicable, that is to say it is 1 when the considered objet is not multiplied, and 2 or more when it is multiplied,
- `\count6` input, counter giving the least weight of non zero digits in top level formatted number integral part, with rounding down to a multiple of 3,
- `\count10` input, counter giving the plural mark control option.

```

1180 \def\fc@@do@plural@mark#1{%
1181   \ifcase\count10 %
1182     #1% 0=always
1183     \or% 1=never
1184     \or% 2=multiple
1185     \ifnum\count1>1 %
1186       #1%
1187       \fi
1188     \or% 3= multiple g-last
1189     \ifnum\count1>1 %
1190       \ifnum\count0=\count6 %
1191         #1%
1192         \fi
1193       \fi
1194     \or% 4= multiple l-last
1195     \ifnum\count1>1 %
1196       \ifnum\count9=1 %
1197         \else
1198         #1%
1199         \fi
1200       \fi
1201     \or% 5= multiple lng-last
1202     \ifnum\count1>1 %
1203       \ifnum\count9=1 %
1204         \else
1205         \if\count0>\count6 %
1206         #1%
1207         \fi
1208       \fi
1209     \fi

```

```

1210 \or% 6= multiple ng-last
1211 \ifnum\count1>1 %
1212 \ifnum\count0>\count6 %
1213 #1%
1214 \fi
1215 \fi
1216 \fi
1217 }%
1218 \global\let\fc@@do@plural@mark\fc@@do@plural@mark

```

`\fc@@nbrstr@Fpreamble` Macro `\fc@@nbrstr@Fpreamble` do the necessary preliminaries before formatting a cardinal with feminine gender.

```

1219 \ifcsundef\fc@@nbrstr@Fpreamble\{}{%
1220 \PackageError{fmtcount}{Duplicate definition}{Redefinition of macro
1221 'fc@@nbrstr@Fpreamble'}}

```

`\fc@@nbrstr@Fpreamble`

```

1222 \def\fc@@nbrstr@Fpreamble{%
1223 \fc@read@unit{\count1}{0}%
1224 \ifnum\count1=1 %
1225 \let\fc@wcase@save\fc@wcase
1226 \def\fc@wcase{\noexpand\fc@wcase}%
1227 \def\@nil{\noexpand\@nil}%
1228 \let\fc@nbrstr@postamble\fc@@nbrstr@Fpostamble
1229 \fi
1230 }%
1231 \global\let\fc@@nbrstr@Fpreamble\fc@@nbrstr@Fpreamble

```

`\fc@@nbrstr@Fpostamble`

```

1232 \def\fc@@nbrstr@Fpostamble{%
1233 \let\fc@wcase\fc@wcase@save
1234 \expandafter\fc@get@last@word\expandafter{\@tempa}\@tempb\@tempc
1235 \def\@tempd{un}%
1236 \ifx\@tempc\@tempd
1237 \let\@tempc\@tempa
1238 \edef\@tempa{\@tempb\fc@wcase une\@nil}%
1239 \fi
1240 }%
1241 \global\let\fc@@nbrstr@Fpostamble\fc@@nbrstr@Fpostamble

```

`\fc@@pot@longscalefrench` Macro `\fc@@pot@longscalefrench` is used to produce powers of ten with long scale convention. The long scale convention is correct for French and elsewhere in Europe. First we check that the macro is not yet defined.

```

1242 \ifcsundef\fc@@pot@longscalefrench\{}{%
1243 \PackageError{fmtcount}{Duplicate definition}{Redefinition of macro
1244 'fc@@pot@longscalefrench'}}

```

Argument are as follows:

#1 input, plural value of  $d$ , that is to say: let  $d$  be the number multiplying the considered power of ten, then the plural value #2 is expected to be 0 if  $d = 0$ , 1 if  $d = 1$ , or  $> 1$  if  $d > 1$

#2 output, counter, maybe 0 when power of ten is 1, 1 when power of ten starts with “mil(le)”, or 2 when power of ten is a “ $\langle n \rangle$ illion(s)| $\langle n \rangle$ illiard(s)”

#3 output, macro into which to place the formatted power of ten

Implicit arguments as follows:

`\count0` input, counter giving the weight  $w$ , this is expected to be multiple of 3

```

1245 \def\fc@pot@longscalefrench#1#2#3{%
1246   {%

```

First the input arguments are saved into local objects: #1 and #1 are respectively saved into `\@tempa` and `\@tempb`.

```

1247   \edef\@tempb{\number#1}%

```

Let `\count1` be the plural value.

```

1248   \count1=\@tempb

```

Let  $n$  and  $r$  the the quotient and remainder of division of weight  $w$  by 6, that is to say  $w = n \times 6 + r$  and  $0 \leq r < 6$ , then `\count2` is set to  $n$  and `\count3` is set to  $r$ .

```

1249   \count2\count0 %
1250   \divide\count2 by 6 %
1251   \count3\count2 %
1252   \multiply\count3 by 6 %
1253   \count3-\count3 %
1254   \advance\count3 by \count0 %
1255   \ifnum\count0>0 %

```

If weight  $w$  (a.k.a. `\count0`) is such that  $w > 0$ , then  $w \geq 3$  because  $w$  is a multiple of 3. So we *may* have to append “mil(le)” or “ $\langle n \rangle$ illion(s)” or “ $\langle n \rangle$ illiard(s)”.

```

1256   \ifnum\count1>0 %

```

Plural value is  $> 0$  so have at least one “mil(le)” or “ $\langle n \rangle$ illion(s)” or “ $\langle n \rangle$ illiard(s)”. We need to distinguish between the case of “mil(le)” and that of “ $\langle n \rangle$ illion(s)” or “ $\langle n \rangle$ illiard(s)”, so we `\define \@tempb` to ‘1’ for “mil(le)”, and to ‘2’ otherwise.

```

1257   \edef\@tempb{%
1258     \ifnum\count2=0 % weight=3

```

Here  $n = 0$ , with  $n = w \div 6$ , but we also know that  $w \geq 3$ , so we have  $w = 3$  which means we are in the “mil(le)” case.

```

1259     1%
1260   \else
1261     \ifnum\count3>2 %

```

Here we are in the case of  $3 \leq r < 6$ , with  $r$  the remainder of division of weight  $w$  by 6, we should have “ $\langle n \rangle$ illiard(s)”, but that may also be “mil(le)” instead depending on option ‘n-illiard upto’, known as `\fc@longscale@nilliard@upto`.

```

1262     \ifnum\fc@longscale@nilliard@upto=0 %

```

Here option ‘n-illiard upto’ is ‘infinity’, so we always use “ $\langle n \rangle$ illiard(s)”.

```

1263     2%
1264   \else

```

Here option ‘n-illiard upto’ indicate some threshold to which to compare  $n$  (a.k.a.  $\backslash\text{count2}$ ).

```

1265         \ifnum\count2>\fc@longscale@nilliard@upto
1266             1%
1267         \else
1268             2%
1269         \fi
1270     \fi
1271 \else
1272     2%
1273 \fi
1274 \fi
1275 }%
1276 \ifnum\@temph=1 %

```

Here  $10^w$  is formatted as “mil(le)”.

```

1277     \count10=\fc@frenchoptions@mil@plural\space
1278     \edef\@tempe{%
1279         \noexpand\fc@wcase
1280         mil%
1281         \fc@@do@plural@mark\fc@frenchoptions@mil@plural@mark
1282         \noexpand\@nil
1283     }%
1284 \else
1285     % weight >= 6
1286     \expandafter\fc@@latin@cardinal@pefix\expandafter{\the\count2}\@tempg
1287     % now form the xxx-illion(s) or xxx-illiard(s) word
1288     \ifnum\count3>2 %
1289         \toks10{illiard}%
1290         \count10=\csname fc@frenchoptions@n-illiard@plural\endcsname\space
1291     \else
1292         \toks10{illion}%
1293         \count10=\csname fc@frenchoptions@n-illion@plural\endcsname\space
1294     \fi
1295     \edef\@tempe{%
1296         \noexpand\fc@wcase
1297         \@tempg
1298         \the\toks10 %
1299         \fc@@do@plural@mark s%
1300         \noexpand\@nil
1301     }%
1302     \fi
1303 \else

```

Here plural indicator of  $d$  indicates that  $d = 0$ , so we have  $0 \times 10^w$ , and it is not worth to format  $10^w$ , because there are none of them.

```

1304         \let\@tempe\@empty
1305         \def\@temph{0}%
1306     \fi
1307 \else

```

Case of  $w = 0$ .

```
1308 \let\@tempe\@empty
1309 \def\@temph{0}%
1310 \fi
```

Now place into `cs@tempa` the assignment of results `\@temph` and `\@tempe` to #2 and #3 for further propagation after closing brace.

```
1311 \expandafter\toks\expandafter1\expandafter{\@tempe}%
1312 \toks0{#2}%
1313 \edef\@tempa{\the\toks0 \@temph \def\noexpand#3{\the\toks1}}%
1314 \expandafter
1315 }\@tempa
1316 }%
1317 \global\let\fc@pot@longscalefrench\fc@pot@longscalefrench
```

`\fc@pot@shortscalefrench` Macro `\fc@pot@shortscalefrench` is used to produce powers of ten with short scale convention. This convention is the US convention and is not correct for French and elsewhere in Europe. First we check that the macro is not yet defined.

```
1318 \ifcsundef\fc@pot@shortscalefrench\{}{%
1319 \PackageError{fmtcount}{Duplicate definition}{Redefinition of macro
1320 'fc@pot@shortscalefrench'}}}
```

Arguments as follows — same interface as for `\fc@pot@longscalefrench`:

- #1 input, plural value of  $d$ , that is to say: let  $d$  be the number multiplying the considered power of ten, then the plural value #2 is expected to be 0 if  $d = 0$ , 1 if  $d = 1$ , or  $> 1$  if  $d > 1$
- #2 output, counter, maybe 0 when power of ten is 1, 1 when power of ten starts with “mil(le)”, or 2 when power of ten is a “ $\langle n \rangle$ illion(s)| $\langle n \rangle$ illiard(s)”
- #3 output, macro into which to place the formatted power of ten

Implicit arguments as follows:

`\count0` input, counter giving the weight  $w$ , this is expected to be multiple of 3

```
1321 \def\fc@pot@shortscalefrench#1#2#3{%
1322 {%
```

First save input arguments #1, #2, and #3 into local macros respectively `\@tempa`, `\@tempb`, `\@tempc` and `\@tempd`.

```
1323 \edef\@tempb{\number#1}%
```

And let `\count1` be the plural value.

```
1324 \count1=\@tempb
```

Now, let `\count2` be the integer  $n$  generating the pseudo latin prefix, i.e.  $n$  is such that  $w = 3 \times n + 3$ .

```
1325 \count2\count0 %
1326 \divide\count2 by 3 %
1327 \advance\count2 by -1 %
```

Here is the real job, the formatted power of ten will go to `\@tempe`, and its power type will go to `\@temph`. Please remember that the power type is an index in  $[0..2]$  indicating whether  $10^w$  is formatted as *nothing*, “mil(le)” or “ $\langle n \rangle$ illion(s)| $\langle n \rangle$ illiard(s)”.

```
1328 \ifnum\count0>0 % If weight>=3, i.e we do have to append thousand or n-illion(s)/n-illiard(s)
```

```

1329 \ifnum\count1>0 % we have at least one thousand/n-illion/n-illiard
1330 \ifnum\count2=0 %
1331 \def\@temph{1}%
1332 \count1=\fc@frenchoptions@mil@plural\space
1333 \edef\@tempe{%
1334 mil%
1335 \fc@@do@plural@mark\fc@frenchoptions@mil@plural@mark
1336 }%
1337 \else
1338 \def\@temph{2}%
1339 % weight >= 6
1340 \expandafter\fc@@latin@cardinal@prefix\expandafter{\the\count2}\@tempg
1341 \count10=\csname fc@frenchoptions@n-illion@plural\endcsname\space
1342 \edef\@tempe{%
1343 \noexpand\fc@wcase
1344 \@tempg
1345 illion%
1346 \fc@@do@plural@mark s%
1347 \noexpand\@nil
1348 }%
1349 \fi
1350 \else

```

Here we have  $d = 0$ , so nothing is to be formatted for  $d \times 10^w$ .

```

1351 \def\@temph{0}%
1352 \let\@tempe\@empty
1353 \fi
1354 \else

```

Here  $w = 0$ .

```

1355 \def\@temph{0}%
1356 \let\@tempe\@empty
1357 \fi
1358 % now place into \@tempa the assignment of results \cs{@temph} and \cs{@tempe} to to \text
1359 % \texttt{\#3} for further propagation after closing brace.
1360 % \begin{macrocode}
1361 \expandafter\toks\expandafter1\expandafter{\@tempe}%
1362 \toks0{\#2}%
1363 \edef\@tempa{\the\toks0 \@temph \def\noexpand#3{\the\toks1}}%
1364 \expandafter
1365 }\@tempa
1366 }%
1367 \global\let\fc@pot@shortscalefrench\fc@pot@shortscalefrench

```

Macro `\fc@pot@recursivefrench` is used to produce power of tens that are of the form “million de milliards de milliards” for  $10^{24}$ . First we check that the macro is not yet defined.

```

1368 \ifcsundef\fc@pot@recursivefrench\{}{%
1369 \PackageError{fmtcount}{Duplicate definition}{Redefinition of macro
1370 'fc@pot@recursivefrench'}}

```

The arguments are as follows — same interface as for `\fc@pot@longscalefrench`:

- #1 input, plural value of  $d$ , that is to say: let  $d$  be the number multiplying the considered power of ten, then the plural value #2 is expected to be 0 if  $d = 0$ , 1 if  $d = 1$ , or  $> 1$  if  $d > 1$
  - #2 output, counter, maybe 0 when power of ten is 1, 1 when power of ten starts with “mil(le)”, or 2 when power of ten is a “ $\langle n \rangle$ illion(s)| $\langle n \rangle$ illiard(s)”
  - #3 output, macro into which to place the formatted power of ten
- Implicit arguments as follows:

`\count0` input, counter giving the weight  $w$ , this is expected to be multiple of 3

```
1371 \def\fc@pot@recursivefrench#1#2#3{%
1372  {%
```

First the input arguments are saved into local objects: #1 and #1 are respectively saved into `\@tempa` and `\@tempb`.

```
1373  \edef\@tempb{\number#1}%
1374  \let\@tempa\@tempa
```

Now get the inputs #1 and #1 into counters `\count0` and `\count1` as this is more practical.

```
1375  \count1=\@tempb\space
```

Now compute into `\count2` how many times “de milliards” has to be repeated.

```
1376  \ifnum\count1>0 %
1377    \count2\count0 %
1378    \divide\count2 by 9 %
1379    \advance\count2 by -1 %
1380    \let\@tempe\@empty
1381    \edef\@tempf{\fc@frenchoptions@supermillion@dos
1382      de\fc@frenchoptions@supermillion@dos\fc@wcase milliards\@nil}%
1383    \count11\count0 %
1384    \ifnum\count2>0 %
1385      \count3\count2 %
1386      \count3-\count3 %
1387      \multiply\count3 by 9 %
1388      \advance\count11 by \count3 %
1389      \loop
1390        % (\count2, \count3) <- (\count2 div 2, \count2 mod 2)
1391        \count3\count2 %
1392        \divide\count3 by 2 %
1393        \multiply\count3 by 2 %
1394        \count3-\count3 %
1395        \advance\count3 by \count2 %
1396        \divide\count2 by 2 %
1397        \ifnum\count3=1 %
1398          \let\@tempg\@tempe
1399          \edef\@tempe{\@tempg\@tempf}%
1400        \fi
1401        \let\@tempg\@tempf
1402        \edef\@tempf{\@tempg\@tempg}%
1403        \ifnum\count2>0 %
1404          \repeat
1405        \fi
1406    \divide\count11 by 3 %
```

```

1407 \ifcase\count11 % 0 .. 5
1408   % 0 => d milliard(s) (de milliards)*
1409   \def\@temph{2}%
1410   \count10=\csname fc@frenchoptions@n-illiard@plural\endcsname\space
1411 \or % 1 => d mille milliard(s) (de milliards)*
1412   \def\@temph{1}%
1413   \count10=\fc@frenchoptions@mil@plural\space
1414 \or % 2 => d million(s) (de milliards)*
1415   \def\@temph{2}%
1416   \count10=\csname fc@frenchoptions@n-illion@plural\endcsname\space
1417 \or % 3 => d milliard(s) (de milliards)*
1418   \def\@temph{2}%
1419   \count10=\csname fc@frenchoptions@n-illiard@plural\endcsname\space
1420 \or % 4 => d mille milliards (de milliards)*
1421   \def\@temph{1}%
1422   \count10=\fc@frenchoptions@mil@plural\space
1423 \else % 5 => d million(s) (de milliards)*
1424   \def\@temph{2}%
1425   \count10=\csname fc@frenchoptions@n-illion@plural\endcsname\space
1426 \fi
1427 \let\@tempg\@tempe
1428 \edef\@tempf{%
1429   \ifcase\count11 % 0 .. 5
1430   \or
1431     mil\fc@@do@plural@mark \fc@frenchoptions@mil@plural@mark
1432   \or
1433     million\fc@@do@plural@mark s%
1434   \or
1435     milliard\fc@@do@plural@mark s%
1436   \or
1437     mil\fc@@do@plural@mark\fc@frenchoptions@mil@plural@mark
1438     \noexpand\@nil\fc@frenchoptions@supermillion@dos
1439     \noexpand\fc@wcase milliards% 4
1440   \or
1441     million\fc@@do@plural@mark s%
1442     \noexpand\@nil\fc@frenchoptions@supermillion@dos
1443     de\fc@frenchoptions@supermillion@dos\noexpand\fc@wcase  milliards% 5
1444   \fi
1445 }%
1446 \edef\@tempe{%
1447   \ifx\@tempf\@empty\else
1448     \expandafter\fc@wcase\@tempf\@nil
1449   \fi
1450 \@tempg
1451 }%
1452 \else
1453   \def\@temph{0}%
1454   \let\@tempe\@empty
1455 \fi

```

Now place into `cs@tempa` the assignment of results `\@tempa` and `\@tempb` to #2 and #3 for further propagation after closing brace.

```

1456   \expandafter\toks\expandafter1\expandafter{\@tempb}%
1457   \toks0{#2}%
1458   \edef\@tempa{\the\toks0 \@tempa \def\noexpand#3{\the\toks1}}%
1459   \expandafter
1460   }\@tempa
1461 }%
1462 \global\let\fc@pot@recursivefrench\fc@pot@recursivefrench

```

`fc@muladdfrench` Macro `\fc@muladdfrench` is used to format the sum of a number  $a$  and the product of a number  $d$  by a power of ten  $10^w$ . Number  $d$  is made of three consecutive digits  $d_{w+2}d_{w+1}d_w$  of respective weights  $w+2$ ,  $w+1$ , and  $w$ , while number  $a$  is made of all digits with weight  $w' > w+2$  that have already been formatted. First check that the macro is not yet defined.

```

1463 \ifcsundef{fc@muladdfrench}{}%
1464 \PackageError{fmtcount}{Duplicate definition}{Redefinition of macro
1465   'fc@muladdfrench'}}

```

Arguments as follows:

#2 input, plural indicator for number  $d$

#3 input, formatted number  $d$

#5 input, formatted number  $10^w$ , i.e. power of ten which is multiplied by  $d$

Implicit arguments from context:

`\@tempa` input, formatted number  $a$

output, macro to which place the mul-add result

`\count8` input, power type indicator for  $10^{w'}$ , where  $w'$  is a weight of  $a$ , this is an index in  $[0..2]$  that reflects whether  $10^{w'}$  is formatted by “mil(le)” — for index = 1 — or by “ $\langle n \rangle$ illion(s)| $\langle n \rangle$ illiard(s)” — for index = 2

`\count9` input, power type indicator for  $10^w$ , this is an index in  $[0..2]$  that reflect whether the weight  $w$  of  $d$  is formatted by “metanothing” — for index = 0, “mil(le)” — for index = 1 — or by “ $\langle n \rangle$ illion(s)| $\langle n \rangle$ illiard(s)” — for index = 2

```

1466 \def\fc@muladdfrench#1#2#3{%
1467   {%

```

First we save input arguments #1 – #3 to local macros `\@tempc`, `\@tempd` and `\@tempf`.

```

1468   \edef\@tempc{#1}%
1469   \edef\@tempd{#2}%
1470   \edef\@tempf{#3}%
1471   \let\@tempc\@tempc
1472   \let\@tempd\@tempd

```

First we want to do the “multiplication” of  $d \Rightarrow \@tempd$  and of  $10^w \Rightarrow \@tempf$ . So, prior to this we do some preprocessing of  $d \Rightarrow \@tempd$ : we force `\@tempd` to `\empty` if both  $d = 1$  and  $10^w \Rightarrow$  “mil(le)”, this is because we, French, we do not say “un mil”, but just “mil”.

```

1473   \ifnum\@tempc=1 %
1474     \ifnum\count9=1 %
1475       \let\@tempd\empty
1476     \fi
1477   \fi

```

Now we do the “multiplication” of  $d = \text{\@tempd}$  and of  $10^w = \text{\@tempf}$ , and place the result into  $\text{\@tempg}$ .

```

1478 \edef\@tempg{%
1479   \@tempd
1480   \ifx\@tempd\@empty\else
1481     \ifx\@tempf\@empty\else
1482       \ifcase\count9 %
1483         \or
1484         \fc@frenchoptions@submillion@dos
1485       \or
1486         \fc@frenchoptions@supermillion@dos
1487     \fi
1488   \fi
1489 \@tempf
1491 }%
```

Now to the “addition” of  $a \Rightarrow \text{\@tempa}$  and  $d \times 10^w \Rightarrow \text{\@tempg}$ , and place the results into  $\text{\@temph}$ .

```

1492 \edef\@temph{%
1493   \@tempa
1494   \ifx\@tempa\@empty\else
1495     \ifx\@tempg\@empty\else
1496       \ifcase\count8 %
1497         \or
1498         \fc@frenchoptions@submillion@dos
1499       \or
1500         \fc@frenchoptions@supermillion@dos
1501     \fi
1502   \fi
1503 \@tempg
1504 }%
```

Now propagate the result — i.e. the expansion of  $\text{\@temph}$  — into macro  $\text{\@tempa}$  after closing brace.

```

1506 \def\@tempb##1{\def\@tempa{\def\@tempa{##1}}}%
1507 \expandafter\@tempb\expandafter{\@temph}%
1508 \expandafter
1509 }\@tempa
1510 }%
```

```

1511 \global\let\fc@muladdfrench\fc@muladdfrench
```

Macro  $\text{\fc@lthundredstringfrench}$  is used to format a number in interval  $[0..99]$ . First we check that it is not already defined.

```

1512 \ifcsundef\fc@lthundredstringfrench\{}{%
1513   \PackageError{fmtcount}{Duplicate definition}{Redefinition of macro
1514     ‘\fc@lthundredstringfrench’}}
```

The number to format is not passed as an argument to this macro, instead each digit of it is in a  $\text{\fc@digit@<w>}$  macro after this number has been parsed. So the only thing that

`\fc@lthundredstringfrench` needs to know  $\langle w \rangle$  which is passed as `\count0` for the less significant digit.

#1 input/output macro to which append the result

Implicit input arguments as follows:

`\count0` weight  $w$  of least significant digit  $d_w$ .

The formatted number is appended to the content of #1, and the result is placed into #1.

```
1515 \def\fc@lthundredstringfrench#1{%
```

```
1516  {%
```

First save arguments into local temporary macro.

```
1517  \let\@tempc#1%
```

Read units  $d_w$  to `\count1`.

```
1518  \fc@read@unit{\count1}{\count0}%
```

Read tens  $d_{w+1}$  to `\count2`.

```
1519  \count3\count0 %
```

```
1520  \advance\count3 1 %
```

```
1521  \fc@read@unit{\count2}{\count3}%
```

Now do the real job, set macro `\@tempa` to #1 followed by  $d_{w+1}d_w$  formatted.

```
1522  \edef\@tempa{%
```

```
1523    \@tempc
```

```
1524    \ifnum\count2>1 %
```

```
1525      % 20 .. 99
```

```
1526      \ifnum\count2>6 %
```

```
1527        % 70 .. 99
```

```
1528        \ifnum\count2<8 %
```

```
1529          % 70 .. 79
```

```
1530          \@seventies{\count1}%
```

```
1531        \else
```

```
1532          % 80..99
```

```
1533          \ifnum\count2<9 %
```

```
1534            % 80 .. 89
```

```
1535            \@eighties{\count1}%
```

```
1536          \else
```

```
1537            % 90 .. 99
```

```
1538            \@nineties{\count1}%
```

```
1539          \fi
```

```
1540        \fi
```

```
1541      \else
```

```
1542        % 20..69
```

```
1543        \@tenstring{\count2}%
```

```
1544        \ifnum\count1>0 %
```

```
1545          % x1 .. x0
```

```
1546          \ifnum\count1=1 %
```

```
1547            % x1
```

```
1548            \fc@frenchoptions@submillion@dos\@andname\fc@frenchoptions@submillion@dos
```

```
1549          \else
```

```
1550            % x2 .. x9
```

```

1551         -%
1552         \fi
1553         \@unitstring{\count1}%
1554     \fi
1555     \fi
1556 \else
1557     % 0 .. 19
1558     \ifnum\count2=0 % when tens = 0
1559     % 0 .. 9
1560     \ifnum\count1=0 % when units = 0
1561     % \count3=1 when #1 = 0, i.e. only for the unit of the top level number
1562     \ifnum\count3=1 %
1563     \ifnum\fc@max@weight=0 %
1564     \@unitstring{0}%
1565     \fi
1566     \fi
1567     \else
1568     % 1 .. 9
1569     \@unitstring{\count1}%
1570     \fi
1571     \else
1572     % 10 .. 19
1573     \@teenstring{\count1}%
1574     \fi
1575 \fi
1576 }%

```

Now propagate the expansion of \@tempa into #1 after closing brace.

```

1577 \def\@tempb##1{\def\@tempa{\def#1{##1}}}%
1578 \expandafter\@tempb\expandafter{\@tempa}%
1579 \expandafter
1580 }\@tempa
1581 }%
1582 \global\let\fc@lthundredstringfrench\fc@lthundredstringfrench

```

~~Macro~~ \fc@ltthousandstringfrench is used to format a number in interval [0..999]. First we check that it is not already defined.

```

1583 \ifcsundef\fc@ltthousandstringfrench\{}{%
1584 \PackageError{fmtcount}{Duplicate definition}{Redefinition of macro
1585 'fc@ltthousandstringfrench'}}

```

Output is empty for 0. Arguments as follows:

#2 output, macro, formatted number  $d = d_{w+2}d_{w+1}d_w$

Implicit input arguments as follows:

\count0 input weight  $10^w$  of number  $d_{w+2}d_{w+1}d_w$  to be formatted.

\count5 least weight of formatted number with a non null digit.

\count9 input, power type indicator of  $10^w$   $0 \Rightarrow \emptyset$ ,  $1 \Rightarrow$  “mil(le)”,  $2 \Rightarrow$   
 $\langle n \rangle$ illion(s)| $\langle n \rangle$ illiard(s)

```

1586 \def\fc@ltthousandstringfrench#1{%
1587 {%

```

Set counter \count2 to digit  $d_{w+2}$ , i.e. hundreds.

```
1588 \count4\count0 %
1589 \advance\count4 by 2 %
1590 \fc@read@unit{\count2 }{\count4 }%
```

Check that the two subsequent digits  $d_{w+1}d_w$  are non zero, place check-result into \@tempa.

```
1591 \advance\count4 by -1 %
1592 \count3\count4 %
1593 \advance\count3 by -1 %
1594 \fc@check@nonzeros{\count3 }{\count4 }\@tempa
```

Compute plural mark of 'cent' into \@temps.

```
1595 \edef\@temps{%
1596 \ifcase\fc@frenchoptions@cent@plural\space
1597 % 0 => always
1598 s%
1599 \or
1600 % 1 => never
1601 \or
1602 % 2 => multiple
1603 \ifnum\count2>1s\fi
1604 \or
1605 % 3 => multiple g-last
1606 \ifnum\count2>1 \ifnum\@tempa=0 \ifnum\count0=\count6s\fi\fi\fi
1607 \or
1608 % 4 => multiple l-last
1609 \ifnum\count2>1 \ifnum\@tempa=0 \ifnum\count9=0s\else\ifnum\count9=2s\fi\fi\fi\fi
1610 \fi
1611 }%
1612 % compute spacing after cent(s?) into \@tempb
1613 \expandafter\let\expandafter\@tempb
1614 \ifnum\@tempa>0 \fc@frenchoptions@submillion@dos\else\@empty\fi
1615 % now place into \@tempa the hundreds
1616 \edef\@tempa{%
1617 \ifnum\count2=0 %
1618 \else
1619 \ifnum\count2=1 %
1620 \expandafter\fc@wcase\@hundred\@nil
1621 \else
1622 \@unitstring{\count2}\fc@frenchoptions@submillion@dos
1623 \noexpand\fc@wcase\@hundred\@temps\noexpand\@nil
1624 \fi
1625 \@tempb
1626 \fi
1627 }%
1628 % now append to \@tempa the ten and unit
1629 \fc@lthundredstringfrench\@tempa
```

Propagate expansion of \@tempa into macro #1 after closing brace.

```
1630 \def\@tempb##1{\def\@tempa{\def#1{##1}}}%
```

```

1631 \expandafter\@tempb\expandafter{\@tempa}%
1632 \expandafter
1633 }\@tempa
1634 }%
1635 \global\let\fc@ltthousandstringfrench\fc@ltthousandstringfrench

```

numberstringfrenchMacro @@numberstringfrench is the main engine for formatting cadinal numbers in French. First we check that the control sequence is not yet defined.

```

1636 \ifcsundef{@@numberstringfrench}{}%
1637 \PackageError{fmtcount}{Duplicate definition}{Redefinition of macro '@@numberstringfrench'}}

```

Arguments are as follows:

- #1 number to convert to string
- #2 macro into which to place the result

```

1638 \def\@@numberstringfrench#1#2{%
1639  {%

```

First parse input number to be formatted and do some error handling.

```

1640 \edef\@tempa{#1}%
1641 \expandafter\fc@number@parser\expandafter{\@tempa}%
1642 \ifnum\fc@min@weight<0 %
1643 \PackageError{fmtcount}{Out of range}%
1644 {This macro does not work with fractional numbers}%
1645 \fi

```

In the sequel, \@tempa is used to accumulate the formatted number. Please note that \space after \fc@sign@case is eaten by preceding number collection. This \space is needed so that when \fc@sign@case expands to '0', then \@tempa is defined to '' (i.e. empty) rather than to '\relax'.

```

1646 \edef\@tempa{\ifcase\fc@sign@case\space\or\fc@wcase plus\@nil\or\fc@wcase moins\@nil\fi}%
1647 \fc@nbrstr@preamble
1648 \fc@@nbrstrfrench@inner
1649 \fc@nbrstr@postamble

```

Propagate the result — i.e. expansion of \@tempa — into macro #2 after closing brace.

```

1650 \def\@tempb##1{\def\@tempa{\def#2{##1}}}%
1651 \expandafter\@tempb\expandafter{\@tempa}%
1652 \expandafter
1653 }\@tempa
1654 }%
1655 \global\let\@@numberstringfrench\@@numberstringfrench

```

@@nbrstrfrench@innerCommon part of @@numberstringfrench and @@ordinalstringfrench. Arguments are as follows:

\@tempa input/output, macro to which the result is to be aggregated, initially empty or contains the sign indication.

```

1656 \def\fc@@nbrstrfrench@inner{%

```

Now loop, first we compute starting weight as  $3 \times \left\lfloor \frac{\fc@max@weight}{3} \right\rfloor$  into \count0.

```

1657 \count0=\fc@max@weight
1658 \divide\count0 by 3 %
1659 \multiply\count0 by 3 %

```

Now we compute final weight into `\count5`, and round down to multiple of 3 into `\count6`.  
Warning: `\count6` is an implicit input argument to macro `\fc@ltthousandstringfrench`.

```
1660 \fc@intpart@find@last{\count5 }%
1661 \count6\count5 %
1662 \divide\count6 3 %
1663 \multiply\count6 3 %
1664 \count8=0 %
1665 \loop
```

First we check whether digits in weight interval  $[w..(w+2)]$  are all zero and place check result into macro `\@tempt`.

```
1666 \count1\count0 %
1667 \advance\count1 by 2 %
1668 \fc@check@nonzeros{\count0 }{\count1 }\@tempt
```

Now we generate the power of ten  $10^w$ , formatted power of ten goes to `\@tempb`, while power type indicator goes to `\count9`.

```
1669 \fc@powerof ten \@tempt{\count9 }\@tempb
```

Now we generate the formatted number  $d$  into macro `\@tempd` by which we need to multiply  $10^w$ . Implicit input argument is `\count9` for power type of  $10^9$ , and `\count6`

```
1670 \fc@ltthousandstringfrench \@tempd
```

Finally do the multiplication-addition. Implicit arguments are `\@tempa` for input/output growing formatted number, `\count8` for input previous power type, i.e. power type of  $10^{w+3}$ , `\count9` for input current power type, i.e. power type of  $10^w$ .

```
1671 \fc@muladdfrench \@tempt \@tempd \@tempb
```

Then iterate.

```
1672 \count8\count9 %
1673 \advance\count0 by -3 %
1674 \ifnum\count6>\count0 \else
1675 \repeat
1676 }%
1677 \global\let\fc@nbrstrfrench@inner\fc@nbrstrfrench@inner
```

`\fc@ordinalstringfrench` Macro `\@@ordinalstringfrench` is the main engine for formatting ordinal numbers in French. First check it is not yet defined.

```
1678 \ifcsundef{\@@ordinalstringfrench}{\}%
1679 \PackageError{fmtcount}{Duplicate definition}{Redefinition of macro
1680 '@@ordinalstringfrench'}}
```

Arguments are as follows:

- #1 number to convert to string
- #2 macro into which to place the result

```
1681 \def\@@ordinalstringfrench#1#2{%
1682 {%
```

First parse input number to be formatted and do some error handling.

```
1683 \edef\@tempa{#1}%
1684 \expandafter\fc@number@parser\expandafter{\@tempa}%
```

```

1685 \ifnum\fc@min@weight<0 %
1686   \PackageError{fmtcount}{Out of range}%
1687   {This macro does not work with fractional numbers}%
1688 \fi
1689 \ifnum\fc@sign@case>0 %
1690   \PackageError{fmtcount}{Out of range}%
1691   {This macro does with negative or explicitly marked as positive numbers}%
1692 \fi

```

Now handle the special case of first. We set `\count0` to 1 if we are in this case, and to 0 otherwise

```

1693 \ifnum\fc@max@weight=0 %
1694   \ifnum\csname fc@digit@0\endcsname=1 %
1695     \count0=1 %
1696   \else
1697     \count0=0 %
1698   \fi
1699 \else
1700   \count0=0 %
1701 \fi
1702 \ifnum\count0=1 %
1703   \protected@edef\@tempa{\expandafter\fc@wcase\fc@first\@nil}%
1704 \else

```

Now we tamper a little bit with the plural handling options to ensure that there is no final plural mark.

```

1705 \def\@tempa##1{%
1706   \expandafter\edef\csname fc@frenchoptions@##1@plural\endcsname{%
1707     \ifcase\csname fc@frenchoptions@##1@plural\endcsname\space
1708     0% 0: always => always
1709     \or
1710     1% 1: never => never
1711     \or
1712     6% 2: multiple => multiple ng-last
1713     \or
1714     1% 3: multiple g-last => never
1715     \or
1716     5% 4: multiple l-last => multiple lng-last
1717     \or
1718     5% 5: multiple lng-last => multiple lng-last
1719     \or
1720     6% 6: multiple ng-last => multiple ng-last
1721     \fi
1722   }%
1723 }%
1724 \@tempa{vingt}%
1725 \@tempa{cent}%
1726 \@tempa{mil}%
1727 \@tempa{n-illion}%

```

1728 \@tempa{n-illiard}%

Now make \fc@wcase and \@nil non expandable

1729 \let\fc@wcase@save\fc@wcase

1730 \def\fc@wcase{\noexpand\fc@wcase}%

1731 \def\@nil{\noexpand\@nil}%

In the sequel, \@tempa is used to accumulate the formatted number.

1732 \let\@tempa\@empty

1733 \fc@@nbrstrfrench@inner

Now restore \fc@wcase

1734 \let\fc@wcase\fc@wcase@save

Now we add the “ième” ending

1735 \expandafter\fc@get@last@word\expandafter{\@tempa}\@tempb\@tempc

1736 \expandafter\fc@get@last@letter\expandafter{\@tempc}\@tempd\@tempe

1737 \def\@tempf{e}%

1738 \ifx\@tempe\@tempf

1739 \protected@edef\@tempa{\@tempb\expandafter\fc@wcase\@tempd i\protect\‘eme\@nil}%

1740 \else

1741 \def\@tempf{q}%

1742 \ifx\@tempe\@tempf

1743 \protected@edef\@tempa{\@tempb\expandafter\fc@wcase\@tempd qui\protect\‘eme\@nil}%

1744 \else

1745 \def\@tempf{f}%

1746 \ifx\@tempe\@tempf

1747 \protected@edef\@tempa{\@tempb\expandafter\fc@wcase\@tempd vi\protect\‘eme\@nil}%

1748 \else

1749 \protected@edef\@tempa{\@tempb\expandafter\fc@wcase\@tempc i\protect\‘eme\@nil}%

1750 \fi

1751 \fi

1752 \fi

1753 \fi

Apply \fc@gc case to the result.

1754 \fc@apply@gc case

Propagate the result — i.e. expansion of \@tempa — into macro #2 after closing brace.

1755 \def\@tempb##1{\def\@tempa{\def#2{##1}}}%

1756 \expandafter\@tempb\expandafter{\@tempa}%

1757 \expandafter

1758 }\@tempa

1759 }%

1760 \global\let\@@ordinalstringfrench\@ordinalstringfrench

Macro \fc@frenchoptions@setdefaults allows to set all options to default for the French.

1761 \newcommand\*\fc@frenchoptions@setdefaults{%

1762 \csname KV@fcfrench@all plural\endcsname{reformed}%

1763 \fc@gl@def\fc@frenchoptions@submillion@dos{-}%

1764 \fc@gl@let\fc@frenchoptions@supermillion@dos\space

1765 \fc@gl@let\fc@u@in@duo\@empty% Could be ‘u’

```

1766 % \fc@gl@let\fc@poweroften\fc@@pot@longscalefrench
1767 \fc@gl@let\fc@poweroften\fc@@pot@recursivefrench
1768 \fc@gl@def\fc@longscale@illiard@upto{0}% infinity
1769 \fc@gl@def\fc@frenchoptions@mil@plural@mark{le}%
1770 }%
1771 \global\let\fc@frenchoptions@setdefaults\fc@frenchoptions@setdefaults
1772 {%
1773 \let\fc@gl@def\gdef
1774 \def\fc@gl@let{\global\let}%
1775 \fc@frenchoptions@setdefaults
1776 }%

```

Make some indirection to call the current French dialect corresponding macro.

```

1777 \gdef\@ordinalstringMfrench{\csuse{@ordinalstringMfrench\fmtcount@french}}%
1778 \gdef\@ordinalstringFfrench{\csuse{@ordinalstringFfrench\fmtcount@french}}%
1779 \gdef\@OrdinalstringMfrench{\csuse{@OrdinalstringMfrench\fmtcount@french}}%
1780 \gdef\@OrdinalstringFfrench{\csuse{@OrdinalstringFfrench\fmtcount@french}}%
1781 \gdef\@numberstringMfrench{\csuse{@numberstringMfrench\fmtcount@french}}%
1782 \gdef\@numberstringFfrench{\csuse{@numberstringFfrench\fmtcount@french}}%
1783 \gdef\@NumberstringMfrench{\csuse{@NumberstringMfrench\fmtcount@french}}%
1784 \gdef\@NumberstringFfrench{\csuse{@NumberstringFfrench\fmtcount@french}}%

```

### 9.1.7 fc-frenchb.def

```

1785 \ProvidesFCLanguage{frenchb}[2013/08/17]%
1786 \FCloadlang{french}%

```

Set frenchb to be equivalent to french.

```

1787 \global\let\@ordinalMfrenchb=\@ordinalMfrench
1788 \global\let\@ordinalFfrenchb=\@ordinalFfrench
1789 \global\let\@ordinalNfrenchb=\@ordinalNfrench
1790 \global\let\@numberstringMfrenchb=\@numberstringMfrench
1791 \global\let\@numberstringFfrenchb=\@numberstringFfrench
1792 \global\let\@numberstringNfrenchb=\@numberstringNfrench
1793 \global\let\@NumberstringMfrenchb=\@NumberstringMfrench
1794 \global\let\@NumberstringFfrenchb=\@NumberstringFfrench
1795 \global\let\@NumberstringNfrenchb=\@NumberstringNfrench
1796 \global\let\@ordinalstringMfrenchb=\@ordinalstringMfrench
1797 \global\let\@ordinalstringFfrenchb=\@ordinalstringFfrench
1798 \global\let\@ordinalstringNfrenchb=\@ordinalstringNfrench
1799 \global\let\@OrdinalstringMfrenchb=\@OrdinalstringMfrench
1800 \global\let\@OrdinalstringFfrenchb=\@OrdinalstringFfrench
1801 \global\let\@OrdinalstringNfrenchb=\@OrdinalstringNfrench

```

### 9.1.8 fc-german.def

German definitions (thank you to K. H. Fricke for supplying this information)

```

1802 \ProvidesFCLanguage{german}[2016/01/12]%

```

Define macro that converts a number or count register (first argument) to an ordinal, and stores the result in the second argument, which must be a control sequence. Masculine:

```

1803 \newcommand{\@ordinalMgerman}[2]{%
1804   \edef#2{\number#1\relax.}%
1805 }%
1806 \global\let\@ordinalMgerman\@ordinalMgerman

```

Feminine:

```

1807 \newcommand{\@ordinalFgerman}[2]{%
1808   \edef#2{\number#1\relax.}%
1809 }%
1810 \global\let\@ordinalFgerman\@ordinalFgerman

```

Neuter:

```

1811 \newcommand{\@ordinalNgerman}[2]{%
1812   \edef#2{\number#1\relax.}%
1813 }%
1814 \global\let\@ordinalNgerman\@ordinalNgerman

```

Convert a number to text. The easiest way to do this is to break it up into units, tens and teens.

Units (argument must be a number from 0 to 9, 1 on its own (eins) is dealt with separately):

```

1815 \newcommand*\@@unitstringgerman[1]{%
1816   \ifcase#1%
1817     null%
1818     \or ein%
1819     \or zwei%
1820     \or drei%
1821     \or vier%
1822     \or f"unf%
1823     \or sechs%
1824     \or sieben%
1825     \or acht%
1826     \or neun%
1827   \fi
1828 }%
1829 \global\let\@@unitstringgerman\@@unitstringgerman

```

Tens (argument must go from 1 to 10):

```

1830 \newcommand*\@@tenstringgerman[1]{%
1831   \ifcase#1%
1832     \or zehn%
1833     \or zwanzig%
1834     \or drei{\ss}ig%
1835     \or vierzig%
1836     \or f"unfzig%
1837     \or sechzig%
1838     \or siebzig%
1839     \or achtzig%
1840     \or neunzig%
1841     \or einhundert%
1842   \fi
1843 }%
1844 \global\let\@@tenstringgerman\@@tenstringgerman

```

`\einhundert` is set to `einhundert` by default, user can redefine this command to just `hundert` if required, similarly for `\eintausend`.

```
1845 \providecommand*\einhundert{einhundert}%
1846 \providecommand*\eintausend{eintausend}%
1847 \global\let\einhundert\einhundert
1848 \global\let\eintausend\eintausend
```

Teens:

```
1849 \newcommand*\@@teenstringgerman[1]{%
1850   \ifcase#1%
1851     zehn%
1852     \or elf%
1853     \or zw"olf%
1854     \or dreizehn%
1855     \or vierzehn%
1856     \or f"unfzehn%
1857     \or sechzehn%
1858     \or siebzehn%
1859     \or achtzehn%
1860     \or neunzehn%
1861   \fi
1862 }%
1863 \global\let\@@teenstringgerman\@@teenstringgerman
```

The results are stored in the second argument, but doesn't display anything.

```
1864 \newcommand*\@numberstringMgerman[2]{%
1865   \let\@unitstring=\@unitstringgerman
1866   \let\@teenstring=\@teenstringgerman
1867   \let\@tenstring=\@tenstringgerman
1868   \@numberstringgerman{#1}{#2}%
1869 }%
1870 \global\let\@numberstringMgerman\@numberstringMgerman
```

Feminine and neuter forms:

```
1871 \global\let\@numberstringFgerman=\@numberstringMgerman
1872 \global\let\@numberstringNgerman=\@numberstringMgerman
```

As above, but initial letters in upper case:

```
1873 \newcommand*\@NumberstringMgerman[2]{%
1874   \@numberstringMgerman{#1}{\@num@str}%
1875   \edef#2{\noexpand\MakeUppercase\expandonce\@num@str}%
1876 }%
1877 \global\let\@NumberstringMgerman\@NumberstringMgerman
```

Feminine and neuter form:

```
1878 \global\let\@NumberstringFgerman=\@NumberstringMgerman
1879 \global\let\@NumberstringNgerman=\@NumberstringMgerman
```

As above, but for ordinals.

```
1880 \newcommand*\@ordinalstringMgerman[2]{%
1881   \let\@unitthstring=\@unitthstringgerman
```

```

1882 \let\@teenthstring=\@teenthstringMgerman
1883 \let\@tenthstring=\@tenthstringMgerman
1884 \let\@unitstring=\@unitstringgerman
1885 \let\@teenstring=\@teenstringgerman
1886 \let\@tenstring=\@tenstringgerman
1887 \def\@thousandth{tausendster}%
1888 \def\@hundredth{hundertster}%
1889 \@ordinalstringgerman{#1}{#2}%
1890 }%
1891 \global\let\@ordinalstringMgerman\@ordinalstringMgerman

```

Feminine form:

```

1892 \newcommand*\@ordinalstringFgerman}[2]{%
1893 \let\@unitthstring=\@unitthstringFgerman
1894 \let\@teenthstring=\@teenthstringFgerman
1895 \let\@tenthstring=\@tenthstringFgerman
1896 \let\@unitstring=\@unitstringgerman
1897 \let\@teenstring=\@teenstringgerman
1898 \let\@tenstring=\@tenstringgerman
1899 \def\@thousandth{tausendste}%
1900 \def\@hundredth{hundertste}%
1901 \@ordinalstringgerman{#1}{#2}%
1902 }%
1903 \global\let\@ordinalstringFgerman\@ordinalstringFgerman

```

Neuter form:

```

1904 \newcommand*\@ordinalstringNgerman}[2]{%
1905 \let\@unitthstring=\@unitthstringNgerman
1906 \let\@teenthstring=\@teenthstringNgerman
1907 \let\@tenthstring=\@tenthstringNgerman
1908 \let\@unitstring=\@unitstringgerman
1909 \let\@teenstring=\@teenstringgerman
1910 \let\@tenstring=\@tenstringgerman
1911 \def\@thousandth{tausendstes}%
1912 \def\@hundredth{hunderstes}%
1913 \@ordinalstringgerman{#1}{#2}%
1914 }%
1915 \global\let\@ordinalstringNgerman\@ordinalstringNgerman

```

As above, but with initial letters in upper case.

```

1916 \newcommand*\@OrdinalstringMgerman}[2]{%
1917 \@ordinalstringMgerman{#1}{\@num@str}%
1918 \edef#2{\noexpand\MakeUppercase\expandonce\@num@str}%
1919 }%
1920 \global\let\@OrdinalstringMgerman\@OrdinalstringMgerman

```

Feminine form:

```

1921 \newcommand*\@OrdinalstringFgerman}[2]{%
1922 \@ordinalstringFgerman{#1}{\@num@str}%
1923 \edef#2{\noexpand\MakeUppercase\expandonce\@num@str}%
1924 }%

```

```
1925 \global\let\@OrdinalstringFgerman\@OrdinalstringFgerman
```

Neuter form:

```
1926 \newcommand*\@OrdinalstringNgerman}[2]{%
1927 \@OrdinalstringNgerman{#1}{\@num@str}%
1928 \edef#2{\noexpand\MakeUppercase\expandonce\@num@str}%
1929 }%
1930 \global\let\@OrdinalstringNgerman\@OrdinalstringNgerman
```

Code for converting numbers into textual ordinals. As before, it is easier to split it into units, tens and teens. Units:

```
1931 \newcommand*\@unitthstringMgerman[1]{%
1932 \ifcase#1%
1933 nullter%
1934 \or erster%
1935 \or zweiter%
1936 \or dritter%
1937 \or vierter%
1938 \or f"unfter%
1939 \or sechster%
1940 \or siebter%
1941 \or achter%
1942 \or neunter%
1943 \fi
1944 }%
1945 \global\let\@unitthstringMgerman\@unitthstringMgerman
```

Tens:

```
1946 \newcommand*\@tenthstringMgerman[1]{%
1947 \ifcase#1%
1948 \or zehnter%
1949 \or zwanzigster%
1950 \or drei{\ss}igster%
1951 \or vierzigster%
1952 \or f"unzigster%
1953 \or sechzigster%
1954 \or siebzigster%
1955 \or achtzigster%
1956 \or neunzigster%
1957 \fi
1958 }%
1959 \global\let\@tenthstringMgerman\@tenthstringMgerman
```

Teens:

```
1960 \newcommand*\@teenthstringMgerman[1]{%
1961 \ifcase#1%
1962 zehnter%
1963 \or elfter%
1964 \or zw"olfter%
1965 \or dreizehnter%
1966 \or vierzehnter%
```

1967 \or f"unfzehnter%  
 1968 \or sechzehnter%  
 1969 \or siebzehnter%  
 1970 \or achtzehnter%  
 1971 \or neunzehnter%  
 1972 \fi  
 1973 }%  
 1974 \global\let\@@teenthstringMgerman\@@teenthstringMgerman

Units (feminine):

1975 \newcommand\*\@@unitthstringFgerman[1]{%  
 1976 \ifcase#1%  
 1977 nullte%  
 1978 \or erste%  
 1979 \or zweite%  
 1980 \or dritte%  
 1981 \or vierte%  
 1982 \or f"unfte%  
 1983 \or sechste%  
 1984 \or siebte%  
 1985 \or achte%  
 1986 \or neunte%  
 1987 \fi  
 1988 }%  
 1989 \global\let\@@unitthstringFgerman\@@unitthstringFgerman

Tens (feminine):

1990 \newcommand\*\@@tenthstringFgerman[1]{%  
 1991 \ifcase#1%  
 1992 \or zehnte%  
 1993 \or zwanzigste%  
 1994 \or drei{\ss}igste%  
 1995 \or vierzigste%  
 1996 \or f"unfzigste%  
 1997 \or sechzigste%  
 1998 \or siebzigste%  
 1999 \or achtzigste%  
 2000 \or neunzigste%  
 2001 \fi  
 2002 }%  
 2003 \global\let\@@tenthstringFgerman\@@tenthstringFgerman

Teens (feminine)

2004 \newcommand\*\@@teenthstringFgerman[1]{%  
 2005 \ifcase#1%  
 2006 zehnte%  
 2007 \or elfte%  
 2008 \or zw"olfte%  
 2009 \or dreizehnte%  
 2010 \or vierzehnte%  
 2011 \or f"unfzehnte%

2012 \or sechzehnte%  
 2013 \or siebzehnte%  
 2014 \or achtzehnte%  
 2015 \or neunzehnte%  
 2016 \fi  
 2017 }%  
 2018 \global\let\@@teenthstringFgerman\@@teenthstringFgerman

Units (neuter):

2019 \newcommand\*\@@unitthstringNgerman[1]{%  
 2020 \ifcase#1%  
 2021 nulltes%  
 2022 \or erstes%  
 2023 \or zweites%  
 2024 \or drittes%  
 2025 \or viertes%  
 2026 \or f"unftes%  
 2027 \or sechstes%  
 2028 \or siebtes%  
 2029 \or achttes%  
 2030 \or neuntes%  
 2031 \fi  
 2032 }%  
 2033 \global\let\@@unitthstringNgerman\@@unitthstringNgerman

Tens (neuter):

2034 \newcommand\*\@@tenthstringNgerman[1]{%  
 2035 \ifcase#1%  
 2036 \or zehntes%  
 2037 \or zwanzigstes%  
 2038 \or drei{\ss}igstes%  
 2039 \or vierzigstes%  
 2040 \or f"unfzigstes%  
 2041 \or sechzigstes%  
 2042 \or siebzigstes%  
 2043 \or achtzigstes%  
 2044 \or neunzigstes%  
 2045 \fi  
 2046 }%  
 2047 \global\let\@@tenthstringNgerman\@@tenthstringNgerman

Teens (neuter)

2048 \newcommand\*\@@teenthstringNgerman[1]{%  
 2049 \ifcase#1%  
 2050 zehntes%  
 2051 \or elftes%  
 2052 \or zw"olftes%  
 2053 \or dreizehntes%  
 2054 \or vierzehntes%  
 2055 \or f"unfzehntes%  
 2056 \or sechzehntes%

```

2057 \or siebzehntes%
2058 \or achtzehntes%
2059 \or neunzehntes%
2060 \fi
2061 }%
2062 \global\let\@@teenthstringNgerman\@@teenthstringNgerman

This appends the results to \#2 for number \#2 (in range 0 to 100.) null and eins are dealt
with separately in \@@numberstringgerman.

2063 \newcommand*\@@numberunderhundredgerman[2]{%
2064 \ifnum#1<10\relax
2065 \ifnum#1>0\relax
2066 \eappto#2{\@unitstring{#1}}%
2067 \fi
2068 \else
2069 \@tmpstrctr=#1\relax
2070 \@FCmodulo{\@tmpstrctr}{10}%
2071 \ifnum#1<20\relax
2072 \eappto#2{\@teenstring{\@tmpstrctr}}%
2073 \else
2074 \ifnum\@tmpstrctr=0\relax
2075 \else
2076 \eappto#2{\@unitstring{\@tmpstrctr}und}%
2077 \fi
2078 \@tmpstrctr=#1\relax
2079 \divide\@tmpstrctr by 10\relax
2080 \eappto#2{\@tenstring{\@tmpstrctr}}%
2081 \fi
2082 \fi
2083 }%
2084 \global\let\@@numberunderhundredgerman\@@numberunderhundredgerman

This stores the results in the second argument (which must be a control sequence), but it
doesn't display anything.

2085 \newcommand*\@@numberstringgerman[2]{%
2086 \ifnum#1>99999\relax
2087 \PackageError{fmtcount}{Out of range}%
2088 {This macro only works for values less than 100000}%
2089 \else
2090 \ifnum#1<0\relax
2091 \PackageError{fmtcount}{Negative numbers not permitted}%
2092 {This macro does not work for negative numbers, however
2093 you can try typing "minus" first, and then pass the modulus of
2094 this number}%
2095 \fi
2096 \fi
2097 \def#2{}%
2098 \@strctr=#1\relax \divide\@strctr by 1000\relax
2099 \ifnum\@strctr>1\relax

```

```

#1 is  $\geq 2000$ , \@strctr now contains the number of thousands
2100 \@numberunderhundredgerman{\@strctr}{#2}%
2101 \appto#2{tausend}%
2102 \else
#1 lies in range [1000,1999]
2103 \ifnum\@strctr=1\relax
2104 \eappto#2{\eintausend}%
2105 \fi
2106 \fi
2107 \@strctr=#1\relax
2108 \@FCmodulo{\@strctr}{1000}%
2109 \divide\@strctr by 100\relax
2110 \ifnum\@strctr>1\relax
now dealing with number in range [200,999]
2111 \eappto#2{\@unitstring{\@strctr}hundert}%
2112 \else
2113 \ifnum\@strctr=1\relax
dealing with number in range [100,199]
2114 \ifnum#1>1000\relax
if original number > 1000, use einhundert
2115 \appto#2{einhundert}%
2116 \else
otherwise use \einhundert
2117 \eappto#2{\einhundert}%
2118 \fi
2119 \fi
2120 \fi
2121 \@strctr=#1\relax
2122 \@FCmodulo{\@strctr}{100}%
2123 \ifnum#1=0\relax
2124 \def#2{null}%
2125 \else
2126 \ifnum\@strctr=1\relax
2127 \appto#2{eins}%
2128 \else
2129 \@numberunderhundredgerman{\@strctr}{#2}%
2130 \fi
2131 \fi
2132 }%
2133 \global\let\@numberstringgerman\@numberstringgerman

As above, but for ordinals
2134 \newcommand*\@numberunderhundredthgerman[2]{%
2135 \ifnum#1<10\relax
2136 \eappto#2{\@unitthstring{#1}}%
2137 \else
2138 \@tmpstrctr=#1\relax

```

```

2139 \@FCmodulo{\@tmpstrctr}{10}%
2140 \ifnum#1<20\relax
2141   \eappto#2{\@teenthstring{\@tmpstrctr}}%
2142 \else
2143   \ifnum\@tmpstrctr=0\relax
2144   \else
2145     \eappto#2{\@unitstring{\@tmpstrctr}und}%
2146   \fi
2147   \@tmpstrctr=#1\relax
2148   \divide\@tmpstrctr by 10\relax
2149   \eappto#2{\@tenthstring{\@tmpstrctr}}%
2150 \fi
2151 \fi
2152 }%
2153 \global\let\@@numberunderhundredthgerman\@@numberunderhundredthgerman

2154 \newcommand*\@@ordinalstringgerman[2]{%
2155 \ifnum#1>99999\relax
2156   \PackageError{fmtcount}{Out of range}%
2157   {This macro only works for values less than 100000}%
2158 \else
2159   \ifnum#1<0\relax
2160     \PackageError{fmtcount}{Negative numbers not permitted}%
2161     {This macro does not work for negative numbers, however
2162     you can try typing "minus" first, and then pass the modulus of
2163     this number}%
2164   \fi
2165 \fi
2166 \def#2{}%
2167 \@strctr=#1\relax \divide\@strctr by 1000\relax
2168 \ifnum\@strctr>1\relax

  #1 is  $\geq 2000$ , \@strctr now contains the number of thousands
2169 \@@numberunderhundredgerman{\@strctr}{#2}%

  is that it, or is there more?
2170 \@tmpstrctr=#1\relax \@FCmodulo{\@tmpstrctr}{1000}%
2171 \ifnum\@tmpstrctr=0\relax
2172   \eappto#2{\@thousandth}%
2173 \else
2174   \appto#2{tausend}%
2175 \fi
2176 \else

  #1 lies in range [1000,1999]
2177 \ifnum\@strctr=1\relax
2178   \ifnum#1=1000\relax
2179     \eappto#2{\@thousandth}%
2180   \else
2181     \eappto#2{eintausend}%
2182   \fi

```

```

2183 \fi
2184 \fi
2185 \@strctr=#1\relax
2186 \@FCmodulo{\@strctr}{1000}%
2187 \divide\@strctr by 100\relax
2188 \ifnum\@strctr>1\relax

    now dealing with number in range [200,999] is that it, or is there more?
2189 \@tmpstrctr=#1\relax \@FCmodulo{\@tmpstrctr}{100}%
2190 \ifnum\@tmpstrctr=0\relax
2191     \ifnum\@strctr=1\relax
2192         \eappto#2{\@hundredth}%
2193     \else
2194         \eappto#2{\@unitstring{\@strctr}\@hundredth}%
2195     \fi
2196 \else
2197     \eappto#2{\@unitstring{\@strctr}hundert}%
2198 \fi
2199 \else
2200     \ifnum\@strctr=1\relax

        dealing with number in range [100,199] is that it, or is there more?
2201     \@tmpstrctr=#1\relax \@FCmodulo{\@tmpstrctr}{100}%
2202     \ifnum\@tmpstrctr=0\relax
2203         \eappto#2{\@hundredth}%
2204     \else
2205         \ifnum#1>1000\relax
2206             \appto#2{einhundert}%
2207         \else
2208             \eappto#2{\einhundert}%
2209         \fi
2210     \fi
2211 \fi
2212 \fi
2213 \@strctr=#1\relax
2214 \@FCmodulo{\@strctr}{100}%
2215 \ifthenelse{\@strctr=0 \and #1>0}{\}{%
2216 \@numberunderhundredthgerman{\@strctr}{#2}%
2217 }%
2218 }%
2219 \global\let\@@ordinalstringgerman\@@ordinalstringgerman

    Load fc-germanb.def if not already loaded
2220 \FCloadlang{germanb}%

```

### 9.1.9 fc-germanb.def

```

2221 \ProvidesFCLanguage{germanb}[2013/08/17]%

    Load fc-german.def if not already loaded
2222 \FCloadlang{german}%

```

Set `germanb` to be equivalent to `german`.

```
2223 \global\let\@ordinalMgermanb=\@ordinalMgerman
2224 \global\let\@ordinalFgermanb=\@ordinalFgerman
2225 \global\let\@ordinalNgermanb=\@ordinalNgerman
2226 \global\let\@numberstringMgermanb=\@numberstringMgerman
2227 \global\let\@numberstringFgermanb=\@numberstringFgerman
2228 \global\let\@numberstringNgermanb=\@numberstringNgerman
2229 \global\let\@NumberstringMgermanb=\@NumberstringMgerman
2230 \global\let\@NumberstringFgermanb=\@NumberstringFgerman
2231 \global\let\@NumberstringNgermanb=\@NumberstringNgerman
2232 \global\let\@ordinalstringMgermanb=\@ordinalstringMgerman
2233 \global\let\@ordinalstringFgermanb=\@ordinalstringFgerman
2234 \global\let\@ordinalstringNgermanb=\@ordinalstringNgerman
2235 \global\let\@OrdinalstringMgermanb=\@OrdinalstringMgerman
2236 \global\let\@OrdinalstringFgermanb=\@OrdinalstringFgerman
2237 \global\let\@OrdinalstringNgermanb=\@OrdinalstringNgerman
```

### 9.1.10 `fc-italian`

Italian support is now handled by interfacing to Enrico Gregorio's `itnumpar` package.

```
2238 \ProvidesFCLanguage{italian}[2013/08/17]
2239
2240 \RequirePackage{itnumpar}
2241
2242 \newcommand{\@numberstringMitalian}[2]{%
2243   \edef#2{\noexpand\printnumeroinparole{#1}}%
2244 }
2245 \global\let\@numberstringMitalian\@numberstringMitalian
2246
2247 \newcommand{\@numberstringFitalian}[2]{%
2248   \edef#2{\noexpand\printnumeroinparole{#1}}}
2249
2250 \global\let\@numberstringFitalian\@numberstringFitalian
2251
2252 \newcommand{\@NumberstringMitalian}[2]{%
2253   \edef#2{\noexpand\printNumeroinparole{#1}}%
2254 }
2255 \global\let\@NumberstringMitalian\@NumberstringMitalian
2256
2257 \newcommand{\@NumberstringFitalian}[2]{%
2258   \edef#2{\noexpand\printNumeroinparole{#1}}%
2259 }
2260 \global\let\@NumberstringFitalian\@NumberstringFitalian
2261
2262 \newcommand{\@ordinalstringMitalian}[2]{%
2263   \edef#2{\noexpand\printordinalem{#1}}%
2264 }
2265 \global\let\@ordinalstringMitalian\@ordinalstringMitalian
2266
2267 \newcommand{\@ordinalstringFitalian}[2]{%
```

```

2268 \edef#2{\noexpand\printordinalef{#1}}%
2269 }
2270 \global\let\@OrdinalstringFitalian\@OrdinalstringFitalian
2271
2272 \newcommand{\@OrdinalstringMitalian}[2]{%
2273 \edef#2{\noexpand\printOrdinalem{#1}}%
2274 }
2275 \global\let\@OrdinalstringMitalian\@OrdinalstringMitalian
2276
2277 \newcommand{\@OrdinalstringFitalian}[2]{%
2278 \edef#2{\noexpand\printOrdinalef{#1}}%
2279 }
2280 \global\let\@OrdinalstringFitalian\@OrdinalstringFitalian
2281
2282 \newcommand{\@ordinalMitalian}[2]{%
2283 \edef#2{#1\relax\noexpand\fmtord{o}}%
2284
2285 \global\let\@ordinalMitalian\@ordinalMitalian
2286
2287 \newcommand{\@ordinalFitalian}[2]{%
2288 \edef#2{#1\relax\noexpand\fmtord{a}}%
2289 \global\let\@ordinalFitalian\@ordinalFitalian

```

### 9.1.11 fc-ngerman.def

```

2290 \ProvidesFCLanguage{ngerman}[2012/06/18]%
2291 \FCloadlang{german}%
2292 \FCloadlang{ngermanb}%

```

Set `ngerman` to be equivalent to `german`. Is it okay to do this? (I don't know the difference between the two.)

```

2293 \global\let\@ordinalMngerman=\@ordinalMgerman
2294 \global\let\@ordinalFngerman=\@ordinalFgerman
2295 \global\let\@ordinalNngerman=\@ordinalNgerman
2296 \global\let\@numberstringMngerman=\@numberstringMgerman
2297 \global\let\@numberstringFngerman=\@numberstringFgerman
2298 \global\let\@numberstringNngerman=\@numberstringNgerman
2299 \global\let\@NumberstringMngerman=\@NumberstringMgerman
2300 \global\let\@NumberstringFngerman=\@NumberstringFgerman
2301 \global\let\@NumberstringNngerman=\@NumberstringNgerman
2302 \global\let\@OrdinalstringMngerman=\@OrdinalstringMgerman
2303 \global\let\@OrdinalstringFngerman=\@OrdinalstringFgerman
2304 \global\let\@OrdinalstringNngerman=\@OrdinalstringNgerman
2305 \global\let\@OrdinalstringMngerman=\@OrdinalstringMgerman
2306 \global\let\@OrdinalstringFngerman=\@OrdinalstringFgerman
2307 \global\let\@OrdinalstringNngerman=\@OrdinalstringNgerman

```

### 9.1.12 fc-ngermanb.def

```

2308 \ProvidesFCLanguage{ngermanb}[2013/08/17]%
2309 \FCloadlang{german}%

```

Set ngermanb to be equivalent to german. Is it okay to do this? (I don't know the difference between the two.)

```

2310 \global\let\@ordinalMngermanb=\@ordinalMgerman
2311 \global\let\@ordinalFngermanb=\@ordinalFgerman
2312 \global\let\@ordinalNngermanb=\@ordinalNgerman
2313 \global\let\@numberstringMngermanb=\@numberstringMgerman
2314 \global\let\@numberstringFngermanb=\@numberstringFgerman
2315 \global\let\@numberstringNngermanb=\@numberstringNgerman
2316 \global\let\@NumberstringMngermanb=\@NumberstringMgerman
2317 \global\let\@NumberstringFngermanb=\@NumberstringFgerman
2318 \global\let\@NumberstringNngermanb=\@NumberstringNgerman
2319 \global\let\@ordinalstringMngermanb=\@ordinalstringMgerman
2320 \global\let\@ordinalstringFngermanb=\@ordinalstringFgerman
2321 \global\let\@ordinalstringNngermanb=\@ordinalstringNgerman
2322 \global\let\@OrdinalstringMngermanb=\@OrdinalstringMgerman
2323 \global\let\@OrdinalstringFngermanb=\@OrdinalstringFgerman
2324 \global\let\@OrdinalstringNngermanb=\@OrdinalstringNgerman

```

Load fc-ngerma.def if not already loaded

```
2325 \FCloadlang{ngerma}%
```

### 9.1.13 fc-portuges.def

Portuguese definitions

```
2326 \ProvidesFCLanguage{portuges}[2017/12/26]%
```

Define macro that converts a number or count register (first argument) to an ordinal, and stores the result in the second argument, which should be a control sequence. Masculine:

```

2327 \newcommand*\@ordinalMportuges[2] {%
2328   \ifnum#1=0\relax
2329     \edef#2{\number#1}%
2330   \else
2331     \edef#2{\number#1\relax\noexpand\fmtord{o}}%
2332   \fi
2333 }%
2334 \global\let\@ordinalMportuges\@ordinalMportuges

```

Feminine:

```

2335 \newcommand*\@ordinalFportuges[2] {%
2336   \ifnum#1=0\relax
2337     \edef#2{\number#1}%
2338   \else
2339     \edef#2{\number#1\relax\noexpand\fmtord{a}}%
2340   \fi
2341 }%
2342 \global\let\@ordinalFportuges\@ordinalFportuges

```

Make neuter same as masculine:

```
2343 \global\let\@ordinalNportuges\@ordinalMportuges
```

Convert a number to a textual representation. To make it easier, split it up into units, tens, teens and hundreds. Units (argument must be a number from 0 to 9):

```

2344 \newcommand*\@@unitstringportuges[1]{%
2345   \ifcase#1\relax
2346     zero%
2347     \or um%
2348     \or dois%
2349     \or tr\^es%
2350     \or quatro%
2351     \or cinco%
2352     \or seis%
2353     \or sete%
2354     \or oito%
2355     \or nove%
2356   \fi
2357 }%
2358 \global\let\@@unitstringportuges\@@unitstringportuges
2359 %   \end{macrocode}
2360 % As above, but for feminine:
2361 %   \begin{macrocode}
2362 \newcommand*\@@unitstringFportuges[1]{%
2363   \ifcase#1\relax
2364     zero%
2365     \or uma%
2366     \or duas%
2367     \or tr\^es%
2368     \or quatro%
2369     \or cinco%
2370     \or seis%
2371     \or sete%
2372     \or oito%
2373     \or nove%
2374   \fi
2375 }%
2376 \global\let\@@unitstringFportuges\@@unitstringFportuges

```

Tens (argument must be a number from 0 to 10):

```

2377 \newcommand*\@@tenstringportuges[1]{%
2378   \ifcase#1\relax
2379     \or dez%
2380     \or vinte%
2381     \or trinta%
2382     \or quarenta%
2383     \or cinq\"uenta%
2384     \or sessenta%
2385     \or setenta%
2386     \or oitenta%
2387     \or noventa%
2388     \or cem%
2389   \fi
2390 }%
2391 \global\let\@@tenstringportuges\@@tenstringportuges

```

Teens (argument must be a number from 0 to 9):

```
2392 \newcommand*{\@teenstringportuges[1]}{%
2393   \ifcase#1\relax
2394     dez%
2395     \or onze%
2396     \or doze%
2397     \or treze%
2398     \or catorze%
2399     \or quinze%
2400     \or dezasseis%
2401     \or dezassete%
2402     \or dezoito%
2403     \or dezanove%
2404   \fi
2405 }%
2406 \global\let\@teenstringportuges\@teenstringportuges
```

Hundreds:

```
2407 \newcommand*{\@hundredstringportuges[1]}{%
2408   \ifcase#1\relax
2409     \or cento%
2410     \or duzentos%
2411     \or trezentos%
2412     \or quatrocentos%
2413     \or quinhentos%
2414     \or seiscentos%
2415     \or setecentos%
2416     \or oitocentos%
2417     \or novecentos%
2418   \fi
2419 }%
2420 \global\let\@hundredstringportuges\@hundredstringportuges
```

Hundreds (feminine):

```
2421 \newcommand*{\@hundredstringFportuges[1]}{%
2422   \ifcase#1\relax
2423     \or cento%
2424     \or duzentas%
2425     \or trezentas%
2426     \or quatrocentas%
2427     \or quinhentas%
2428     \or seiscentas%
2429     \or setecentas%
2430     \or oitocentas%
2431     \or novecentas%
2432   \fi
2433 }%
2434 \global\let\@hundredstringFportuges\@hundredstringFportuges
```

Units (initial letter in upper case):

```

2435 \newcommand*\@@Unitstringportuges[1]{%
2436   \ifcase#1\relax
2437     Zero%
2438     \or Um%
2439     \or Dois%
2440     \or Tr\^es%
2441     \or Quatro%
2442     \or Cinco%
2443     \or Seis%
2444     \or Sete%
2445     \or Oito%
2446     \or Nove%
2447   \fi
2448 }%
2449 \global\let\@@Unitstringportuges\@@Unitstringportuges

```

As above, but feminine:

```

2450 \newcommand*\@@UnitstringFportuges[1]{%
2451   \ifcase#1\relax
2452     Zera%
2453     \or Uma%
2454     \or Duas%
2455     \or Tr\^es%
2456     \or Quatro%
2457     \or Cinco%
2458     \or Seis%
2459     \or Sete%
2460     \or Oito%
2461     \or Nove%
2462   \fi
2463 }%
2464 \global\let\@@UnitstringFportuges\@@UnitstringFportuges

```

Tens (with initial letter in upper case):

```

2465 \newcommand*\@@Tenstringportuges[1]{%
2466   \ifcase#1\relax
2467     \or Dez%
2468     \or Vinte%
2469     \or Trinta%
2470     \or Quarenta%
2471     \or Cinq\"uenta%
2472     \or Sessenta%
2473     \or Setenta%
2474     \or Oitenta%
2475     \or Noventa%
2476     \or Cem%
2477   \fi
2478 }%
2479 \global\let\@@Tenstringportuges\@@Tenstringportuges

```

Teens (with initial letter in upper case):

```

2480 \newcommand*\@@Teenstringportuges[1]{%
2481   \ifcase#1\relax
2482     Dez%
2483     \or Onze%
2484     \or Doze%
2485     \or Treze%
2486     \or Catorze%
2487     \or Quinze%
2488     \or Dezasseis%
2489     \or Dezassete%
2490     \or Dezoito%
2491     \or Dezanove%
2492   \fi
2493 }%
2494 \global\let\@@Teenstringportuges\@@Teenstringportuges

  Hundreds (with initial letter in upper case):
2495 \newcommand*\@@Hundredstringportuges[1]{%
2496   \ifcase#1\relax
2497     \or Cento%
2498     \or Duzentos%
2499     \or Trezentos%
2500     \or Quatrocentos%
2501     \or Quinhentos%
2502     \or Seiscentos%
2503     \or Setecentos%
2504     \or Oitocentos%
2505     \or Novecentos%
2506   \fi
2507 }%
2508 \global\let\@@Hundredstringportuges\@@Hundredstringportuges

  As above, but feminine:
2509 \newcommand*\@@HundredstringFportuges[1]{%
2510   \ifcase#1\relax
2511     \or Cento%
2512     \or Duzentas%
2513     \or Trezentas%
2514     \or Quatrocentas%
2515     \or Quinhentas%
2516     \or Seiscentas%
2517     \or Setecentas%
2518     \or Oitocentas%
2519     \or Novecentas%
2520   \fi
2521 }%
2522 \global\let\@@HundredstringFportuges\@@HundredstringFportuges

```

This has changed in version 1.08, so that it now stores the result in the second argument, but doesn't display anything. Since it only affects internal macros, it shouldn't affect documents

created with older versions. (These internal macros are not meant for use in documents.)

```
2523 \newcommand*{\@numberstringMportuges}[2]{%
2524   \let\@unitstring=\@unitstringportuges
2525   \let\@teenstring=\@teenstringportuges
2526   \let\@tenstring=\@tenstringportuges
2527   \let\@hundredstring=\@hundredstringportuges
2528   \def\@hundred{cem}\def\@thousand{mil}%
2529   \def\@andname{e}%
2530   \@numberstringportuges{#1}{#2}%
2531 }%
2532 \global\let\@numberstringMportuges\@numberstringMportuges
```

As above, but feminine form:

```
2533 \newcommand*{\@numberstringFportuges}[2]{%
2534   \let\@unitstring=\@unitstringFportuges
2535   \let\@teenstring=\@teenstringportuges
2536   \let\@tenstring=\@tenstringportuges
2537   \let\@hundredstring=\@hundredstringFportuges
2538   \def\@hundred{cem}\def\@thousand{mil}%
2539   \def\@andname{e}%
2540   \@numberstringportuges{#1}{#2}%
2541 }%
2542 \global\let\@numberstringFportuges\@numberstringFportuges
```

Make neuter same as masculine:

```
2543 \global\let\@numberstringNportuges\@numberstringMportuges
```

As above, but initial letters in upper case:

```
2544 \newcommand*{\@NumberstringMportuges}[2]{%
2545   \let\@unitstring=\@Unitstringportuges
2546   \let\@teenstring=\@Teenstringportuges
2547   \let\@tenstring=\@Tenstringportuges
2548   \let\@hundredstring=\@Hundredstringportuges
2549   \def\@hundred{Cem}\def\@thousand{Mil}%
2550   \def\@andname{e}%
2551   \@numberstringportuges{#1}{#2}%
2552 }%
2553 \global\let\@NumberstringMportuges\@NumberstringMportuges
```

As above, but feminine form:

```
2554 \newcommand*{\@NumberstringFportuges}[2]{%
2555   \let\@unitstring=\@UnitstringFportuges
2556   \let\@teenstring=\@Teenstringportuges
2557   \let\@tenstring=\@Tenstringportuges
2558   \let\@hundredstring=\@HundredstringFportuges
2559   \def\@hundred{Cem}\def\@thousand{Mil}%
2560   \def\@andname{e}%
2561   \@numberstringportuges{#1}{#2}%
2562 }%
2563 \global\let\@NumberstringFportuges\@NumberstringFportuges
```

Make neuter same as masculine:

```
2564 \global\let\@NumberstringNportuges\@NumberstringMportuges
```

As above, but for ordinals.

```
2565 \newcommand*\@ordinalstringMportuges}[2]{%
2566 \let\@unitthstring=\@unitthstringportuges
2567 \let\@unitstring=\@unitstringportuges
2568 \let\@teenthstring=\@teenthstringportuges
2569 \let\@tenthstring=\@tenthstringportuges
2570 \let\@hundredthstring=\@hundredthstringportuges
2571 \def\@thousandth{mil\'esimo}%
2572 \@ordinalstringportuges{#1}{#2}%
2573 }%
2574 \global\let\@ordinalstringMportuges\@ordinalstringMportuges
```

Feminine form:

```
2575 \newcommand*\@ordinalstringFportuges}[2]{%
2576 \let\@unitthstring=\@unitthstringFportuges
2577 \let\@unitstring=\@unitstringFportuges
2578 \let\@teenthstring=\@teenthstringFportuges
2579 \let\@tenthstring=\@tenthstringFportuges
2580 \let\@hundredthstring=\@hundredthstringFportuges
2581 \def\@thousandth{mil\'esima}%
2582 \@ordinalstringportuges{#1}{#2}%
2583 }%
2584 \global\let\@ordinalstringFportuges\@ordinalstringFportuges
```

Make neuter same as masculine:

```
2585 \global\let\@OrdinalstringNportuges\@OrdinalstringMportuges
```

As above, but initial letters in upper case (masculine):

```
2586 \newcommand*\@OrdinalstringMportuges}[2]{%
2587 \let\@unitthstring=\@Unitthstringportuges
2588 \let\@unitstring=\@Unitstringportuges
2589 \let\@teenthstring=\@Teenthstringportuges
2590 \let\@tenthstring=\@Tenthstringportuges
2591 \let\@hundredthstring=\@Hundredthstringportuges
2592 \def\@thousandth{Mil\'esimo}%
2593 \@Ordinalstringportuges{#1}{#2}%
2594 }%
2595 \global\let\@OrdinalstringMportuges\@OrdinalstringMportuges
```

Feminine form:

```
2596 \newcommand*\@OrdinalstringFportuges}[2]{%
2597 \let\@unitthstring=\@UnitthstringFportuges
2598 \let\@unitstring=\@UnitstringFportuges
2599 \let\@teenthstring=\@Teenthstringportuges
2600 \let\@tenthstring=\@TenthstringFportuges
2601 \let\@hundredthstring=\@HundredthstringFportuges
2602 \def\@thousandth{Mil\'esima}%
2603 \@Ordinalstringportuges{#1}{#2}%
```

```

2604 }%
2605 \global\let\@OrdinalstringFportuges\@OrdinalstringFportuges
    Make neuter same as masculine:
2606 \global\let\@OrdinalstringMportuges\@OrdinalstringMportuges
    In order to do the ordinals, split into units, teens, tens and hundreds. Units:
2607 \newcommand*\@unitthstringportuges[1]{%
2608   \ifcase#1\relax
2609     zero%
2610     \or primeiro%
2611     \or segundo%
2612     \or terceiro%
2613     \or quarto%
2614     \or quinto%
2615     \or sexto%
2616     \or s\'etimo%
2617     \or oitavo%
2618     \or nono%
2619   \fi
2620 }%
2621 \global\let\@unitthstringportuges\@unitthstringportuges
    Tens:
2622 \newcommand*\@tenthstringportuges[1]{%
2623   \ifcase#1\relax
2624     \or d\'ecimo%
2625     \or vig\'esimo%
2626     \or trig\'esimo%
2627     \or quadrag\'esimo%
2628     \or q\'uinquag\'esimo%
2629     \or sexag\'esimo%
2630     \or setuag\'esimo%
2631     \or octog\'esimo%
2632     \or nonag\'esimo%
2633   \fi
2634 }%
2635 \global\let\@tenthstringportuges\@tenthstringportuges
    Teens:
2636 \newcommand*\@teenthstringportuges[1]{%
2637   \@tenthstring{1}%
2638   \ifnum#1>0\relax
2639     -\@unitthstring{#1}%
2640   \fi
2641 }%
2642 \global\let\@teenthstringportuges\@teenthstringportuges
    Hundreds:
2643 \newcommand*\@hundredthstringportuges[1]{%
2644   \ifcase#1\relax

```

2645 \or cent\'esimo%  
 2646 \or ducent\'esimo%  
 2647 \or trecent\'esimo%  
 2648 \or quadringent\'esimo%  
 2649 \or q\'uingent\'esimo%  
 2650 \or seiscent\'esimo%  
 2651 \or setingent\'esimo%  
 2652 \or octingent\'esimo%  
 2653 \or nongent\'esimo%  
 2654 \fi  
 2655 }%  
 2656 \global\let\@@hundredthstringportuges\@@hundredthstringportuges

Units (feminine):

2657 \newcommand\*\@@unitthstringFportuges[1]{%  
 2658 \ifcase#1\relax  
 2659 zero%  
 2660 \or primeira%  
 2661 \or segunda%  
 2662 \or terceira%  
 2663 \or quarta%  
 2664 \or quinta%  
 2665 \or sexta%  
 2666 \or s\'etima%  
 2667 \or oitava%  
 2668 \or nona%  
 2669 \fi  
 2670 }%  
 2671 \global\let\@@unitthstringFportuges\@@unitthstringFportuges

Tens (feminine):

2672 \newcommand\*\@@tenthstringFportuges[1]{%  
 2673 \ifcase#1\relax  
 2674 \or d\'ecima%  
 2675 \or vig\'esima%  
 2676 \or trig\'esima%  
 2677 \or quadrag\'esima%  
 2678 \or q\'uinquag\'esima%  
 2679 \or sexag\'esima%  
 2680 \or setuag\'esima%  
 2681 \or octog\'esima%  
 2682 \or nonag\'esima%  
 2683 \fi  
 2684 }%  
 2685 \global\let\@@tenthstringFportuges\@@tenthstringFportuges

Hundreds (feminine):

2686 \newcommand\*\@@hundredthstringFportuges[1]{%  
 2687 \ifcase#1\relax  
 2688 \or cent\'esima%  
 2689 \or ducent\'esima%

```

2690 \or trecent\'esima%
2691 \or quadringent\'esima%
2692 \or quingent\'esima%
2693 \or seiscent\'esima%
2694 \or setingent\'esima%
2695 \or octingent\'esima%
2696 \or nongent\'esima%
2697 \fi
2698 }%
2699 \global\let\@@hundredthstringFportuges\@@hundredthstringFportuges

```

As above, but with initial letter in upper case. Units:

```

2700 \newcommand*\@@Unitthstringportuges[1]{%
2701 \ifcase#1\relax
2702 Zero%
2703 \or Primeiro%
2704 \or Segundo%
2705 \or Terceiro%
2706 \or Quarto%
2707 \or Quinto%
2708 \or Sexto%
2709 \or S\'etimo%
2710 \or Oitavo%
2711 \or Nono%
2712 \fi
2713 }%
2714 \global\let\@@Unitthstringportuges\@@Unitthstringportuges

```

Tens:

```

2715 \newcommand*\@@Tenthstringportuges[1]{%
2716 \ifcase#1\relax
2717 \or D\'ecimo%
2718 \or Vig\'esimo%
2719 \or Trig\'esimo%
2720 \or Quadrag\'esimo%
2721 \or Q\'uinquag\'esimo%
2722 \or Sexag\'esimo%
2723 \or Setuag\'esimo%
2724 \or Octog\'esimo%
2725 \or Nonag\'esimo%
2726 \fi
2727 }%
2728 \global\let\@@Tenthstringportuges\@@Tenthstringportuges

```

Hundreds:

```

2729 \newcommand*\@@Hundredthstringportuges[1]{%
2730 \ifcase#1\relax
2731 \or Cent\'esimo%
2732 \or Ducent\'esimo%
2733 \or Trecent\'esimo%
2734 \or Quadringent\'esimo%

```

2735 \or Q"uingent\'esimo%  
 2736 \or Seiscent\'esimo%  
 2737 \or Setingent\'esimo%  
 2738 \or Octingent\'esimo%  
 2739 \or Nongent\'esimo%  
 2740 \fi  
 2741 }%  
 2742 \global\let\@@Hundredthstringportuges\@@Hundredthstringportuges

As above, but feminine. Units:

2743 \newcommand\*\@@UnitthstringFportuges[1]{%  
 2744 \ifcase#1\relax  
 2745 Zera%  
 2746 \or Primeira%  
 2747 \or Segunda%  
 2748 \or Terceira%  
 2749 \or Quarta%  
 2750 \or Quinta%  
 2751 \or Sexta%  
 2752 \or S\'etima%  
 2753 \or Oitava%  
 2754 \or Nona%  
 2755 \fi  
 2756 }%  
 2757 \global\let\@@UnitthstringFportuges\@@UnitthstringFportuges

Tens (feminine);

2758 \newcommand\*\@@TenthstringFportuges[1]{%  
 2759 \ifcase#1\relax  
 2760 \or D\'ecima%  
 2761 \or Vig\'esima%  
 2762 \or Trig\'esima%  
 2763 \or Quadrag\'esima%  
 2764 \or Q"uinquag\'esima%  
 2765 \or Sexag\'esima%  
 2766 \or Setuag\'esima%  
 2767 \or Octog\'esima%  
 2768 \or Nonag\'esima%  
 2769 \fi  
 2770 }%  
 2771 \global\let\@@TenthstringFportuges\@@TenthstringFportuges

Hundreds (feminine):

2772 \newcommand\*\@@HundredthstringFportuges[1]{%  
 2773 \ifcase#1\relax  
 2774 \or Cent\'esima%  
 2775 \or Ducent\'esima%  
 2776 \or Trecent\'esima%  
 2777 \or Quadringent\'esima%  
 2778 \or Q"uingent\'esima%  
 2779 \or Seiscent\'esima%

```

2780 \or Setingent\'esima%
2781 \or Octingent\'esima%
2782 \or Nongent\'esima%
2783 \fi
2784 }%
2785 \global\let\@@HundredthstringFportuges\@@HundredthstringFportuges

```

This has changed in version 1.09, so that it now stores the result in the second argument (a control sequence), but it doesn't display anything. Since it only affects internal macros, it shouldn't affect documents created with older versions. (These internal macros are not meant for use in documents.)

```

2786 \newcommand*\@@numberstringportuges[2]{%
2787 \ifnum#1>99999\relax
2788 \PackageError{fmtcount}{Out of range}%
2789 {This macro only works for values less than 100000}%
2790 \else
2791 \ifnum#1<0\relax
2792 \PackageError{fmtcount}{Negative numbers not permitted}%
2793 {This macro does not work for negative numbers, however
2794 you can try typing "minus" first, and then pass the modulus of
2795 this number}%
2796 \fi
2797 \fi
2798 \def#2{}%
2799 \@strctr=#1\relax \divide\@strctr by 1000\relax
2800 \ifnum\@strctr>9\relax
    #1 is greater or equal to 10000
2801 \divide\@strctr by 10\relax
2802 \ifnum\@strctr>1\relax
2803 \let\@@fc@numstr#2\relax
2804 \protected@edef#2{\@@fc@numstr\@tenstring{\@strctr}}%
2805 \@strctr=#1 \divide\@strctr by 1000\relax
2806 \@FCmodulo{\@strctr}{10}%
2807 \ifnum\@strctr>0
2808 \ifnum\@strctr=1\relax
2809 \let\@@fc@numstr#2\relax
2810 \protected@edef#2{\@@fc@numstr\ \@andname}%
2811 \fi
2812 \let\@@fc@numstr#2\relax
2813 \protected@edef#2{\@@fc@numstr\ \@unitstring{\@strctr}}%
2814 \fi
2815 \else
2816 \@strctr=#1\relax
2817 \divide\@strctr by 1000\relax
2818 \@FCmodulo{\@strctr}{10}%
2819 \let\@@fc@numstr#2\relax
2820 \protected@edef#2{\@@fc@numstr\@teenstring{\@strctr}}%
2821 \fi
2822 \let\@@fc@numstr#2\relax

```

```

2823 \protected@edef#2{\@@fc@numstr\ \@thousand}%
2824 \else
2825 \ifnum\@strctr>0\relax
2826 \ifnum\@strctr>1\relax
2827 \let\@@fc@numstr#2\relax
2828 \protected@edef#2{\@@fc@numstr\@unitstring{\@strctr}\ }%
2829 \fi
2830 \let\@@fc@numstr#2\relax
2831 \protected@edef#2{\@@fc@numstr\@thousand}%
2832 \fi
2833 \fi
2834 \@strctr=#1\relax \@FCmodulo{\@strctr}{1000}%
2835 \divide\@strctr by 100\relax
2836 \ifnum\@strctr>0\relax
2837 \ifnum#1>1000 \relax
2838 \let\@@fc@numstr#2\relax
2839 \protected@edef#2{\@@fc@numstr\ }%
2840 \fi
2841 \@tmpstrctr=#1\relax
2842 \@FCmodulo{\@tmpstrctr}{1000}%
2843 \let\@@fc@numstr#2\relax
2844 \ifnum\@tmpstrctr=100\relax
2845 \protected@edef#2{\@@fc@numstr\@tenstring{10}}%
2846 \else
2847 \protected@edef#2{\@@fc@numstr\@hundredstring{\@strctr}}%
2848 \fi%
2849 \fi
2850 \@strctr=#1\relax \@FCmodulo{\@strctr}{100}%
2851 \ifnum#1>100\relax
2852 \ifnum\@strctr>0\relax
2853 \let\@@fc@numstr#2\relax
2854 \protected@edef#2{\@@fc@numstr\ \@andname\ }%
2855 \fi
2856 \fi
2857 \ifnum\@strctr>19\relax
2858 \divide\@strctr by 10\relax
2859 \let\@@fc@numstr#2\relax
2860 \protected@edef#2{\@@fc@numstr\@tenstring{\@strctr}}%
2861 \@strctr=#1\relax \@FCmodulo{\@strctr}{10}%
2862 \ifnum\@strctr>0
2863 \ifnum\@strctr=1\relax
2864 \let\@@fc@numstr#2\relax
2865 \protected@edef#2{\@@fc@numstr\ \@andname}%
2866 \else
2867 \ifnum#1>100\relax
2868 \let\@@fc@numstr#2\relax
2869 \protected@edef#2{\@@fc@numstr\ \@andname}%
2870 \fi
2871 \fi

```

```

2872 \let\@fc@numstr#2\relax
2873 \protected@edef#2{\@fc@numstr\ \@unitstring{\@strctr}}%
2874 \fi
2875 \else
2876 \ifnum\@strctr<10\relax
2877 \ifnum\@strctr=0\relax
2878 \ifnum#1<100\relax
2879 \let\@fc@numstr#2\relax
2880 \protected@edef#2{\@fc@numstr\@unitstring{\@strctr}}%
2881 \fi
2882 \else %(>0,<10)
2883 \let\@fc@numstr#2\relax
2884 \protected@edef#2{\@fc@numstr\@unitstring{\@strctr}}%
2885 \fi
2886 \else%>10
2887 \@FCmodulo{\@strctr}{10}%
2888 \let\@fc@numstr#2\relax
2889 \protected@edef#2{\@fc@numstr\@teenstring{\@strctr}}%
2890 \fi
2891 \fi
2892 }%
2893 \global\let\@numberstringportuges\@numberstringportuges

```

As above, but for ordinals.

```

2894 \newcommand*\@ordinalstringportuges[2]{%
2895 \@strctr=#1\relax
2896 \ifnum#1>99999
2897 \PackageError{fmtcount}{Out of range}%
2898 {This macro only works for values less than 100000}%
2899 \else
2900 \ifnum#1<0
2901 \PackageError{fmtcount}{Negative numbers not permitted}%
2902 {This macro does not work for negative numbers, however
2903 you can try typing "minus" first, and then pass the modulus of
2904 this number}%
2905 \else
2906 \def#2{}%
2907 \ifnum\@strctr>999\relax
2908 \divide\@strctr by 1000\relax
2909 \ifnum\@strctr>1\relax
2910 \ifnum\@strctr>9\relax
2911 \@tmpstrctr=\@strctr
2912 \ifnum\@strctr<20
2913 \@FCmodulo{\@tmpstrctr}{10}%
2914 \let\@fc@ordstr#2\relax
2915 \protected@edef#2{\@fc@ordstr\@teenthstring{\@tmpstrctr}}%
2916 \else
2917 \divide\@tmpstrctr by 10\relax
2918 \let\@fc@ordstr#2\relax
2919 \protected@edef#2{\@fc@ordstr\@tenthstring{\@tmpstrctr}}%

```

```

2920     \@tmpstrctr=\@strctr
2921     \@FCmodulo{\@tmpstrctr}{10}%
2922     \ifnum\@tmpstrctr>0\relax
2923         \let\@fc@ordstr#2\relax
2924         \protected@edef#2{\@fc@ordstr\@unitthstring{\@tmpstrctr}}%
2925     \fi
2926 \fi
2927 \else
2928     \let\@fc@ordstr#2\relax
2929     \protected@edef#2{\@fc@ordstr\@unitstring{\@strctr}}%
2930 \fi
2931 \fi
2932 \let\@fc@ordstr#2\relax
2933 \protected@edef#2{\@fc@ordstr\@thousandth}%
2934 \fi
2935 \@strctr=#1\relax
2936 \@FCmodulo{\@strctr}{1000}%
2937 \ifnum\@strctr>99\relax
2938     \@tmpstrctr=\@strctr
2939     \divide\@tmpstrctr by 100\relax
2940     \ifnum#1>1000\relax
2941         \let\@fc@ordstr#2\relax
2942         \protected@edef#2{\@fc@ordstr-}%
2943     \fi
2944     \let\@fc@ordstr#2\relax
2945     \protected@edef#2{\@fc@ordstr\@hundredthstring{\@tmpstrctr}}%
2946 \fi
2947 \@FCmodulo{\@strctr}{100}%
2948 \ifnum#1>99\relax
2949     \ifnum\@strctr>0\relax
2950         \let\@fc@ordstr#2\relax
2951         \protected@edef#2{\@fc@ordstr-}%
2952     \fi
2953 \fi
2954 \ifnum\@strctr>9\relax
2955     \@tmpstrctr=\@strctr
2956     \divide\@tmpstrctr by 10\relax
2957     \let\@fc@ordstr#2\relax
2958     \protected@edef#2{\@fc@ordstr\@tenthstring{\@tmpstrctr}}%
2959     \@tmpstrctr=\@strctr
2960     \@FCmodulo{\@tmpstrctr}{10}%
2961     \ifnum\@tmpstrctr>0\relax
2962         \let\@fc@ordstr#2\relax
2963         \protected@edef#2{\@fc@ordstr-\@unitthstring{\@tmpstrctr}}%
2964     \fi
2965 \else
2966     \ifnum\@strctr=0\relax
2967         \ifnum#1=0\relax
2968             \let\@fc@ordstr#2\relax

```

```

2969     \protected@edef#2{\@fc@ordstr\@unitstring{0}}%
2970     \fi
2971 \else
2972     \let\@fc@ordstr#2\relax
2973     \protected@edef#2{\@fc@ordstr\@unitthstring{\@strctr}}%
2974     \fi
2975 \fi
2976 \fi
2977 \fi
2978 }%
2979 \global\let\@ordinalstringportuges\@ordinalstringportuges

```

#### 9.1.14 fc-portuguese.def

```

2980 \ProvidesFCLanguage{portuguese}[2014/06/09]%

```

Load fc-portuges.def if not already loaded.

```

2981 \FCloadlang{portuges}%

```

Set portuguese to be equivalent to portuges.

```

2982 \global\let\@ordinalMportuguese=\@ordinalMportuges
2983 \global\let\@ordinalFportuguese=\@ordinalFportuges
2984 \global\let\@ordinalNportuguese=\@ordinalNportuges
2985 \global\let\@numberstringMportuguese=\@numberstringMportuges
2986 \global\let\@numberstringFportuguese=\@numberstringFportuges
2987 \global\let\@numberstringNportuguese=\@numberstringNportuges
2988 \global\let\@NumberstringMportuguese=\@NumberstringMportuges
2989 \global\let\@NumberstringFportuguese=\@NumberstringFportuges
2990 \global\let\@NumberstringNportuguese=\@NumberstringNportuges
2991 \global\let\@ordinalstringMportuguese=\@ordinalstringMportuges
2992 \global\let\@ordinalstringFportuguese=\@ordinalstringFportuges
2993 \global\let\@ordinalstringNportuguese=\@ordinalstringNportuges
2994 \global\let\@OrdinalstringMportuguese=\@OrdinalstringMportuges
2995 \global\let\@OrdinalstringFportuguese=\@OrdinalstringFportuges
2996 \global\let\@OrdinalstringNportuguese=\@OrdinalstringNportuges

```

#### 9.1.15 fc-spanish.def

Spanish definitions

```

2997 \ProvidesFCLanguage{spanish}[2016/01/12]%

```

Define macro that converts a number or count register (first argument) to an ordinal, and stores the result in the second argument, which must be a control sequence. Masculine:

```

2998 \newcommand*\@ordinalMspanish[2]{%
2999   \edef#2{\number#1\relax\noexpand\fmtord{o}}%
3000 }%
3001 \global\let\@ordinalMspanish\@ordinalMspanish

```

Feminine:

```

3002 \newcommand{\@ordinalFspanish}[2]{%
3003   \edef#2{\number#1\relax\noexpand\fmtord{a}}%
3004 }%

```

```
3005 \global\let\@OrdinalFspanish\@OrdinalFspanish
```

Make neuter same as masculine:

```
3006 \global\let\@OrdinalNspanish\@OrdinalMspanish
```

Convert a number to text. The easiest way to do this is to break it up into units, tens, teens, twenties and hundreds. Units (argument must be a number from 0 to 9):

```
3007 \newcommand*\@unitstringspanish[1]{%
```

```
3008   \ifcase#1\relax
```

```
3009     cero%
```

```
3010     \or uno%
```

```
3011     \or dos%
```

```
3012     \or tres%
```

```
3013     \or cuatro%
```

```
3014     \or cinco%
```

```
3015     \or seis%
```

```
3016     \or siete%
```

```
3017     \or ocho%
```

```
3018     \or nueve%
```

```
3019   \fi
```

```
3020 }%
```

```
3021 \global\let\@unitstringspanish\@unitstringspanish
```

Feminine:

```
3022 \newcommand*\@unitstringFspanish[1]{%
```

```
3023   \ifcase#1\relax
```

```
3024     cera%
```

```
3025     \or una%
```

```
3026     \or dos%
```

```
3027     \or tres%
```

```
3028     \or cuatro%
```

```
3029     \or cinco%
```

```
3030     \or seis%
```

```
3031     \or siete%
```

```
3032     \or ocho%
```

```
3033     \or nueve%
```

```
3034   \fi
```

```
3035 }%
```

```
3036 \global\let\@unitstringFspanish\@unitstringFspanish
```

Tens (argument must go from 1 to 10):

```
3037 \newcommand*\@tenstringspanish[1]{%
```

```
3038   \ifcase#1\relax
```

```
3039     \or diez%
```

```
3040     \or veinte%
```

```
3041     \or treinta%
```

```
3042     \or cuarenta%
```

```
3043     \or cincuenta%
```

```
3044     \or sesenta%
```

```
3045     \or setenta%
```

```
3046     \or ochenta%
```

```

3047 \or noventa%
3048 \or cien%
3049 \fi
3050 }%
3051 \global\let\@@tenstringspanish\@@tenstringspanish

  Teens:
3052 \newcommand*\@@teenstringspanish[1]{%
3053 \ifcase#1\relax
3054 diez%
3055 \or once%
3056 \or doce%
3057 \or trece%
3058 \or catorce%
3059 \or quince%
3060 \or diecis\'eis%
3061 \or diecisiete%
3062 \or dieciocho%
3063 \or diecinueve%
3064 \fi
3065 }%
3066 \global\let\@@teenstringspanish\@@teenstringspanish

  Twenties:
3067 \newcommand*\@@twentystringspanish[1]{%
3068 \ifcase#1\relax
3069 veinte%
3070 \or veintiuno%
3071 \or veintid\'os%
3072 \or veintitr\'es%
3073 \or veinticuatro%
3074 \or veinticinco%
3075 \or veintis\'eis%
3076 \or veintisiete%
3077 \or veintiocho%
3078 \or veintinueve%
3079 \fi
3080 }%
3081 \global\let\@@twentystringspanish\@@twentystringspanish

  Feminine form:
3082 \newcommand*\@@twentystringFspanish[1]{%
3083 \ifcase#1\relax
3084 veinte%
3085 \or veintiuna%
3086 \or veintid\'os%
3087 \or veintitr\'es%
3088 \or veinticuatro%
3089 \or veinticinco%
3090 \or veintis\'eis%
3091 \or veintisiete%

```

3092 \or veintiocho%  
3093 \or veintinueve%  
3094 \fi  
3095 }%  
3096 \global\let\@@twentystringFspanish\@@twentystringFspanish

Hundreds:

3097 \newcommand\*\@@hundredstringspanish[1]{%  
3098 \ifcase#1\relax  
3099 \or ciento%  
3100 \or doscientos%  
3101 \or trescientos%  
3102 \or cuatrocientos%  
3103 \or quinientos%  
3104 \or seiscientos%  
3105 \or setecientos%  
3106 \or ochocientos%  
3107 \or novecientos%  
3108 \fi  
3109 }%  
3110 \global\let\@@hundredstringspanish\@@hundredstringspanish

Feminine form:

3111 \newcommand\*\@@hundredstringFspanish[1]{%  
3112 \ifcase#1\relax  
3113 \or cienta%  
3114 \or doscientas%  
3115 \or trescientas%  
3116 \or cuatrocientas%  
3117 \or quinientas%  
3118 \or seiscientas%  
3119 \or setecientas%  
3120 \or ochocientas%  
3121 \or novecientas%  
3122 \fi  
3123 }%  
3124 \global\let\@@hundredstringFspanish\@@hundredstringFspanish

As above, but with initial letter uppercase:

3125 \newcommand\*\@@Unitstringspanish[1]{%  
3126 \ifcase#1\relax  
3127 Cero%  
3128 \or Uno%  
3129 \or Dos%  
3130 \or Tres%  
3131 \or Cuatro%  
3132 \or Cinco%  
3133 \or Seis%  
3134 \or Siete%  
3135 \or Ocho%  
3136 \or Nueve%

3137 \fi  
3138 }%  
3139 \global\let\@@Unitstringspanish\@@Unitstringspanish

Feminine form:

3140 \newcommand\*\@@UnitstringFspanish[1]{%  
3141 \ifcase#1\relax  
3142 Cera%  
3143 \or Una%  
3144 \or Dos%  
3145 \or Tres%  
3146 \or Cuatro%  
3147 \or Cinco%  
3148 \or Seis%  
3149 \or Siete%  
3150 \or Ocho%  
3151 \or Nueve%  
3152 \fi  
3153 }%  
3154 \global\let\@@UnitstringFspanish\@@UnitstringFspanish

Tens:

3155%\changes{2.0}{2012-06-18}{fixed spelling mistake (correction  
3156%provided by Fernando Maldonado)}  
3157 \newcommand\*\@@Tenstringspanish[1]{%  
3158 \ifcase#1\relax  
3159 \or Diez%  
3160 \or Veinte%  
3161 \or Treinta%  
3162 \or Cuarenta%  
3163 \or Cincuenta%  
3164 \or Sesenta%  
3165 \or Setenta%  
3166 \or Ochenta%  
3167 \or Noventa%  
3168 \or Cien%  
3169 \fi  
3170 }%  
3171 \global\let\@@Tenstringspanish\@@Tenstringspanish

Teens:

3172 \newcommand\*\@@Teenstringspanish[1]{%  
3173 \ifcase#1\relax  
3174 Diez%  
3175 \or Once%  
3176 \or Doce%  
3177 \or Trece%  
3178 \or Catorce%  
3179 \or Quince%  
3180 \or Diecis\'eis%  
3181 \or Diecisiete%

```

3182 \or Dieciocho%
3183 \or Diecinueve%
3184 \fi
3185 }%
3186 \global\let\@@Teenstringspanish\@@Teenstringspanish

```

Twenties:

```

3187 \newcommand*\@@Twentystringspanish[1]{%
3188 \ifcase#1\relax
3189 \or Veinte%
3190 \or Veintiuno%
3191 \or Veintid\'os%
3192 \or Veintitr\'es%
3193 \or Veinticuatro%
3194 \or Veinticinco%
3195 \or Veintis\'eis%
3196 \or Veintisiete%
3197 \or Veintiocho%
3198 \or Veintinueve%
3199 \fi
3200 }%
3201 \global\let\@@Twentystringspanish\@@Twentystringspanish

```

Feminine form:

```

3202 \newcommand*\@@TwentystringFspanish[1]{%
3203 \ifcase#1\relax
3204 \or Veinte%
3205 \or Veintiuna%
3206 \or Veintid\'os%
3207 \or Veintitr\'es%
3208 \or Veinticuatro%
3209 \or Veinticinco%
3210 \or Veintis\'eis%
3211 \or Veintisiete%
3212 \or Veintiocho%
3213 \or Veintinueve%
3214 \fi
3215 }%
3216 \global\let\@@TwentystringFspanish\@@TwentystringFspanish

```

Hundreds:

```

3217 \newcommand*\@@Hundredstringspanish[1]{%
3218 \ifcase#1\relax
3219 \or Ciento%
3220 \or Doscientos%
3221 \or Trescientos%
3222 \or Cuatrocientos%
3223 \or Quinientos%
3224 \or Seiscientos%
3225 \or Setecientos%
3226 \or Ochocientos%

```

```

3227 \or Novecientos%
3228 \fi
3229 }%
3230 \global\let\@@Hundredstringspanish\@@Hundredstringspanish

```

Feminine form:

```

3231 \newcommand*\@@HundredstringFspanish[1]{%
3232 \ifcase#1\relax
3233 \or Cienta%
3234 \or Doscientas%
3235 \or Trescientas%
3236 \or Cuatrocientas%
3237 \or Quinientas%
3238 \or Seiscientas%
3239 \or Setecientas%
3240 \or Ochocientas%
3241 \or Novecientas%
3242 \fi
3243 }%
3244 \global\let\@@HundredstringFspanish\@@HundredstringFspanish

```

This has changed in version 1.09, so that it now stores the result in the second argument, but doesn't display anything. Since it only affects internal macros, it shouldn't affect documents created with older versions. (These internal macros are not meant for use in documents.)

```

3245 \newcommand*\@@numberstringMspanish}[2]{%
3246 \let\@unitstring=\@@unitstringspanish
3247 \let\@teenstring=\@@teenstringspanish
3248 \let\@tenstring=\@@tenstringspanish
3249 \let\@twentystring=\@@twentystringspanish
3250 \let\@hundredstring=\@@hundredstringspanish
3251 \def\@hundred{cien}\def\@thousand{mil}%
3252 \def\@andname{y}%
3253 \@@numberstringspanish{#1}{#2}%
3254 }%
3255 \global\let\@numberstringMspanish\@numberstringMspanish

```

Feminine form:

```

3256 \newcommand*\@@numberstringFspanish}[2]{%
3257 \let\@unitstring=\@@unitstringFspanish
3258 \let\@teenstring=\@@teenstringspanish
3259 \let\@tenstring=\@@tenstringspanish
3260 \let\@twentystring=\@@twentystringFspanish
3261 \let\@hundredstring=\@@hundredstringFspanish
3262 \def\@hundred{cien}\def\@thousand{mil}%
3263 \def\@andname{b}%
3264 \@@numberstringspanish{#1}{#2}%
3265 }%
3266 \global\let\@numberstringFspanish\@numberstringFspanish

```

Make neuter same as masculine:

```

3267 \global\let\@numberstringNspanish\@numberstringMspanish

```

As above, but initial letters in upper case:

```
3268 \newcommand*{\@NumberstringMspanish}[2]{%
3269   \let\@unitstring=\@Unitstringspanish
3270   \let\@teenstring=\@Teenstringspanish
3271   \let\@tenstring=\@Tenstringspanish
3272   \let\@twentystring=\@Twentystringspanish
3273   \let\@hundredstring=\@Hundredstringspanish
3274   \def\@andname{y}%
3275   \def\@hundred{Cien}\def\@thousand{Mil}%
3276   \@numberstringspanish{#1}{#2}%
3277 }%
3278 \global\let\@NumberstringMspanish\@NumberstringMspanish
```

Feminine form:

```
3279 \newcommand*{\@NumberstringFspanish}[2]{%
3280   \let\@unitstring=\@UnitstringFspanish
3281   \let\@teenstring=\@Teenstringspanish
3282   \let\@tenstring=\@Tenstringspanish
3283   \let\@twentystring=\@TwentystringFspanish
3284   \let\@hundredstring=\@HundredstringFspanish
3285   \def\@andname{b}%
3286   \def\@hundred{Cien}\def\@thousand{Mil}%
3287   \@numberstringspanish{#1}{#2}%
3288 }%
3289 \global\let\@NumberstringFspanish\@NumberstringFspanish
```

Make neuter same as masculine:

```
3290 \global\let\@NumberstringNspanish\@NumberstringMspanish
```

As above, but for ordinals.

```
3291 \newcommand*{\@ordinalstringMspanish}[2]{%
3292   \let\@unitthstring=\@unitthstringspanish
3293   \let\@unitstring=\@unitstringspanish
3294   \let\@teenthstring=\@teenthstringspanish
3295   \let\@tenthstring=\@tenthstringspanish
3296   \let\@hundredthstring=\@hundredthstringspanish
3297   \def\@thousandth{mil'esimo}%
3298   \@ordinalstringspanish{#1}{#2}%
3299 }%
3300 \global\let\@ordinalstringMspanish\@ordinalstringMspanish
```

Feminine form:

```
3301 \newcommand*{\@ordinalstringFspanish}[2]{%
3302   \let\@unitthstring=\@unitthstringFspanish
3303   \let\@unitstring=\@unitstringFspanish
3304   \let\@teenthstring=\@teenthstringFspanish
3305   \let\@tenthstring=\@tenthstringFspanish
3306   \let\@hundredthstring=\@hundredthstringFspanish
3307   \def\@thousandth{mil'esima}%
3308   \@ordinalstringspanish{#1}{#2}%
3309 }%
```

```

3310 \global\let\@OrdinalstringFspanish\@OrdinalstringFspanish
    Make neuter same as masculine:
3311 \global\let\@OrdinalstringMspanish\@OrdinalstringMspanish
    As above, but with initial letters in upper case.
3312 \newcommand*{\@OrdinalstringMspanish}[2]{%
3313   \let\@unitthstring=\@Unitthstringspanish
3314   \let\@unitstring=\@Unitstringspanish
3315   \let\@teenthstring=\@Teenthstringspanish
3316   \let\@tenthstring=\@Tenthstringspanish
3317   \let\@hundredthstring=\@Hundredthstringspanish
3318   \def\@thousandth{Mil\'esimo}%
3319   \@Ordinalstringspanish{#1}{#2}%
3320 }
3321 \global\let\@OrdinalstringMspanish\@OrdinalstringMspanish
    Feminine form:
3322 \newcommand*{\@OrdinalstringFspanish}[2]{%
3323   \let\@unitthstring=\@UnitthstringFspanish
3324   \let\@unitstring=\@UnitstringFspanish
3325   \let\@teenthstring=\@TeenthstringFspanish
3326   \let\@tenthstring=\@TenthstringFspanish
3327   \let\@hundredthstring=\@HundredthstringFspanish
3328   \def\@thousandth{Mil\'esima}%
3329   \@Ordinalstringspanish{#1}{#2}%
3330 }%
3331 \global\let\@OrdinalstringFspanish\@OrdinalstringFspanish
    Make neuter same as masculine:
3332 \global\let\@OrdinalstringNspanish\@OrdinalstringMspanish
    Code for convert numbers into textual ordinals. As before, it is easier to split it into units,
    tens, teens and hundreds. Units:
3333 \newcommand*\@@unitthstringspanish[1]{%
3334   \ifcase#1\relax
3335     cero%
3336     \or primero%
3337     \or segundo%
3338     \or tercero%
3339     \or cuarto%
3340     \or quinto%
3341     \or sexto%
3342     \or s\'eptimo%
3343     \or octavo%
3344     \or noveno%
3345   \fi
3346 }%
3347 \global\let\@@unitthstringspanish\@@unitthstringspanish
    Tens:
3348 \newcommand*\@@tenthstringspanish[1]{%

```

```

3349 \ifcase#1\relax
3350 \or d\'ecimo%
3351 \or vig\'esimo%
3352 \or trig\'esimo%
3353 \or cuadrag\'esimo%
3354 \or quincuag\'esimo%
3355 \or sexag\'esimo%
3356 \or septuag\'esimo%
3357 \or octog\'esimo%
3358 \or nonag\'esimo%
3359 \fi
3360 }%
3361 \global\let\@@tenthstringspanish\@@tenthstringspanish

```

Teens:

```

3362 \newcommand*\@@teenthstringspanish[1]{%
3363 \ifcase#1\relax
3364 d\'ecimo%
3365 \or und\'ecimo%
3366 \or duod\'ecimo%
3367 \or decimotercero%
3368 \or decimocuarto%
3369 \or decimoquinto%
3370 \or decimosexto%
3371 \or decimos\'eptimo%
3372 \or decimoctavo%
3373 \or decimonoveno%
3374 \fi
3375 }%
3376 \global\let\@@teenthstringspanish\@@teenthstringspanish

```

Hundreds:

```

3377 \newcommand*\@@hundredthstringspanish[1]{%
3378 \ifcase#1\relax
3379 \or cent\'esimo%
3380 \or ducent\'esimo%
3381 \or tricent\'esimo%
3382 \or cuadringent\'esimo%
3383 \or quingent\'esimo%
3384 \or sexcent\'esimo%
3385 \or septing\'esimo%
3386 \or octingent\'esimo%
3387 \or noningent\'esimo%
3388 \fi
3389 }%
3390 \global\let\@@hundredthstringspanish\@@hundredthstringspanish

```

Units (feminine):

```

3391 \newcommand*\@@unitthstringFspanish[1]{%
3392 \ifcase#1\relax
3393 cera%

```

3394 \or primera%  
 3395 \or segunda%  
 3396 \or tercera%  
 3397 \or cuarta%  
 3398 \or quinta%  
 3399 \or sexta%  
 3400 \or s\septima%  
 3401 \or octava%  
 3402 \or novena%  
 3403 \fi  
 3404 }%  
 3405 \global\let\@@unitthstringFspanish\@@unitthstringFspanish

Tens (feminine):

3406 \newcommand\*\@@tenthstringFspanish[1]{%  
 3407 \ifcase#1\relax  
 3408 \or d\decima%  
 3409 \or vig\vesima%  
 3410 \or trig\vesima%  
 3411 \or cuadrag\vesima%  
 3412 \or quincuag\vesima%  
 3413 \or sexag\vesima%  
 3414 \or septuag\vesima%  
 3415 \or octog\vesima%  
 3416 \or nonag\vesima%  
 3417 \fi  
 3418 }%  
 3419 \global\let\@@tenthstringFspanish\@@tenthstringFspanish

Teens (feminine)

3420 \newcommand\*\@@teenthstringFspanish[1]{%  
 3421 \ifcase#1\relax  
 3422 d\decima%  
 3423 \or und\decima%  
 3424 \or duod\decima%  
 3425 \or decimotercera%  
 3426 \or decimocuarta%  
 3427 \or decimoquinta%  
 3428 \or decimosexta%  
 3429 \or decimos\septima%  
 3430 \or decimoctava%  
 3431 \or decimonovena%  
 3432 \fi  
 3433 }%  
 3434 \global\let\@@teenthstringFspanish\@@teenthstringFspanish

Hundreds (feminine)

3435 \newcommand\*\@@hundredthstringFspanish[1]{%  
 3436 \ifcase#1\relax  
 3437 \or cent\vesima%  
 3438 \or ducent\vesima%

```

3439 \or tricent\'esima%
3440 \or cuadingent\'esima%
3441 \or quingent\'esima%
3442 \or sexcent\'esima%
3443 \or septing\'esima%
3444 \or octingent\'esima%
3445 \or noningent\'esima%
3446 \fi
3447 }%
3448 \global\let\@@hundredthstringFspanish\@@hundredthstringFspanish

```

As above, but with initial letters in upper case

```

3449 \newcommand*\@@Unitthstringspanish[1]{%
3450 \ifcase#1\relax
3451 Cero%
3452 \or Primero%
3453 \or Segundo%
3454 \or Tercero%
3455 \or Cuarto%
3456 \or Quinto%
3457 \or Sexto%
3458 \or S\'eptimo%
3459 \or Octavo%
3460 \or Noveno%
3461 \fi
3462 }%
3463 \global\let\@@Unitthstringspanish\@@Unitthstringspanish

```

Tens:

```

3464 \newcommand*\@@Tenthstringspanish[1]{%
3465 \ifcase#1\relax
3466 \or D\'ecimo%
3467 \or Vig\'esimo%
3468 \or Trig\'esimo%
3469 \or Cuadrag\'esimo%
3470 \or Quincuag\'esimo%
3471 \or Sexag\'esimo%
3472 \or Septuag\'esimo%
3473 \or Octog\'esimo%
3474 \or Nonag\'esimo%
3475 \fi
3476 }%
3477 \global\let\@@Tenthstringspanish\@@Tenthstringspanish

```

Teens:

```

3478 \newcommand*\@@Teenthstringspanish[1]{%
3479 \ifcase#1\relax
3480 D\'ecimo%
3481 \or Und\'ecimo%
3482 \or Duod\'ecimo%
3483 \or Decimotercero%

```

3484 \or Decimocuarto%  
 3485 \or Decimoquinto%  
 3486 \or Decimosexto%  
 3487 \or Decimos\'eptimo%  
 3488 \or Decimooctavo%  
 3489 \or Decimonoveno%  
 3490 \fi  
 3491 }%  
 3492 \global\let\@@Teenthstringspanish\@@Teenthstringspanish

#### Hundreds

3493 \newcommand\*\@@Hundredthstringspanish[1]{%  
 3494 \ifcase#1\relax  
 3495 \or Cent\'esimo%  
 3496 \or Ducent\'esimo%  
 3497 \or Tricent\'esimo%  
 3498 \or Cuadringent\'esimo%  
 3499 \or Quingent\'esimo%  
 3500 \or Sexcent\'esimo%  
 3501 \or Septing\'esimo%  
 3502 \or Octingent\'esimo%  
 3503 \or Noningent\'esimo%  
 3504 \fi  
 3505 }%  
 3506 \global\let\@@Hundredthstringspanish\@@Hundredthstringspanish

As above, but feminine.

3507 \newcommand\*\@@UnitthstringFspanish[1]{%  
 3508 \ifcase#1\relax  
 3509 Cera%  
 3510 \or Primera%  
 3511 \or Segunda%  
 3512 \or Tercera%  
 3513 \or Cuarta%  
 3514 \or Quinta%  
 3515 \or Sexta%  
 3516 \or S\'eptima%  
 3517 \or Octava%  
 3518 \or Novena%  
 3519 \fi  
 3520 }%  
 3521 \global\let\@@UnitthstringFspanish\@@UnitthstringFspanish

#### Tens (feminine)

3522 \newcommand\*\@@TenthstringFspanish[1]{%  
 3523 \ifcase#1\relax  
 3524 \or D\'esima%  
 3525 \or Vig\'esima%  
 3526 \or Trig\'esima%  
 3527 \or Cuadrag\'esima%  
 3528 \or Quincuag\'esima%

```

3529 \or Sexag\'esima%
3530 \or Septuag\'esima%
3531 \or Octog\'esima%
3532 \or Nonag\'esima%
3533 \fi
3534 }%
3535 \global\let\@@TenthstringFspanish\@@TenthstringFspanish

```

Teens (feminine):

```

3536 \newcommand*\@@TeenthstringFspanish[1]{%
3537 \ifcase#1\relax
3538 D\'esima%
3539 \or Und\'esima%
3540 \or Duod\'esima%
3541 \or Decimotercera%
3542 \or Decimocuarta%
3543 \or Decimoquinta%
3544 \or Decimosexta%
3545 \or Decimos\'eptima%
3546 \or Decimoctava%
3547 \or Decimonovena%
3548 \fi
3549 }%
3550 \global\let\@@TeenthstringFspanish\@@TeenthstringFspanish

```

Hundreds (feminine):

```

3551 \newcommand*\@@HundredthstringFspanish[1]{%
3552 \ifcase#1\relax
3553 \or Cent\'esima%
3554 \or Ducent\'esima%
3555 \or Tricent\'esima%
3556 \or Cuadringent\'esima%
3557 \or Quingent\'esima%
3558 \or Sexcent\'esima%
3559 \or Septing\'esima%
3560 \or Octingent\'esima%
3561 \or Noningent\'esima%
3562 \fi
3563 }%
3564 \global\let\@@HundredthstringFspanish\@@HundredthstringFspanish

```

This has changed in version 1.09, so that it now stores the results in the second argument (which must be a control sequence), but it doesn't display anything. Since it only affects internal macros, it shouldn't affect documents created with older versions. (These internal macros are not meant for use in documents.)

```

3565 \newcommand*\@@numberstringspanish[2]{%
3566 \ifnum#1>99999
3567 \PackageError{fmtcount}{Out of range}%
3568 {This macro only works for values less than 100000}%
3569 \else

```

```

3570 \ifnum#1<0
3571 \PackageError{fmtcount}{Negative numbers not permitted}%
3572 {This macro does not work for negative numbers, however
3573 you can try typing "minus" first, and then pass the modulus of
3574 this number}%
3575 \fi
3576 \fi
3577 \def#2{}%
3578 \@strctr=#1\relax \divide\@strctr by 1000\relax
3579 \ifnum\@strctr>9
    #1 is greater or equal to 10000
3580 \divide\@strctr by 10
3581 \ifnum\@strctr>1
3582 \let\@fc@numstr#2\relax
3583 \edef#2{\@fc@numstr\@tenstring{\@strctr}}%
3584 \@strctr=#1 \divide\@strctr by 1000\relax
3585 \@FCmodulo{\@strctr}{10}%
3586 \ifnum\@strctr>0\relax
3587 \let\@fc@numstr#2\relax
3588 \edef#2{\@fc@numstr\ \@andname\ \@unitstring{\@strctr}}%
3589 \fi
3590 \else
3591 \@strctr=#1\relax
3592 \divide\@strctr by 1000\relax
3593 \@FCmodulo{\@strctr}{10}%
3594 \let\@fc@numstr#2\relax
3595 \edef#2{\@fc@numstr\@teenstring{\@strctr}}%
3596 \fi
3597 \let\@fc@numstr#2\relax
3598 \edef#2{\@fc@numstr\ \@thousand}%
3599 \else
3600 \ifnum\@strctr>0\relax
3601 \ifnum\@strctr>1\relax
3602 \let\@fc@numstr#2\relax
3603 \edef#2{\@fc@numstr\@unitstring{\@strctr}\ }%
3604 \fi
3605 \let\@fc@numstr#2\relax
3606 \edef#2{\@fc@numstr\@thousand}%
3607 \fi
3608 \fi
3609 \@strctr=#1\relax \@FCmodulo{\@strctr}{1000}%
3610 \divide\@strctr by 100\relax
3611 \ifnum\@strctr>0\relax
3612 \ifnum#1>1000\relax
3613 \let\@fc@numstr#2\relax
3614 \edef#2{\@fc@numstr\ }%
3615 \fi
3616 \@tmpstrctr=#1\relax
3617 \@FCmodulo{\@tmpstrctr}{1000}%

```

```

3618 \ifnum\tmpstrctr=100\relax
3619   \let\@fc@numstr#2\relax
3620   \edef#2{\@fc@numstr\@tenstring{10}}%
3621 \else
3622   \let\@fc@numstr#2\relax
3623   \edef#2{\@fc@numstr\@hundredstring{\@strctr}}%
3624 \fi
3625 \fi
3626 \@strctr=#1\relax \@FCmodulo{\@strctr}{100}%
3627 \ifnum#1>100\relax
3628   \ifnum\@strctr>0\relax
3629     \let\@fc@numstr#2\relax
3630     \edef#2{\@fc@numstr\ }%
3631   \fi
3632 \fi
3633 \ifnum\@strctr>29\relax
3634   \divide\@strctr by 10\relax
3635   \let\@fc@numstr#2\relax
3636   \edef#2{\@fc@numstr\@tenstring{\@strctr}}%
3637   \@strctr=#1\relax \@FCmodulo{\@strctr}{10}%
3638   \ifnum\@strctr>0\relax
3639     \let\@fc@numstr#2\relax
3640     \edef#2{\@fc@numstr\ \@andname\ \@unitstring{\@strctr}}%
3641   \fi
3642 \else
3643   \ifnum\@strctr<10\relax
3644     \ifnum\@strctr=0\relax
3645       \ifnum#1<100\relax
3646         \let\@fc@numstr#2\relax
3647         \edef#2{\@fc@numstr\@unitstring{\@strctr}}%
3648       \fi
3649     \else
3650       \let\@fc@numstr#2\relax
3651       \edef#2{\@fc@numstr\@unitstring{\@strctr}}%
3652     \fi
3653   \else
3654     \ifnum\@strctr>19\relax
3655       \@FCmodulo{\@strctr}{10}%
3656       \let\@fc@numstr#2\relax
3657       \edef#2{\@fc@numstr\@twentystring{\@strctr}}%
3658     \else
3659       \@FCmodulo{\@strctr}{10}%
3660       \let\@fc@numstr#2\relax
3661       \edef#2{\@fc@numstr\@teenstring{\@strctr}}%
3662     \fi
3663   \fi
3664 \fi
3665 }%
3666 \global\let\@numberstringspanish\@numberstringspanish

```

As above, but for ordinals

```
3667 \newcommand*{\@@ordinalstringspanish[2]}{%
3668 \@strctr=#1\relax
3669 \ifnum#1>99999
3670 \PackageError{fmtcount}{Out of range}%
3671 {This macro only works for values less than 100000}%
3672 \else
3673 \ifnum#1<0
3674 \PackageError{fmtcount}{Negative numbers not permitted}%
3675 {This macro does not work for negative numbers, however
3676 you can try typing "minus" first, and then pass the modulus of
3677 this number}%
3678 \else
3679 \def#2{}%
3680 \ifnum\@strctr>999\relax
3681 \divide\@strctr by 1000\relax
3682 \ifnum\@strctr>1\relax
3683 \ifnum\@strctr>9\relax
3684 \@tmpstrctr=\@strctr
3685 \ifnum\@strctr<20
3686 \@FCmodulo{\@tmpstrctr}{10}%
3687 \let\@@fc@ordstr#2\relax
3688 \edef#2{\@@fc@ordstr\@teenthstring{\@tmpstrctr}}%
3689 \else
3690 \divide\@tmpstrctr by 10\relax
3691 \let\@@fc@ordstr#2\relax
3692 \edef#2{\@@fc@ordstr\@tenthstring{\@tmpstrctr}}%
3693 \@tmpstrctr=\@strctr
3694 \@FCmodulo{\@tmpstrctr}{10}%
3695 \ifnum\@tmpstrctr>0\relax
3696 \let\@@fc@ordstr#2\relax
3697 \edef#2{\@@fc@ordstr\@unitthstring{\@tmpstrctr}}%
3698 \fi
3699 \fi
3700 \else
3701 \let\@@fc@ordstr#2\relax
3702 \edef#2{\@@fc@ordstr\@unitstring{\@strctr}}%
3703 \fi
3704 \fi
3705 \let\@@fc@ordstr#2\relax
3706 \edef#2{\@@fc@ordstr\@thousandth}%
3707 \fi
3708 \@strctr=#1\relax
3709 \@FCmodulo{\@strctr}{1000}%
3710 \ifnum\@strctr>99\relax
3711 \@tmpstrctr=\@strctr
3712 \divide\@tmpstrctr by 100\relax
3713 \ifnum#1>1000\relax
3714 \let\@@fc@ordstr#2\relax
```

```

3715 \edef#2{\@fc@ordstr\ }%
3716 \fi
3717 \let\@fc@ordstr#2\relax
3718 \edef#2{\@fc@ordstr\@hundredthstring{\@tmpstrctr}}%
3719 \fi
3720 \@FCmodulo{\@strctr}{100}%
3721 \ifnum#1>99\relax
3722 \ifnum\@strctr>0\relax
3723 \let\@fc@ordstr#2\relax
3724 \edef#2{\@fc@ordstr\ }%
3725 \fi
3726 \fi
3727 \ifnum\@strctr>19\relax
3728 \@tmpstrctr=\@strctr
3729 \divide\@tmpstrctr by 10\relax
3730 \let\@fc@ordstr#2\relax
3731 \edef#2{\@fc@ordstr\@tenthstring{\@tmpstrctr}}%
3732 \@tmpstrctr=\@strctr
3733 \@FCmodulo{\@tmpstrctr}{10}%
3734 \ifnum\@tmpstrctr>0\relax
3735 \let\@fc@ordstr#2\relax
3736 \edef#2{\@fc@ordstr\ \@unitthstring{\@tmpstrctr}}%
3737 \fi
3738 \else
3739 \ifnum\@strctr>9\relax
3740 \@FCmodulo{\@strctr}{10}%
3741 \let\@fc@ordstr#2\relax
3742 \edef#2{\@fc@ordstr\@teenthstring{\@strctr}}%
3743 \else
3744 \ifnum\@strctr=0\relax
3745 \ifnum#1=0\relax
3746 \let\@fc@ordstr#2\relax
3747 \edef#2{\@fc@ordstr\@unitstring{0}}%
3748 \fi
3749 \else
3750 \let\@fc@ordstr#2\relax
3751 \edef#2{\@fc@ordstr\@unitthstring{\@strctr}}%
3752 \fi
3753 \fi
3754 \fi
3755 \fi
3756 \fi
3757 }%
3758 \global\let\@@ordinalstringspanish\@@ordinalstringspanish

```

### 9.1.16 fc-UKenglish.def

#### English definitions

```

3759 \ProvidesFCLanguage{UKenglish}[2013/08/17]%

```

Loaded fc-english.def if not already loaded

```
3760 \FCloadlang{english}%
```

These are all just synonyms for the commands provided by fc-english.def.

```
3761 \global\let\@ordinalMUKenglish\@ordinalMenglish
3762 \global\let\@ordinalFUKenglish\@ordinalMenglish
3763 \global\let\@ordinalNUKenglish\@ordinalMenglish
3764 \global\let\@numberstringMUKenglish\@numberstringMenglish
3765 \global\let\@numberstringFUKenglish\@numberstringMenglish
3766 \global\let\@numberstringNUKenglish\@numberstringMenglish
3767 \global\let\@NumberstringMUKenglish\@NumberstringMenglish
3768 \global\let\@NumberstringFUKenglish\@NumberstringMenglish
3769 \global\let\@NumberstringNUKenglish\@NumberstringMenglish
3770 \global\let\@ordinalstringMUKenglish\@ordinalstringMenglish
3771 \global\let\@ordinalstringFUKenglish\@ordinalstringMenglish
3772 \global\let\@ordinalstringNUKenglish\@ordinalstringMenglish
3773 \global\let\@OrdinalstringMUKenglish\@OrdinalstringMenglish
3774 \global\let\@OrdinalstringFUKenglish\@OrdinalstringMenglish
3775 \global\let\@OrdinalstringNUKenglish\@OrdinalstringMenglish
```

### 9.1.17 fc-USenglish.def

US English definitions

```
3776 \ProvidesFCLanguage{USenglish}[2013/08/17]%
```

Loaded fc-english.def if not already loaded

```
3777 \FCloadlang{english}%
```

These are all just synonyms for the commands provided by fc-english.def. (This needs fixing as there are some differences between UK and US number strings.)

```
3778 \global\let\@ordinalMUSenglish\@ordinalMenglish
3779 \global\let\@ordinalFUSenglish\@ordinalMenglish
3780 \global\let\@ordinalNUSenglish\@ordinalMenglish
3781 \global\let\@numberstringMUSenglish\@numberstringMenglish
3782 \global\let\@numberstringFUSenglish\@numberstringMenglish
3783 \global\let\@numberstringNUSenglish\@numberstringMenglish
3784 \global\let\@NumberstringMUSenglish\@NumberstringMenglish
3785 \global\let\@NumberstringFUSenglish\@NumberstringMenglish
3786 \global\let\@NumberstringNUSenglish\@NumberstringMenglish
3787 \global\let\@ordinalstringMUSenglish\@ordinalstringMenglish
3788 \global\let\@ordinalstringFUSenglish\@ordinalstringMenglish
3789 \global\let\@ordinalstringNUSenglish\@ordinalstringMenglish
3790 \global\let\@OrdinalstringMUSenglish\@OrdinalstringMenglish
3791 \global\let\@OrdinalstringFUSenglish\@OrdinalstringMenglish
3792 \global\let\@OrdinalstringNUSenglish\@OrdinalstringMenglish
```

## 9.2 fcnumparser.sty

```
3793 \NeedsTeXFormat{LaTeX2e}
```

```
3794 \ProvidesPackage{fcnumparser}[2017/06/15]
```

\fc@counter@parser is just a shorthand to parse a number held in a counter.

```
3795 \def\fc@counter@parser#1{%
3796   \expandafter\fc@number@parser\expandafter{\the#1.}%
3797 }
3798 \newcount\fc@digit@counter
3799
3800 \def\fc@end@{\fc@end}
```

number@analysis First of all we need to separate the number between integer and fractional part. Number to be analysed is in '#1'. Decimal separator may be . or , whichever first. At end of this macro, integer part goes to \fc@integer@part and fractional part goes to \fc@fractional@part.

```
3801 \def\fc@number@analysis#1\fc@nil{%
```

First check for the presence of a decimal point in the number.

```
3802   \def\@tempb##1.##2\fc@nil{\def\fc@integer@part{##1}\def\@tempa{##2}}%
3803   \@tempb#1.\fc@end\fc@nil
3804   \ifx\@tempa\fc@end@
```

Here \@tempa is \ifx-equal to \fc@end, which means that the number does not contain any decimal point. So we do the same trick to search for a comma.

```
3805     \def\@tempb##1,##2\fc@nil{\def\fc@integer@part{##1}\def\@tempa{##2}}%
3806     \@tempb#1,\fc@end\fc@nil
3807     \ifx\@tempa\fc@end@
```

No comma either, so fractional part is set empty.

```
3808       \def\fc@fractional@part{}%
3809       \else
```

Comma has been found, so we just need to drop ', \fc@end' from the end of \@tempa to get the fractional part.

```
3810         \def\@tempb##1,\fc@end{\def\fc@fractional@part{##1}}%
3811         \expandafter\@tempb\@tempa
3812     \fi
3813 \else
```

Decimal point has been found, so we just need to drop '. \fc@end' from the end \@tempa to get the fractional part.

```
3814         \def\@tempb##1.\fc@end{\def\fc@fractional@part{##1}}%
3815         \expandafter\@tempb\@tempa
3816     \fi
3817 }
```

number@parser Macro \fc@number@parser is the main engine to parse a number. Argument '#1' is input and contains the number to be parsed. At end of this macro, each digit is stored separately in a \fc@digit@<n>, and macros \fc@min@weight and \fc@max@weight are set to the bounds for <n>.

```
3818 \def\fc@number@parser#1{%
```

First remove all the spaces in #1, and place the result into \@tempa.

```
3819   \let\@tempa@empty
3820   \def\@tempb##1##2\fc@nil{%
3821     \def\@tempc{##1}%
3822     \ifx\@tempc\space
```

```

3823 \else
3824 \expandafter\def\expandafter\@tempa\expandafter{\@tempa ##1}%
3825 \fi
3826 \def\@tempc{##2}%
3827 \ifx\@tempc\@empty
3828 \expandafter\@gobble
3829 \else
3830 \expandafter\@tempb
3831 \fi
3832 ##2\fc@nil
3833 }%
3834 \@tempb#1\fc@nil

  Get the sign into \fc@sign and the unsigned number part into \fc@number.
3835 \def\@tempb##1##2\fc@nil{\def\fc@sign{##1}\def\fc@number{##2}}%
3836 \expandafter\@tempb\@tempa\fc@nil
3837 \expandafter\if\fc@sign+%
3838 \def\fc@sign@case{1}%
3839 \else
3840 \expandafter\if\fc@sign-%
3841 \def\fc@sign@case{2}%
3842 \else
3843 \def\fc@sign{}%
3844 \def\fc@sign@case{0}%
3845 \let\fc@number\@tempa
3846 \fi
3847 \fi
3848 \ifx\fc@number\@empty
3849 \PackageError{fcnumparser}{Invalid number}{Number must contain at least one non blank
3850 character after sign}%
3851 \fi

  Now, split \fc@number into \fc@integer@part and \fc@fractional@part.
3852 \expandafter\fc@number@analysis\fc@number\fc@nil

  Now, split \fc@integer@part into a sequence of \fc@digit@<n> with <n> ranging from
  \fc@unit@weight to \fc@max@weight. We will use macro \fc@parse@integer@digits
  for that, but that will place the digits into \fc@digit@<n> with <n> ranging from 2 ×
  \fc@unit@weight – \fc@max@weight upto \fc@unit@weight – 1.
3853 \expandafter\fc@digit@counter\fc@unit@weight
3854 \expandafter\fc@parse@integer@digits\fc@integer@part\fc@end\fc@nil

  First we compute the weight of the most significant digit: after \fc@parse@integer@digits,
  \fc@digit@counter is equal to \fc@unit@weight – mw – 1 and we want to set \fc@max@weight
  to \fc@unit@weight + mw so we do:

      \fc@max@weight ← (–\fc@digit@counter) + 2 × \fc@unit@weight – 1
3855 \fc@digit@counter – \fc@digit@counter
3856 \advance\fc@digit@counter by \fc@unit@weight
3857 \advance\fc@digit@counter by \fc@unit@weight
3858 \advance\fc@digit@counter by -1 %

```

```
3859 \edef\fc@max@weight{\the\fc@digit@counter}%
Now we loop for  $i = \text{\fc@unit@weight}$  to  $\text{\fc@max@weight}$  in order to copy all the digits
from  $\text{\fc@digit@}(i + \text{offset})$  to  $\text{\fc@digit@}(i)$ . First we compute offset into  $\text{\@temp i}$ .
```

```
3860 {%
3861 \count0 \fc@unit@weight\relax
3862 \count1 \fc@max@weight\relax
3863 \advance\count0 by -\count1 %
3864 \advance\count0 by -1 %
3865 \def\@tempa#1{\def\@tempb{\def\@temp i{##1}}}%
3866 \expandafter\@tempa\expandafter{\the\count0}%
3867 \expandafter
3868 }\@tempb
```

Now we loop to copy the digits. To do that we define a macro  $\text{\@temp l}$  for terminal recursion.

```
3869 \expandafter\fc@digit@counter\fc@unit@weight
3870 \def\@temp l{%
3871 \ifnum\fc@digit@counter>\fc@max@weight
3872 \let\next\relax
3873 \else
```

Here is the loop body:

```
3874 {%
3875 \count0 \@temp i
3876 \advance\count0 by \fc@digit@counter
3877 \expandafter\def\expandafter\@temp d\expandafter{\csname fc@digit@the\count0\endcsname
3878 \expandafter\def\expandafter\@temp e\expandafter{\csname fc@digit@the\fc@digit@counte
3879 \def\@tempa###1###2{\def\@tempb{\let###1###2}}}%
3880 \expandafter\expandafter\expandafter\@tempa\expandafter\@temp e\@temp d
3881 \expandafter
3882 }\@tempb
3883 \advance\fc@digit@counter by 1 %
3884 \fi
3885 \next
3886 }%
3887 \let\next\@temp l
3888 \@temp l
```

Split  $\text{\fc@fractional@part}$  into a sequence of  $\text{\fc@digit@}(n)$  with  $\langle n \rangle$  ranging from  $\text{\fc@unit@weight} - 1$  to  $\text{\fc@min@weight}$  by step of  $-1$ . This is much more simpler because we get the digits with the final range of index, so no post-processing loop is needed.

```
3889 \expandafter\fc@digit@counter\fc@unit@weight
3890 \expandafter\fc@parse@integer@digits\fc@fractional@part\fc@end\fc@nil
3891 \edef\fc@min@weight{\the\fc@digit@counter}%
3892 }
```

Macro  $\text{\fc@parse@integer@digits}$  is used to

```
3893 \ifcsundef\fc@parse@integer@digits\{}{%
3894 \PackageError{fcnumparser}{Duplicate definition}{Redefinition of
3895 macro 'fc@parse@integer@digits'}}
3896 \def\fc@parse@integer@digits#1#2\fc@nil{%
3897 \def\@tempa{#1}%
```

```

3898 \ifx\@tempa\fc@end@
3899 \def\next##1\fc@nil{}%
3900 \else
3901 \let\next\fc@parse@integer@digits
3902 \advance\fc@digit@counter by -1
3903 \expandafter\def\csname fc@digit@\the\fc@digit@counter\endcsname{#1}%
3904 \fi
3905 \next#2\fc@nil
3906 }
3907
3908
3909 \newcommand*\fc@unit@weight{0}
3910

```

Now we have macros to read a few digits from the `\fc@digit@<n>` array and form a corresponding number.

`\fc@read@unit` `\fc@read@unit` just reads one digit and form an integer in the range [0..9]. First we check that the macro is not yet defined.

```

3911 \ifcsundef\fc@read@unit\fi{%
3912 \PackageError{fcnumparser}{Duplicate definition}{Redefinition of macro 'fc@read@unit'}}

```

Arguments as follows:

- #1 output counter: into which the read value is placed
  - #2 input number: unit weight at which reach the value is to be read #2
- does not need to be comprised between `\fc@min@weight` and `fc@min@weight`, if outside this interval, then a zero is read.

```

3913 \def\fc@read@unit#1#2{%
3914 \ifnum#2>\fc@max@weight
3915 #1=0\relax
3916 \else
3917 \ifnum#2<\fc@min@weight
3918 #1=0\relax
3919 \else
3920 {%
3921 \edef\@tempa{\number#2}%
3922 \count0=\@tempa
3923 \edef\@tempa{\csname fc@digit@\the\count0\endcsname}%
3924 \def\@tempb##1{\def\@tempa{#1=##1\relax}}%
3925 \expandafter\@tempb\expandafter{\@tempa}%
3926 \expandafter
3927 }\@tempa
3928 \fi
3929 \fi
3930 }

```

`\fc@read@hundred` Macro `\fc@read@hundred` is used to read a pair of digits and form an integer in the range [0..99]. First we check that the macro is not yet defined.

```

3931 \ifcsundef\fc@read@hundred\fi{%
3932 \PackageError{fcnumparser}{Duplicate definition}{Redefinition of macro 'fc@read@hundred'}}

```

Arguments as follows — same interface as `\fc@read@unit`:

#1 output counter: into which the read value is placed  
#2 input number: unit weight at which reach the value is to be read

```

3933 \def\fc@read@hundred#1#2{%
3934   {%
3935     \fc@read@unit{\count0}{#2}%
3936     \def\@tempa##1{\fc@read@unit{\count1}{##1}}%
3937     \count2=#2%
3938     \advance\count2 by 1 %
3939     \expandafter\@tempa{\the\count2}%
3940     \multiply\count1 by 10 %
3941     \advance\count1 by \count0 %
3942     \def\@tempa##1{\def\@tempb{#1=##1\relax}}
3943     \expandafter\@tempa\expandafter{\the\count1}%
3944     \expandafter
3945   }\@tempb
3946 }

```

`\fc@read@thousand` Macro `\fc@read@thousand` is used to read a trio of digits and form an integer in the range [0..999]. First we check that the macro is not yet defined.

```

3947 \ifcsundef{fc@read@thousand}{}{%
3948   \PackageError{fncparser}{Duplicate definition}{Redefinition of macro
3949     'fc@read@thousand'}}

```

Arguments as follows — same interface as `\fc@read@unit`:

#1 output counter: into which the read value is placed  
#2 input number: unit weight at which reach the value is to be read

```

3950 \def\fc@read@thousand#1#2{%
3951   {%
3952     \fc@read@unit{\count0}{#2}%
3953     \def\@tempa##1{\fc@read@hundred{\count1}{##1}}%
3954     \count2=#2%
3955     \advance\count2 by 1 %
3956     \expandafter\@tempa{\the\count2}%
3957     \multiply\count1 by 10 %
3958     \advance\count1 by \count0 %
3959     \def\@tempa##1{\def\@tempb{#1=##1\relax}}
3960     \expandafter\@tempa\expandafter{\the\count1}%
3961     \expandafter
3962   }\@tempb
3963 }

```

`\fc@read@thousand` Note: one myriad is ten thousand. Macro `\fc@read@myriad` is used to read a quatuor of digits and form an integer in the range [0..9999]. First we check that the macro is not yet defined.

```

3964 \ifcsundef{fc@read@myriad}{}{%
3965   \PackageError{fncparser}{Duplicate definition}{Redefinition of macro
3966     'fc@read@myriad'}}

```

Arguments as follows — same interface as `\fc@read@unit`:

#1 output counter: into which the read value is placed  
#2 input number: unit weight at which reach the value is to be read

```

3967 \def\fc@read@myriad#1#2{%
3968   {%
3969     \fc@read@hundred{\count0}{#2}%
3970     \def\@tempa##1{\fc@read@hundred{\count1}{##1}}%
3971     \count2=#2
3972     \advance\count2 by 2
3973     \expandafter\@tempa{\the\count2}%
3974     \multiply\count1 by 100 %
3975     \advance\count1 by \count0 %
3976     \def\@tempa##1{\def\@tempb{#1=##1\relax}}%
3977     \expandafter\@tempa\expandafter{\the\count1}%
3978     \expandafter
3979   }\@tempb
3980 }

```

`\fc@check@nonzeros` Macro `\fc@check@nonzeros` is used to check whether the number represented by digits `\fc@digit@ $n$` , with  $n$  in some interval, is zero, one, or more than one. First we check that the macro is not yet defined.

```

3981 \ifcsundef{fc@check@nonzeros}{}{%
3982   \PackageError{fcnumparser}{Duplicate definition}{Redefinition of macro
3983     'fc@check@nonzeros'}}

```

Arguments as follows:

- #1 input number: minimum unit weight at which start to search the non-zeros
- #2 input number: maximum unit weight at which end to search the non-zeros
- #3 output macro: let  $n$  be the number represented by digits the weight of which span from #1 to #2, then #3 is set to the number  $\min(n,9)$ .

Actually `\fc@check@nonzeros` is just a wrapper to collect arguments, and the real job is delegated to `\fc@@check@nonzeros@inner` which is called inside a group.

```

3984 \def\fc@check@nonzeros#1#2#3{%
3985   {%

```

So first we save inputs into local macros used by `\fc@@check@nonzeros@inner` as input arguments

```

3986     \edef\@tempa{\number#1}%
3987     \edef\@tempb{\number#2}%
3988     \count0=\@tempa
3989     \count1=\@tempb\relax

```

Then we do the real job

```

3990     \fc@@check@nonzeros@inner

```

And finally, we propagate the output after end of group — i.e. closing brace.

```

3991     \def\@tempd##1{\def\@tempa{\def#3{##1}}}%
3992     \expandafter\@tempd\expandafter{\@tempc}%
3993     \expandafter
3994   }\@tempa
3995 }

```

`\fc@@check@nonzeros@inner` Macro `\fc@@check@nonzeros@inner` Check whether some part of the parsed value contains some non-zero digit At the call of this macro we expect that:

```

\@tempa input/output macro:
      input minimum unit weight at which start to search the non-zeros
      output macro may have been redefined
\@tempb input/output macro:
      input maximum unit weight at which end to search the non-zeros
      output macro may have been redefined
\@tempc output macro: 0 if all-zeros, 1 if at least one zero is found
\count0 output counter: weight + 1 of the first found non zero starting from minimum
weight.
3996 \def\fc@@check@nonzeros@inner{%
3997   \ifnum\count0<\fc@min@weight
3998     \count0=\fc@min@weight\relax
3999   \fi
4000   \ifnum\count1>\fc@max@weight\relax
4001     \count1=\fc@max@weight
4002   \fi
4003   \count2\count0 %
4004   \advance\count2 by 1 %
4005   \ifnum\count0>\count1 %
4006     \PackageError{fcnumparser}{Unexpected arguments}{Number in argument 2 of macro
4007       'fc@check@nonzeros' must be at least equal to number in argument 1}%
4008   \else
4009     \fc@@check@nonzeros@inner@loopbody
4010     \ifnum\@tempc>0 %
4011       \ifnum\@tempc<9 %
4012         \ifnum\count0>\count1 %
4013           \else
4014             \let\@tempd\@tempc
4015             \fc@@check@nonzeros@inner@loopbody
4016             \ifnum\@tempc=0 %
4017               \let\@tempc\@tempd
4018             \else
4019               \def\@tempc{9}%
4020             \fi
4021           \fi
4022         \fi
4023       \fi
4024     \fi
4025 }
4026 \def\fc@@check@nonzeros@inner@loopbody{%
4027   % \@tempc <- digit of weight \count0
4028   \expandafter\let\expandafter\@tempc\csname fc@digit@the\count0\endcsname
4029   \advance\count0 by 1 %
4030   \ifnum\@tempc=0 %
4031     \ifnum\count0>\count1 %
4032       \let\next\relax
4033     \else
4034       \let\next\fc@@check@nonzeros@inner@loopbody

```

```

4035     \fi
4036   \else
4037     \ifnum\count0>\count2 %
4038       \def\@tempc{9}%
4039     \fi
4040     \let\next\relax
4041   \fi
4042   \next
4043 }

```

`\intpart@find@last` Macro `\fc@intpart@find@last` find the rightmost non zero digit in the integer part. First check that the macro is not yet defined.

```

4044 \ifcsundef{fc@intpart@find@last}{-}{%
4045   \PackageError{fcnumparser}{Duplicate definition}{Redefinition of macro
4046     'fc@intpart@find@last'}}

```

When macro is called, the number of interest is already parsed, that is to say each digit of weight  $w$  is stored in macro `\fc@digit@<math>w</math>`. Macro `\fc@intpart@find@last` takes one single argument which is a counter to set to the result.

```

4047 \def\fc@intpart@find@last#1{%
4048   {%

```

Counter `\count0` will hold the result. So we will loop on `\count0`, starting from  $\min\{u, w_{\min}\}$ , where  $u \triangleq \text{\fc@unit@weight}$ , and  $w_{\min} \triangleq \text{\fc@min@weight}$ . So first set `\count0` to  $\min\{u, w_{\min}\}$ :

```

4049   \count0=\fc@unit@weight\space
4050   \ifnum\count0<\fc@min@weight\space
4051     \count0=\fc@min@weight\space
4052   \fi

```

Now the loop. This is done by defining macro `\@templ` for final recursion.

```

4053   \def\@templ{%
4054     \ifnum\csname fc@digit@the\count0\endcsname=0 %
4055       \advance\count0 by 1 %
4056       \ifnum\count0>\fc@max@weight\space
4057         \let\next\relax
4058       \fi
4059     \else
4060       \let\next\relax
4061     \fi
4062     \next
4063   }%
4064   \let\next\@templ
4065   \@templ

```

Now propagate result after closing bracket into counter #1.

```

4066     \toks0{#1}%
4067     \edef\@tempa{\the\toks0=\the\count0}%
4068     \expandafter
4069   }\@tempa\space
4070 }

```

```

c@get@last@word Getting last word. Arguments as follows:
#1 input: full sequence
#2 output macro 1: all sequence without last word
#3 output macro 2: last word
4071 \ifcsundef{fc@get@last@word}{\PackageError{fcnumparser}{Duplicate definition}{Redefinition
4072 of macro 'fc@get@last@word'}}}%
4073 \def\fc@get@last@word#1#2#3{%
4074   {%

First we split #1 into two parts: everything that is upto \fc@wcase exclusive goes to \toks0,
and evrything from \fc@wcase exclusive upto the final \@nil exclusive goes to \toks1.
4075   \def\@tempa##1\fc@wcase##2\@nil\fc@end{%
4076     \toks0{##1}%

Actually a dummy \fc@wcase is appended to \toks1, because that makes easier further
checking that it does not contains any other \fc@wcase.
4077     \toks1{##2\fc@wcase}%
4078   }%
4079   \@tempa#1\fc@end

Now leading part upto last word should be in \toks0, and last word should be in \toks1.
However we need to check that this is really the last word, i.e. we need to check that there
is no \fc@wcase inside \toks1 other than the tailing dummy one. To that purpose we will
loop while we find that \toks1 contains some \fc@wcase. First we define \@tempa to split
\the\toks1 between parts before and after some potential \fc@wcase.
4080   \def\@tempa##1\fc@wcase##2\fc@end{%
4081     \toks2{##1}%
4082     \def\@tempb{##2}%
4083     \toks3{##2}%
4084   }%

\@tempt is just an aliases of \toks0 to make its handling easier later on.
4085   \toksdef\@tempt0 %

Now the loop itself, this is done by terminal recursion with macro \@templ.
4086   \def\@templ{%
4087     \expandafter\@tempa\the\toks1 \fc@end
4088     \ifx\@tempb\@empty

\@tempb empty means that the only \fc@wcase found in \the\toks1 is the dummy one. So
we end the loop here, \toks2 contains the last word.
4089     \let\next\relax
4090     \else

\@tempb is not empty, first we use
4091     \expandafter\expandafter\expandafter\@tempt
4092     \expandafter\expandafter\expandafter{%
4093     \expandafter\the\expandafter\@tempt
4094     \expandafter\fc@wcase\the\toks2}%
4095     \toks1\toks3 %
4096     \fi
4097     \next
4098   }%

```

```

4099   \let\next\@templ
4100   \@templ
4101   \edef\@tempa{\def\noexpand#2{\the\toks0}\def\noexpand#3{\the\toks2}}%
4102   \expandafter
4103   }\@tempa
4104 }

```

`fc@get@last@word` Getting last letter. Arguments as follows:

- #1 input: full word
- #2 output macro 1: all word without last letter
- #3 output macro 2: last letter

```

4105 \ifcsundef{fc@get@last@letter}{\PackageError{fcnumparser}{Duplicate definition}{Redefinition
4106   of macro 'fc@get@last@letter'}}%
4107 \def\fc@get@last@letter#1#2#3{%
4108   {%

```

First copy input to local `\toks1`. What we are going to do is to bubble one by one letters from `\toks1` which initially contains the whole word, into `\toks0`. At the end of the macro `\toks0` will therefore contain the whole word but the last letter, and the last letter will be in `\toks1`.

```

4109   \toks1{#1}%
4110   \toks0{%
4111   \toksdef\@tempt0 %

```

We define `\@tempa` in order to pop the first letter from the remaining of word.

```

4112   \def\@tempa##1##2\fc@nil{%
4113     \toks2{##1}%
4114     \toks3{##2}%
4115     \def\@tempb{##2}%
4116   }%

```

Now we define `\@templ` to do the loop by terminal recursion.

```

4117   \def\@templ{%
4118     \expandafter\@tempa\the\toks1 \fc@nil
4119     \ifx\@tempb\@empty

```

Stop loop, as `\toks1` has been detected to be one single letter.

```

4120     \let\next\relax
4121     \else

```

Here we append to `\toks0` the content of `\toks2`, i.e. the next letter.

```

4122     \expandafter\expandafter\expandafter\@tempt
4123     \expandafter\expandafter\expandafter{%
4124     \expandafter\the\expandafter\@tempt
4125     \the\toks2}%

```

And the remaining letters go to `\toks1` for the next iteration.

```

4126     \toks1\toks3 %
4127     \fi
4128     \next
4129   }%

```

Here run the loop.

```

4130   \let\next\@templ
4131   \next

```

Now propagate the results into macros #2 and #3 after closing brace.

```
4132 \edef\@tempa{\def\noexpand#2{\the\toks0}\def\noexpand#3{\the\toks1}}%
4133 \expandafter
4134 }\@tempa
4135 }%
```

### 9.3 fcprefix.sty

Pseudo-latin prefixes.

```
4136 \NeedsTeXFormat{LaTeX2e}
4137 \ProvidesPackage{fcprefix}[2012/09/28]
4138 \RequirePackage{ifthen}
4139 \RequirePackage{keyval}
4140 \RequirePackage{fcnumparser}
```

Option ‘use duode and unde’ is to select whether 18 and suchlikes ( $\langle x \rangle 8$ ,  $\langle x \rangle 9$ ) writes like duodevicies, or like octodecies. For French it should be ‘below 20’. Possible values are ‘below 20’ and ‘never’.

```
4141 \define@key{fcprefix}{use duode and unde}[below20]{%
4142 \ifthenelse{\equal{#1}{below20}}{%
4143 \def\fc@duodeandunde{2}%
4144 }{%
4145 \ifthenelse{\equal{#1}{never}}{%
4146 \def\fc@duodeandunde{0}%
4147 }{%
4148 \PackageError{fcprefix}{Unexpected option}{%
4149 Option ‘use duode and unde’ expects ‘below 20’ or ‘never’ }%
4150 }%
4151 }%
4152 }
```

Default is ‘below 20’ like in French.

```
4153 \def\fc@duodeandunde{2}
```

Option ‘numeral u in duo’, this can be ‘true’ or ‘false’ and is used to select whether 12 and suchlikes write like dodec $\langle xxx \rangle$  or duodec $\langle xxx \rangle$  for numerals.

```
4154 \define@key{fcprefix}{numeral u in duo}[false]{%
4155 \ifthenelse{\equal{#1}{false}}{%
4156 \let\fc@u@in@duo\@empty
4157 }{%
4158 \ifthenelse{\equal{#1}{true}}{%
4159 \def\fc@u@in@duo{u}%
4160 }{%
4161 \PackageError{fcprefix}{Unexpected option}{%
4162 Option ‘numeral u in duo’ expects ‘true’ or ‘false’ }%
4163 }%
4164 }%
4165 }
```

Option ‘e accute’, this can be ‘true’ or ‘false’ and is used to select whether letter ‘e’ has an accute accent when it pronounce [e] in French.

```

4166 \define@key{fcprefix}{e accute}[false]{%
4167   \ifthenelse{\equal{#1}{false}}{%
4168     \let\fc@prefix@eaccute\@firstofone
4169   }{%
4170     \ifthenelse{\equal{#1}{true}}{%
4171       \let\fc@prefix@eaccute\'%
4172     }{%
4173       \PackageError{fcprefix}{Unexpected option}{%
4174         Option 'e accute' expects 'true' or 'false' }%
4175     }%
4176   }%
4177 }

```

Default is to set accute accent like in French.

```
4178 \let\fc@prefix@eaccute\'%
```

Option 'power of millia' tells how millia is raise to power n. It expects value:  
 recursive for which millia squared is noted as 'milliamillia'

arabic for which millia squared is noted as 'millia^2'

prefix for which millia squared is noted as 'bismillia'

```

4179 \define@key{fcprefix}{power of millia}[prefix]{%
4180   \ifthenelse{\equal{#1}{prefix}}{%
4181     \let\fc@power@of@millia@init\@gobbletwo
4182     \let\fc@power@of@millia\fc@prefix@millia
4183   }{%
4184     \ifthenelse{\equal{#1}{arabic}}{%
4185       \let\fc@power@of@millia@init\@gobbletwo
4186       \let\fc@power@of@millia\fc@@arabic@millia
4187     }{%
4188       \ifthenelse{\equal{#1}{recursive}}{%
4189         \let\fc@power@of@millia@init\fc@@recurse@millia@init
4190         \let\fc@power@of@millia\fc@@recurse@millia
4191       }{%
4192         \PackageError{fcprefix}{Unexpected option}{%
4193           Option 'power of millia' expects 'recursive', 'arabic', or 'prefix' }%
4194       }%
4195     }%
4196   }%
4197 }

```

Arguments as follows:

#1 output macro

#2 number with current weight  $w$

```

4198 \def\fc@@recurse@millia#1#2{%
4199   \let\@temp#1%
4200   \edef#1{millia\@temp}%
4201 }

```

Arguments as follows — same interface as \fc@@recurse@millia:

```

#1 output macro
#2 number with current weight  $w$ 
4202 \def\fc@@recurse@millia@init#1#2{%
4203   {%
      Save input argument current weight  $w$  into local macro \@tempb.
4204   \edef\@tempb{\number#2}%
      Now main loop from 0 to  $w$ . Final value of \@tempa will be the result.
4205   \count0=0 %
4206   \let\@tempa\@empty
4207   \loop
4208     \ifnum\count0<\@tempb
4209       \advance\count0 by 1 %
4210       \expandafter\def
4211       \expandafter\@tempa\expandafter{\@tempa millia}%
4212     \repeat
      Now propagate the expansion of \@tempa into #1 after closing brace.
4213     \edef\@tempb{\def\noexpand#1{\@tempa}}%
4214     \expandafter
4215   }\@tempb
4216 }

```

Arguments as follows — same interface as \fc@@recurse@millia:

```

#1 output macro
#2 number with current weight  $w$ 
4217 \def\fc@@arabic@millia#1#2{%
4218   \ifnum#2=0 %
4219     \let#1\@empty
4220   \else
4221     \edef#1{millia\^{\the#2}}%
4222   \fi
4223 }

```

Arguments as follows — same interface as \fc@@recurse@millia:

```

#1 output macro
#2 number with current weight  $w$ 
4224 \def\fc@@prefix@millia#1#2{%
4225   \fc@@latin@numeral@pefix{#2}{#1}%
4226 }

```

Default value of option ‘power of millia’ is ‘prefix’:

```

4227 \let\fc@power@of@millia@init\@gobbletwo
4228 \let\fc@power@of@millia\fc@@prefix@millia

```

Compute a cardinal prefix for  $n$ -illion, like  $1 \Rightarrow$  ‘m’,  $2 \Rightarrow$  ‘bi’,  $3 \Rightarrow$  ‘tri’. The algorithm to derive this prefix is that of Russ Rowlett I founds its documentation on Alain Lassine’s site: [http://www.alain.be/Boece/grands\\_nombres.html](http://www.alain.be/Boece/grands_nombres.html). First check that macro is not yet defined.

```

4229 \ifcsundef\fc@@latin@cardinal@pefix{}{%
4230   \PackageError{fmtcount}{Duplicate definition}{Redefinition of macro ‘fc@@latin@cardinal@pefix

```

Arguments as follows:

#1 input number to be formatted

#2 outut macro name into which to place the formatted result

```
4231 \def\fc@@latin@cardinal@pefix#1#2{%
```

```
4232  {%
```

First we put input argument into local macro @cs@tempa with full expansion.

```
4233  \edef\@tempa{\number#1}%
```

Now parse number from expanded input.

```
4234  \expandafter\fc@number@parser\expandafter{\@tempa}%
```

```
4235  \count2=0 %
```

\@tempt will hold the optional final t, \@tempu is used to initialize \@tempt to 't' when the first non-zero 3digit group is met, which is the job made by \@tempu.

```
4236  \let\@tempt\@empty
```

```
4237  \def\@tempu{t}%
```

\@tempm will hold the millia<sup>n+3</sup>

```
4238  \let\@tempm\@empty
```

Loop by means of terminal recursion of herinafter defined macro \@templ. We loop by group of 3 digits.

```
4239  \def\@templ{%
```

```
4240    \ifnum\count2>\fc@max@weight
```

```
4241      \let\next\relax
```

```
4242    \else
```

Loop body. Here we read a group of 3 consecutive digits  $d_2d_1d_0$  and place them respectively into \count3, \count4, and \count5.

```
4243      \fc@read@unit{\count3}{\count2}%
```

```
4244      \advance\count2 by 1 %
```

```
4245      \fc@read@unit{\count4}{\count2}%
```

```
4246      \advance\count2 by 1 %
```

```
4247      \fc@read@unit{\count5}{\count2}%
```

```
4248      \advance\count2 by 1 %
```

If the 3 considered digits  $d_2d_1d_0$  are not all zero, then set \@tempt to 't' for the first time this event is met.

```
4249      \edef\@tempn{%
```

```
4250        \ifnum\count3=0\else 1\fi
```

```
4251        \ifnum\count4=0\else 1\fi
```

```
4252        \ifnum\count5=0\else 1\fi
```

```
4253      }%
```

```
4254      \ifx\@tempn\@empty\else
```

```
4255        \let\@tempt\@tempu
```

```
4256        \let\@tempu\@empty
```

```
4257      \fi
```

Now process the current group  $d_2d_1d_0$  of 3 digits.

```
4258      \let\@temp\@tempa
```

```
4259      \edef\@tempa{%
```

Here we process  $d_2$  held by `\count5`, that is to say hundreds.

```

4260         \ifcase\count5 %
4261         \or cen%
4262         \or ducen%
4263         \or trecen%
4264         \or quadringen%
4265         \or quingen%
4266         \or sescen%
4267         \or septigen%
4268         \or octingen%
4269         \or nongen%
4270         \fi

```

Here we process  $d_1d_0$  held by `\count4` & `\count3`, that is to say tens and units.

```

4271         \ifnum\count4=0 %
4272         % x0(0..9)
4273         \ifnum\count2=3 %
4274         % Absolute weight zero
4275         \ifcase\count3 \@tempt
4276         \or m%
4277         \or b%
4278         \or tr%
4279         \or quadr%
4280         \or quin\@tempt
4281         \or sex\@tempt
4282         \or sep\@tempt
4283         \or oc\@tempt
4284         \or non%
4285         \fi
4286         \else

```

Here the weight of `\count3` is  $3 \times n$ , with  $n > 0$ , i.e. this is followed by a millia<sup>n</sup>.

```

4287         \ifcase\count3 %
4288         \or \ifnum\count2>\fc@max@weight\else un\fi
4289         \or d\fc@u@in@duo o%
4290         \or tre%
4291         \or quattuor%
4292         \or quin%
4293         \or sex%
4294         \or septen%
4295         \or octo%
4296         \or novem%
4297         \fi
4298         \fi
4299         \else
4300         % x(10..99)
4301         \ifcase\count3 %
4302         \or un%
4303         \or d\fc@u@in@duo o%
4304         \or tre%

```

```

4305         \or quattuor%
4306         \or quin%
4307         \or sex%
4308         \or septen%
4309         \or octo%
4310         \or novem%
4311         \fi
4312         \ifcase\count4 %
4313         \or dec%
4314         \or vigin\@tempt
4315         \or trigin\@tempt
4316         \or quadragin\@tempt
4317         \or quinquagin\@tempt
4318         \or sexagin\@tempt
4319         \or septuagin\@tempt
4320         \or octogin\@tempt
4321         \or nonagin\@tempt
4322         \fi
4323     \fi
    Insert the millia(n+3) only if  $d_2 d_1 d_0 \neq 0$ , i.e. if one of \count3 \count4 or \count5 is non
    zero.
4324     \@tempm
    And append previous version of \@tempa.
4325     \@temp
4326     }%
    “Concatenate” millia to \@tempm, so that \@tempm will expand to millia(n+3)+1 at the next
    iteration. Actually whether this is a concatenation or some millia prefixing depends of op-
    tion ‘power of millia’.
4327     \fc@power@of@millia\@tempm{\count2}%
4328     \fi
4329     \next
4330     }%
4331     \let\@tempa\@empty
4332     \let\@next\@templ
4333     \@templ
    Propagate expansion of \@tempa into #2 after closing bracket.
4334     \def\@tempb##1{\def\@tempa{\def#2{##1}}}%
4335     \expandafter\@tempb\expandafter{\@tempa}%
4336     \expandafter
4337     }\@tempa
4338 }

```

`@latin@numeral@pe` Compute a numeral prefix like ‘sémel’, ‘bis’, ‘ter’, ‘quater’, etc... I found the algorithm to derive
 this prefix on Alain Lassine’s site: [http://www.alain.be/Boece/nombres\\_gargantuesques.html](http://www.alain.be/Boece/nombres_gargantuesques.html).
 First check that the macro is not yet defined.

```

4339 \ifcsundef{fc@latin@numeral@pefix}{}%
4340 \PackageError{fmtcount}{Duplicate definition}{Redefinition of macro

```

```
4341   'fc@@latin@numeral@pefix'}}}
```

Arguments as follows:

- #1 input number to be formatted,
- #2 output macro name into which to place the result

```
4342 \def\fc@@latin@numeral@pefix#1#2{%
4343   {%
4344     \edef\@tempa{\number#1}%
4345     \def\fc@unit@weight{0}%
4346     \expandafter\fc@number@parser\expandafter{\@tempa}%
4347     \count2=0 %
```

Macro \@tempm will hold the millies  $^{n+3}$ .

```
4348   \let\@tempm\@empty
```

Loop over digits. This is done by defining macro \@templ for terminal recursion.

```
4349   \def\@templ{%
4350     \ifnum\count2>\fc@max@weight
4351       \let\next\relax
4352     \else
```

Loop body. Three consecutive digits  $d_2d_1d_0$  are read into counters \count3, \count4, and \count5.

```
4353       \fc@read@unit{\count3}{\count2}%
4354       \advance\count2 by 1 %
4355       \fc@read@unit{\count4}{\count2}%
4356       \advance\count2 by 1 %
4357       \fc@read@unit{\count5}{\count2}%
4358       \advance\count2 by 1 %
```

Check the use of duodevicies instead of octodecies.

```
4359       \let\@tempn\@secondoftwo
4360       \ifnum\count3>7 %
4361         \ifnum\count4<\fc@duodeandunde
4362           \ifnum\count4>0 %
4363             \let\@tempn\@firstoftwo
4364           \fi
4365         \fi
4366       \fi
4367       \@tempn
4368       {% use duodevicies for eighteen
4369         \advance\count4 by 1 %
4370         \let\@temps\@secondoftwo
4371       }{% do not use duodevicies for eighteen
4372         \let\@temps\@firstoftwo
4373       }%
4374       \let\@temppp\@tempa
4375       \edef\@tempa{%
4376         % hundreds
4377         \ifcase\count5 %
4378         \expandafter\@gobble
```

```

4379         \or c%
4380         \or duc%
4381         \or trec%
4382         \or quadring%
4383         \or quing%
4384         \or sesc%
4385         \or septing%
4386         \or octing%
4387         \or nong%
4388         \fi
4389         {enties}%
4390         \ifnum\count4=0 %

```

Here  $d_2d_1d_0$  is such that  $d_1 = 0$ .

```

4391         \ifcase\count3 %
4392         \or
4393         \ifnum\count2=3 %
4394             s\fc@prefix@eacute emel%
4395         \else
4396             \ifnum\count2>\fc@max@weight\else un\fi
4397         \fi
4398         \or bis%
4399         \or ter%
4400         \or quater%
4401         \or quinquies%
4402         \or sexies%
4403         \or septies%
4404         \or octies%
4405         \or novies%
4406         \fi
4407         \else

```

Here  $d_2d_1d_0$  is such that  $d_1 \geq 1$ .

```

4408         \ifcase\count3 %
4409         \or un%
4410         \or d\fc@u@in@duo o%
4411         \or ter%
4412         \or quater%
4413         \or quin%
4414         \or sex%
4415         \or septen%
4416         \or \@temps{octo}{duod\fc@prefix@eacute e}% x8 = two before next (x+1)0
4417         \or \@temps{novem}{und\fc@prefix@eacute e}% x9 = one before next (x+1)0
4418         \fi
4419         \ifcase\count4 %
4420         % can't get here
4421         \or d\fc@prefix@eacute ec%
4422         \or vic%
4423         \or tric%
4424         \or quadrag%

```

```

4425         \or quinquag%
4426         \or sexag%
4427         \or septuag%
4428         \or octog%
4429         \or nonag%
4430         \fi
4431         ies%
4432     \fi
4433     % Insert the millies^(n/3) only if one of \count3 \count4 \count5 is non zero
4434     \@tempm
4435     % add up previous version of \@tempa
4436     \@tempm
4437 }%

```

Concatenate millies to \@tempm so that it is equal to  $\text{millies}^{n+3}$  at the next iteration. Here we just have plain concatenation, contrary to cardinal for which a prefix can be used instead.

```

4438     \let\@tempm\@tempm
4439     \edef\@tempm{\millies\@tempm}%
4440     \fi
4441     \next
4442 }%
4443 \let\@tempa\@empty
4444 \let\next\@templ
4445 \@templ

```

Now propagate expansion of tempa into #2 after closing bracket.

```

4446 \def\@tempb##1{\def\@tempa{\def#2{##1}}}%
4447 \expandafter\@tempb\expandafter{\@tempa}%
4448 \expandafter
4449 }\@tempa
4450 }

```

Stuff for calling macros. Construct `\fc@call<some macro>` can be used to pass two arguments to `<some macro>` with a configurable calling convention:

- the calling convention is such that there is one mandatory argument `< marg >` and an optional argument `< oarg >`
- either `\fc@call` is `\let` to be equal to `\fc@call@opt@arg@second`, and then calling convention is that the `< marg >` is first and `< oarg >` is second,
- or `\fc@call` is `\let` to be equal to `\fc@call@opt@arg@first`, and then calling convention is that the `< oarg >` is first and `< aarg >` is second,
- if `< oarg >` is absent, then it is by convention set empty,
- `< some macro >` is supposed to have two mandatory arguments of which `< oarg >` is passed to the first, and `< marg >` is passed to the second, and
- `< some macro >` is called within a group.

```

4451 \def\fc@call@opt@arg@second#1#2{%
4452   \def\@tempb{%
4453     \ifx[\@tempa
4454       \def\@tempc[####1]{%
4455         {#1{####1}{#2}}%
4456       }%
4457     \else
4458       \def\@tempc{{#1}{#2}}%
4459     \fi
4460   \@tempc
4461 }%
4462 \futurelet\@tempa
4463 \@tempb
4464 }

4465 \def\fc@call@opt@arg@first#1{%
4466   \def\@tempb{%
4467     \ifx[\@tempa
4468       \def\@tempc[####1]####2{{#1{####1}{####2}}}%
4469     \else
4470       \def\@tempc####1{{#1}{####1}}%
4471     \fi
4472   \@tempc
4473 }%
4474 \futurelet\@tempa
4475 \@tempb
4476 }
4477
4478 \let\fc@call\fc@call@opt@arg@first

```

#### User API.

`\latinnumeralstringnum` Macro `\@latinnumeralstringnum`. Arguments as follows:

- #1 local options
- #2 input number

```

4479 \newcommand*{\@latinnumeralstringnum}[2]{%
4480   \setkeys{fcprefix}{#1}%
4481   \fc@latin@numeral@pefix{#2}\@tempa
4482   \@tempa
4483 }

```

#### Arguments as follows:

- #1 local options
- #2 input counter

```

4484 \newcommand*{\@latinnumeralstring}[2]{%
4485   \setkeys{fcprefix}{#1}%
4486   \expandafter\let\expandafter
4487     \@tempa\expandafter\csname c@#2\endcsname
4488   \expandafter\fc@latin@numeral@pefix\expandafter{\the\@tempa}\@tempa
4489   \@tempa
4490 }

```

```

4491 \newcommand*\latinnumeralstring}{%
4492   \fc@call\@latinnumeralstring
4493 }

4494 \newcommand*\latinnumeralstringnum}{%
4495   \fc@call\@latinnumeralstringnum
4496 }

```

## 9.4 fmtcount.sty

This section deals with the code for `fmtcount.sty`

```

4497 \NeedsTeXFormat{LaTeX2e}
4498 \ProvidesPackage{fmtcount}[2017/12/24 v3.05]
4499 \RequirePackage{ifthen}

4500 \RequirePackage{xkeyval}
4501 \RequirePackage{etoolbox}
4502 \RequirePackage{fcprefix}

```

Need to use `\new@ifnextchar` instead of `\@ifnextchar` in commands that have a final optional argument (such as `\gls`) so require `amsgen`.

```
4503 \RequirePackage{amsgen}
```

These commands need to be defined before the configuration file is loaded.

Define the macro to format the `st`, `nd`, `rd` or `th` of an ordinal.

```

\fc@orddef@ult
4504 \providecommand*\fc@orddef@ult}[1]{\fc@textsuperscript{#1}}

```

`c@ord@multiling`

```

4505 \providecommand*\fc@ord@multiling}[1]{%
4506   \ifcsundef{fc@\language@name @alias@of}{%

```

Not a supported language, just use the default setting:

```

4507   \fc@orddef@ult{#1}}{%
4508   \expandafter\let\expandafter\@tempa\csname fc@\language@name @alias@of\endcsname
4509   \ifcsundef{fc@ord@\@tempa}{%

```

Not language specific setting, just use the default setting:

```
4510   \fc@orddef@ult{#1}}{%

```

Language with specific setting, use that setting:

```
4511 \csname fc@ord@\@tempa\endcsname{#1}}}}

```

`\padzeroes` `\padzeroes[⟨n⟩]`

Specifies how many digits should be displayed for commands such as `\decimal` and `\binary`.

```

4512 \newcount\c@padzeroesN
4513 \c@padzeroesN=1\relax
4514 \providecommand*\padzeroes}[1][17]{\c@padzeroesN=#1}

```

`\FCloadlang` `\FCloadlang{<language>}`

Load `fmtcount` language file, `fc-<language>.def`, unless already loaded. Unfortunately neither `babel` nor `polyglossia` keep a list of loaded dialects, so we can't load all the necessary `def` files in the preamble as we don't know which dialects the user requires. Therefore the dialect definitions get loaded when a command such as `\ordinalnum` is used, if they haven't already been loaded.

```
4515 \newcount\fc@tmpcatcode
4516 \def\fc@languages{}%
4517 \def\fc@mainlang{}%
4518 \newcommand*\FCloadlang[1]{%
4519   \@FC@iflangloaded{#1}{}%
4520   {%
4521     \fc@tmpcatcode=\catcode'\@ \relax
4522     \catcode '\@ 11 \relax
4523     \InputIfFileExists{fc-#1.def}%
4524     {%
4525       \ifdefempty{\fc@languages}%
4526       {%
4527         \gdef\fc@languages{#1}%
4528       }%
4529       {%
4530         \gappto\fc@languages{,#1}%
4531       }%
4532       \gdef\fc@mainlang{#1}%
4533     }%
4534   }%
4535   \catcode '\@ \fc@tmpcatcode \relax
4536 }%
4537 }
```

`\@FC@iflangloaded` `\@FC@iflangloaded{<language>}{<true>}{<false>}`

If `fmtcount` language definition file `fc-<language>.def` has been loaded, do `<true>` otherwise do `<false>`

```
4538 \newcommand*\@FC@iflangloaded[3]{%
4539   \ifcsundef{ver@fc-#1.def}{#3}{#2}%
4540 }
```

`\ProvidesFCLanguage` Declare `fmtcount` language definition file. Adapted from `\ProvidesFile`.

```
4541 \newcommand*\ProvidesFCLanguage[1]{%
4542   \ProvidesFile{fc-#1.def}%
4543 }
```

We need that flag to remember that a language has been loaded via package option, so that in the end we can set `fmtcount` in `multiling`

```
4544 \newif\iffmtcount@language@option
4545 \fmtcount@language@optionfalse
```

`\DeclareLanguageList` Declare list of supported languages, as a comma separated list. No space, no empty items. Each item is a language for which `\fmtcount` is able to load language specific definitions. Aliases but be *after* their meaning, for instance ‘american’ being an alias of ‘USenglish’, it has to appear after it in the list. The raison d’être of this list is to commonalize iteration on languages for the two following purposes:

- loading language definition as a result of the language being used by babel/polyglossia
- loading language definition as a result of package option

These two purposes cannot be handled in the same pass, we need two different passes otherwise there would be some corner cases when a package would be required — as a result of loading language definition for one language — between a `\DeclareOption` and a `\ProcessOption` which is forbidden by L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>.

```
4546 \newcommand*\fc@supported@language@list{%
4547 english,%
4548 UKenglish,%
4549 brazilian,%
4550 british,%
4551 USenglish,%
4552 american,%
4553 spanish,%
4554 portuges,%
4555 portuguese,%
4556 french,%
4557 frenchb,%
4558 francais,%
4559 german,%
4560 germanb,%
4561 ngerman,%
4562 ngermanb,%
4563 italian}
```

`\fc@iterate@on@languages` `\fc@iterate@on@languages{\<body>}`

Now make some language iterator, note that for the following to work properly `\fc@supported@language@list` must not be empty. `\<body>` is a macro that takes one argument, and `\fc@iterate@on@languages` applies it iteratively:

```
4564 \newcommand*\fc@iterate@on@languages[1]{%
4565 \ifx\fc@supported@language@list\empty
```

That case should never happen!

```
4566 \PackageError{fmtcount}{Macro ‘\protect\fc@iterate@on@languages’ is empty}{You should never
4567 Something is broken within \texttt{fmtcount}, please report the issue on
```

```

4568     \texttt{https://github.com/search?q=fmtcount\&ref=cmdform\&type=Issues}}}%
4569 \else
4570     \let\fc@iterate@on@languages@body#1
4571     \expandafter\@fc@iterate@on@languages\fc@supported@language@list,\@nil,%
4572 \fi
4573 }
4574 \def\@fc@iterate@on@languages#1,{%
4575     {%
4576     \def\@tempa{#1}%
4577     \ifx\@tempa\@nnil
4578     \let\@tempa\@empty
4579     \else
4580     \def\@tempa{%
4581     \fc@iterate@on@languages@body{#1}%
4582     \@fc@iterate@on@languages
4583     }%
4584     \fi
4585     \expandafter
4586     }\@tempa
4587 }%

```

orpolyglossialdf

```
\@fc@loadifbabelorpolyglossialdf{<language>}
```

Loads `fmtcount` language file, `fc-⟨language⟩.def`, if one of the following condition is met:

- `babel` language definition file `⟨language⟩.ldf` has been loaded — conditionally to compilation with `latex`, not `xelatex`.
- `polyglossia` language definition file `gloss-⟨language⟩.ldf` has been loaded — conditionally to compilation with `xelatex`, not `latex`.
- `⟨language⟩` option has been passed to package `fmtcount`.

```

4588 \newcommand*\@fc@loadifbabelldf[1]{\ifcsundef{ver@#1.ldf}}{\FCloadlang{#1}}
4589 \newcommand*\@fc@loadifbabelorpolyglossialdf[1]{
4590 \ifpackageloaded{polyglossia}{%
4591 \def\@fc@loadifbabelorpolyglossialdf#1{\IfFileExists{gloss-#1.ldf}{\ifcsundef{#1@loaded}}{\F
4592 \@fc@loadifbabelldf{#1}%
4593 }%
4594 }\ifpackageloaded{babel}{%
4595 \let\@fc@loadifbabelorpolyglossialdf\@fc@loadifbabelldf
4596 }{}}

```

Load appropriate language definition files:

```
4597 \fc@iterate@on@languages\@fc@loadifbabelorpolyglossialdf
```

By default all languages are unique — i.e. aliases not yet defined.

```

4598 \def\fc@iterate@on@languages@body#1{%
4599 \expandafter\def\csname fc@#1@alias@of\endcsname{#1}}
4600 \expandafter\@fc@iterate@on@languages\fc@supported@language@list,\@nil,%

```

Now define those languages that are aliases of another language. This is done with: `\@tempa {<alias>}{<language>}`

```
4601 \def\@tempa#1#2{%
4602   \expandafter\def\csname fc@#1@alias@of\endcsname{#2}%
4603 }%
4604 \@tempa{frenchb}{french}
4605 \@tempa{français}{french}
4606 \@tempa{germanb}{german}
4607 \@tempa{ngermanb}{german}
4608 \@tempa{ngerman}{german}
4609 \@tempa{british}{english}
4610 \@tempa{american}{USenglish}
```

Now, thanks to the aliases, we are going to define one option for each language, so that each language can have its own settings.

```
4611 \def\fc@iterate@on@languages@body#1{%
4612   \define@key{fmtcount}{#1}[]{%
4613     \@FC@iflangloaded{#1}%
4614     {%
4615       \setkeys{fc\csname fc@#1@alias@of\endcsname}{##1}%
4616     }{%
4617       \PackageError{fmtcount}%
4618         {Language ‘#1’ not defined}%
4619         {You need to load \ifxetex polyglossia\else babel\fi\space before loading fmtcount}%
4620     }%
4621 }%
4622 \ifthenelse{\equal{\csname fc@#1@alias@of\endcsname}{#1}}{%
4623   \define@key{fc\csname fc@#1@alias@of\endcsname}{fmtord}{%
4624     \ifthenelse{\equal{##1}{raise}}{\or\equal{##1}{level}}{%
4625       \expandafter\let\expandafter\@tempa\csname fc@set@ord@as@##1\endcsname
4626       \expandafter\@tempa\csname fc@ord@#1\endcsname
4627     }{%
4628       \ifthenelse{\equal{##1}{undefine}}{%
4629         \expandafter\let\csname fc@ord@#1\endcsname\undefined
4630       }{%
4631         \PackageError{fmtcount}%
4632           {Invalid value ‘##1’ to fmtord key}%
4633           {Option ‘fmtord’ can only take the values ‘level’, ‘raise’
4634             or ‘undefine’}%
4635       }%
4636     }%
4637 }%
```

When the language #1 is an alias, do the same as the language of which it is an alias:

```
4638   \expandafter\let\expandafter\@tempa\csname KV@\csname fc@#1@alias@of\endcsname @fmtord\endc
4639   \expandafter\let\csname KV@#1@fmtord\endcsname\@tempa
4640 }%
4641 }
4642 \expandafter\@fc@iterate@on@languages\fc@supported@language@list,\@nil,%
```

`fmtord` Key to determine how to display the ordinal

```
4643 \def\fc@set@ord@as@level#1{%
4644   \def#1##1{##1}%
4645 }
4646 \def\fc@set@ord@as@raise#1{%
4647   \let#1\fc@textsuperscript
4648 }
4649 \define@key{fmtcount}{fmtord}{%
4650   \ifthenelse{\equal{#1}{level}
4651             \or\equal{#1}{raise}}%
4652   {%
4653     \csname fc@set@ord@as@#1\endcsname\fc@orddef@ult
4654     \def\fmtcount@fmtord{#1}%
4655   }%
4656   {%
4657     \PackageError{fmtcount}%
4658     {Invalid value ‘#1’ to fmtord key}%
4659     {Option ‘fmtord’ can only take the values ‘level’ or ‘raise’}%
4660   }%
4661 }
```

`\iffmtord@abbrv` Key to determine whether the ordinal superscript should be abbreviated (language dependent, currently only affects French ordinals, non-abbreviated French ordinals ending — i.e. ‘ier’ and ‘ième’ — are considered faulty.)

```
4662 \newif\iffmtord@abbrv
4663 \fmtord@abbrvtrue
4664 \define@key{fmtcount}{abbrv}[true]{%
4665   \ifthenelse{\equal{#1}{true}\or\equal{#1}{false}}%
4666   {%
4667     \csname fmtord@abbrv#1\endcsname
4668   }%
4669   {%
4670     \PackageError{fmtcount}%
4671     {Invalid value ‘#1’ to fmtord key}%
4672     {Option ‘abbrv’ can only take the values ‘true’ or
4673     ‘false’}%
4674   }%
4675 }
```

`prefix`

```
4676 \define@key{fmtcount}{prefix}[scale=long]{%
4677   \RequirePackage{fmtprefix}%
4678   \fmtprefixsetoption{#1}%
4679 }
```

`countsetoptions` Define command to set options.

```
4680 \def\fmtcountsetoptions{%
4681   \def\fmtcount@fmtord{}}%
```

```
4682 \setkeys{fmtcount}}%
```

Load configuration file if it exists. This needs to be done before the package options, to allow the user to override the settings in the configuration file.

```
4683 \InputIfFileExists{fmtcount.cfg}%
4684 {%
4685 \PackageInfo{fmtcount}{Using configuration file fmtcount.cfg}%
4686 }%
4687 {%
4688 }
```

ption@lang@list

```
4689 \newcommand*{\fmtcount@loaded@by@option@lang@list}{}
```

`\metalanguage` Option *<language>* causes language *<language>* to be registered for loading.

```
4690 \newcommand*{\@fc@declare@language@option[1]}{%
4691 \DeclareOption{#1}{%
4692 \ifx\fmtcount@loaded@by@option@lang@list\@empty
4693 \def\fmtcount@loaded@by@option@lang@list{#1}%
4694 \else
4695 \edef\fmtcount@loaded@by@option@lang@list{\fmtcount@loaded@by@option@lang@list,#1}%
4696 \fi
4697 }%
4698 \@fc@iterate@on@languages\@fc@declare@language@option
```

level

```
4699 \DeclareOption{level}{\def\fmtcount@fmtord{level}}%
4700 \def\@fc@orddef@ult#1{#1}}
```

raise

```
4701 \DeclareOption{raise}{\def\fmtcount@fmtord{raise}}%
4702 \def\@fc@orddef@ult#1{\@fc@textsuperscript{#1}}
```

Process package options

```
4703 \ProcessOptions\relax
```

Now we do the loading of all languages that have been set by option to be loaded.

```
4704 \ifx\fmtcount@loaded@by@option@lang@list\@empty\else
4705 \def\@fc@iterate@on@languages@body#1{%
4706 \@FC@iflangloaded{#1}{%
4707 \fmtcount@language@optiontrue
4708 \FCloadlang{#1}%
4709 }}
4710 \expandafter\@fc@iterate@on@languages\fmtcount@loaded@by@option@lang@list,\@nil,%
4711 \fi
```

```
\@FCmodulo \@FCmodulo{<count reg>}{<n>}
```

Sets the count register to be its value modulo  $\langle n \rangle$ . This is used for the date, time, ordinal and numberstring commands. (The fmtcount package was originally part of the datetime package.)

```
4712 \newcount\@DT@modctr
4713 \newcommand*\@FCmodulo}[2]{%
4714   \@DT@modctr=#1\relax
4715   \divide \@DT@modctr by #2\relax
4716   \multiply \@DT@modctr by #2\relax
4717   \advance #1 by -\@DT@modctr
4718 }
```

The following registers are needed by \@ordinal etc

```
4719 \newcount\@ordinalctr
4720 \newcount\@orgargctr
4721 \newcount\@strctr
4722 \newcount\@tmpstrctr
```

Define commands that display numbers in different bases. Define counters and conditionals needed.

```
4723 \newif\if@DT@padzeroes
4724 \newcount\@DT@loopN
4725 \newcount\@DT@X
```

`\binarynum` Converts a decimal number to binary, and display.

```
4726 \newrobustcmd*\@binary}[1]{%
4727   \@DT@padzeroestrue
4728   \@DT@loopN=17\relax
4729   \@strctr=\@DT@loopN
4730   \whiledo{\@strctr<\c@padzeroesN}{0\advance\@strctr by \@ne}%
4731   \@strctr=65536\relax
4732   \@DT@X=#1\relax
4733   \loop
4734     \@DT@modctr=\@DT@X
4735     \divide\@DT@modctr by \@strctr
4736     \ifthenelse{\boolean{@DT@padzeroes}
4737       \and \(\@DT@modctr=0\}
4738       \and \(\@DT@loopN>\c@padzeroesN\)}%
4739     {}%
4740     {\the\@DT@modctr}%
4741     \ifnum\@DT@modctr=0\else\@DT@padzeroesfalse\fi
4742     \multiply\@DT@modctr by \@strctr
4743     \advance\@DT@X by -\@DT@modctr
4744     \divide\@strctr by \tw@
4745     \advance\@DT@loopN by \m@ne
4746     \ifnum\@strctr>\@ne
4747     \repeat
4748     \the\@DT@X
4749 }
4750
```

```
4751 \let\binarynum=\@binary
```

`\octalnum` Converts a decimal number to octal, and displays.

```
4752 \newrobustcmd*{\@octal}[1]{%
4753   \@DT@X=#1\relax
4754   \ifnum\@DT@X>32768
4755     \PackageError{fmtcount}%
4756     {Value of counter too large for \protect\@octal}
4757     {Maximum value 32768}
4758   \else
4759     \@DT@padzeroestru
4760     \@DT@loopN=6\relax
4761     \@strctr=\@DT@loopN
4762     \whiledo{\@strctr<\c@padzeroesN}{0\advance\@strctr by \@ne}%
4763     \@strctr=32768\relax
4764     \loop
4765       \@DT@modctr=\@DT@X
4766       \divide\@DT@modctr by \@strctr
4767       \ifthenelse{\boolean{@DT@padzeroes}
4768         \and \(\@DT@modctr=0\}
4769         \and \(\@DT@loopN>\c@padzeroesN\)}%
4770       {\the\@DT@modctr}%
4771       \ifnum\@DT@modctr=0\else\@DT@padzeroesfalse\fi
4772       \multiply\@DT@modctr by \@strctr
4773       \advance\@DT@X by -\@DT@modctr
4774       \divide\@strctr by \@viiipt
4775       \advance\@DT@loopN by \m@ne
4776     \ifnum\@strctr>\@ne
4777     \repeat
4778     \the\@DT@X
4779   \fi
4780 }
4781 \let\octalnum=\@octal
```

`@hexadecimalnum` Converts number from 0 to 15 into lowercase hexadecimal notation.

```
4782 \newcommand*{\@hexadecimal}[1]{%
4783   \ifcase#1\or1\or2\or3\or4\or5\or
4784   6\or7\or8\or9\or a\or b\or c\or d\or e\or f\fi
4785 }
```

`\hexadecimalnum` Converts a decimal number to a lowercase hexadecimal number, and displays it.

```
4786 \newrobustcmd*{\@hexadecimal}[1]{%
4787   \@DT@padzeroestru
4788   \@DT@loopN=\@vpt
4789   \@strctr=\@DT@loopN
4790   \whiledo{\@strctr<\c@padzeroesN}{0\advance\@strctr by \@ne}%
4791   \@strctr=65536\relax
4792   \@DT@X=#1\relax
4793   \loop
```

```

4794 \DT@modctr=\DT@X
4795 \divide\DT@modctr by \@strctr
4796 \ifthenelse{\boolean{DT@padzeroes}
4797 \and \(\DT@modctr=0\}
4798 \and \(\DT@loopN>\c@padzeroesN\)}
4799 {\@@hexadecimal\DT@modctr}%
4800 \ifnum\DT@modctr=0\else\DT@padzeroesfalse\fi
4801 \multiply\DT@modctr by \@strctr
4802 \advance\DT@X by -\DT@modctr
4803 \divide\@strctr by 16\relax
4804 \advance\DT@loopN by \m@ne
4805 \ifnum\@strctr>\@ne
4806 \repeat
4807 \@@hexadecimal\DT@X
4808 }
4809 \let\hexadecimalnum=\@hexadecimal

```

**\Hexadecimalnum** Converts number from 0 to 15 into uppercase hexadecimal notation.

```

4810 \newcommand*{\@@Hexadecimal}[1]{%
4811 \ifcase#1\or1\or2\or3\or4\or5\or6\or
4812 7\or8\or9\or A\or B\or C\or D\or E\or F\fi
4813 }

```

**\Hexadecimalnum** Uppercase hexadecimal

```

4814 \newrobustcmd*{\@@Hexadecimal}[1]{%
4815 \DT@padzeroestrue
4816 \DT@loopN=5\relax
4817 \@strctr=\DT@loopN
4818 \whiledo{\@strctr<\c@padzeroesN}{0\advance\@strctr by \@ne}%
4819 \@strctr=65536\relax
4820 \DT@X=#1\relax
4821 \loop
4822 \DT@modctr=\DT@X
4823 \divide\DT@modctr by \@strctr
4824 \ifthenelse{\boolean{DT@padzeroes}
4825 \and \(\DT@modctr=0\}
4826 \and \(\DT@loopN>\c@padzeroesN\)}%
4827 {\@@Hexadecimal\DT@modctr}%
4828 \ifnum\DT@modctr=0\else\DT@padzeroesfalse\fi
4829 \multiply\DT@modctr by \@strctr
4830 \advance\DT@X by -\DT@modctr
4831 \divide\@strctr by 16\relax
4832 \advance\DT@loopN by \m@ne
4833 \ifnum\@strctr>\@ne
4834 \repeat
4835 \@@Hexadecimal\DT@X
4836 }
4837
4838 \let\Hexadecimalnum=\@@Hexadecimal

```

`\aaalphnum` Lowercase alphabetical representation (a ... z aa ... zz)

```
4839 \newrobustcmd*{\@aaalph}{\fc@aaalph\@alph}
4840 \newcommand*\fc@aaalph[2]{%
4841   \@DT@loopN=#2\relax
4842   \@DT@X\@DT@loopN
4843   \advance\@DT@loopN by \m@ne
4844   \divide\@DT@loopN by 26\relax
4845   \@DT@modctr=\@DT@loopN
4846   \multiply\@DT@modctr by 26\relax
4847   \advance\@DT@X by \m@ne
4848   \advance\@DT@X by -\@DT@modctr
4849   \advance\@DT@loopN by \@ne
4850   \advance\@DT@X by \@ne
4851   \edef\@tempa{#1\@DT@X}%
4852   \loop
4853     \@tempa
4854     \advance\@DT@loopN by \m@ne
4855   \ifnum\@DT@loopN>0
4856   \repeat
4857 }
4858
4859 \let\aaalphnum=\@aaalph
```

`\AAAalphnum` Uppercase alphabetical representation (a ... z aa ... zz)

```
4860 \newrobustcmd*{\@AAAalph}{\fc@aaalph\@Alph}%
4861
4862 \let\AAAalphnum=\@AAAalph
```

`\abalphnum` Lowercase alphabetical representation

```
4863 \newrobustcmd*{\@abalph}{\fc@abalph\@alph}%
4864 \newcommand*\fc@abalph[2]{%
4865   \@DT@X=#2\relax
4866   \ifnum\@DT@X>17576\relax
4867     \ifx#1\@alph\def\@tempa{\@abalph}%
4868     \else\def\@tempa{\@ABAlph}\fi
4869     \PackageError{fmtcount}{%
4870       {Value of counter too large for \expandafter\protect\@tempa}%
4871       {Maximum value 17576}}%
4872   \else
4873     \@DT@padzeroestruer
4874     \@strctr=17576\relax
4875     \advance\@DT@X by \m@ne
4876     \loop
4877       \@DT@modctr=\@DT@X
4878       \divide\@DT@modctr by \@strctr
4879       \ifthenelse{\boolean{\@DT@padzeroes}
4880         \and \!(\@DT@modctr=1)}%
4881       {{#1\@DT@modctr}}%
4882     \ifnum\@DT@modctr=\@ne\else\@DT@padzeroesfalse\fi
```

```

4883     \multiply\@DT@modctr by \@strctr
4884     \advance\@DT@X by -\@DT@modctr
4885     \divide\@strctr by 26\relax
4886     \ifnum\@strctr>\@ne
4887     \repeat
4888     \advance\@DT@X by \@ne
4889     #1\@DT@X
4890     \fi
4891 }
4892
4893 \let\abalphnum=\@abalph

```

`\ABAlphnum` Uppercase alphabetical representation

```

4894 \newrobustcmd*{\@ABAlph}{\fc@abalph\@Alph}%
4895 \let\ABAlphnum=\@ABAlph

```

`\@fmtc@count` Recursive command to count number of characters in argument. `\@strctr` should be set to zero before calling it.

```

4896 \def\@fmtc@count#1#2\relax{%
4897   \if\relax#1%
4898   \else
4899     \advance\@strctr by 1\relax
4900     \@fmtc@count#2\relax
4901   \fi
4902 }

```

`\@decimal` Format number as a decimal, possibly padded with zeroes in front.

```

4903 \newrobustcmd*{\@decimal}[1]{%
4904   \@strctr=0\relax
4905   \expandafter\@fmtc@count\number#1\relax
4906   \@DT@loopN=\c@padzeroesN
4907   \advance\@DT@loopN by -\@strctr
4908   \ifnum\@DT@loopN>0\relax
4909     \@strctr=0\relax
4910     \whiledo{\@strctr < \@DT@loopN}{0\advance\@strctr by 1\relax}%
4911   \fi
4912   \number#1\relax
4913 }
4914
4915 \let\decimalnum=\@decimal

```

`\FCordinal` `\FCordinal{<number>}`

This is a bit cumbersome. Previously `\@ordinal` was defined in a similar way to `\abalph` etc. This ensured that the actual value of the counter was written in the new label stuff in the `.aux` file. However adding in an optional argument to determine the gender for multilingual compatibility messed things up somewhat. This was the only work around I could get to keep

the the cross-referencing stuff working, which is why the optional argument comes *after* the compulsory argument, instead of the usual manner of placing it before. Note however, that putting the optional argument means that any spaces will be ignored after the command if the optional argument is omitted. Version 1.04 changed `\ordinal` to `\FCordinal` to prevent it clashing with the memoir class.

```
4916 \newcommand{\FCordinal}[1]{%
4917   \ordinalnum{%
4918     \the\value{#1}}%
4919 }
```

`\ordinal` If `\ordinal` isn't defined make `\ordinal` a synonym for `\FCordinal` to maintain compatibility with previous versions.

```
4920 \ifcsundef{ordinal}
4921 {\let\ordinal\FCordinal}%
4922 {%
4923   \PackageWarning{fmtcount}%
4924   {\protect\ordinal \space already defined use
4925    \protect\FCordinal \space instead.}
4926 }
```

`\ordinalnum` Display ordinal where value is given as a number or count register instead of a counter:

```
4927 \newrobustcmd*{\ordinalnum}[1]{%
4928   \new@ifnextchar[%
4929   {\@ordinalnum{#1}}%
4930   {\@ordinalnum{#1}[m]}%
4931 }
```

`\@ordinalnum` Display ordinal according to gender (neuter added in v1.1, `\xspace` added in v1.2, and removed in v1.3<sup>7</sup>):

```
4932 \def\@ordinalnum#1[#2]{%
4933   {%
4934     \ifthenelse{\equal{#2}{f}}%
4935     {%
4936       \protect\@ordinalF{#1}{\@fc@ordstr}%
4937     }%
4938     {%
4939       \ifthenelse{\equal{#2}{n}}%
4940       {%
4941         \protect\@ordinalN{#1}{\@fc@ordstr}%
4942       }%
4943       {%
4944         \ifthenelse{\equal{#2}{m}}%
4945         {}%
4946         {%
4947           \PackageError{fmtcount}%
4948           {Invalid gender option ‘#2’}%

```

---

<sup>7</sup>I couldn't get it to work consistently both with and without the optional argument

```

4949         {Available options are m, f or n}%
4950     }%
4951     \protect\@ordinalM{#1}{\@fc@ordstr}%
4952 }%
4953 }%
4954 \@fc@ordstr
4955 }%
4956 }

```

`\storeordinal` Store the ordinal (first argument is identifying name, second argument is a counter.)

```

4957 \newcommand*\storeordinal}[2]{%
4958   {%
4959     \toks0{\storeordinalnum{#1}}%
4960     \expandafter
4961   }\the\toks0\expandafter{%
4962     \the\value{#2}}%
4963 }

```

`storeordinalnum` Store ordinal (first argument is identifying name, second argument is a number or count register.)

```

4964 \newrobustcmd*\storeordinalnum}[2]{%
4965   \@ifnextchar[%
4966   {\@storeordinalnum{#1}{#2}}%
4967   {\@storeordinalnum{#1}{#2}[m]}%
4968 }

```

`storeordinalnum` Store ordinal according to gender:

```

4969 \def\@storeordinalnum#1#2[#3]{%
4970   \ifthenelse{\equal{#3}{f}}%
4971   {%
4972     \protect\@ordinalF{#2}{\@fc@ord}
4973   }%
4974   {%
4975     \ifthenelse{\equal{#3}{n}}%
4976     {%
4977       \protect\@ordinalN{#2}{\@fc@ord}%
4978     }%
4979     {%
4980       \ifthenelse{\equal{#3}{m}}%
4981       {}%
4982       {%
4983         \PackageError{fmtcount}%
4984         {Invalid gender option '#3'}%
4985         {Available options are m or f}%
4986       }%
4987       \protect\@ordinalM{#2}{\@fc@ord}%
4988     }%
4989   }%
4990 \expandafter\let\csname @fcs@#1\endcsname\@fc@ord

```

4991 }

`\FMCuse` Get stored information:

```
4992 \newcommand*{\FMCuse}[1]{\csname @fcs@#1\endcsname}
```

`\ordinalstring` Display ordinal as a string (argument is a counter)

```
4993 \newcommand*{\ordinalstring}[1]{%
4994   \ordinalstringnum{\expandafter\expandafter\expandafter
4995     \the\value{#1}}%
4996 }
```

`ordinalstringnum` Display ordinal as a string (argument is a count register or number.)

```
4997 \newrobustcmd*{\ordinalstringnum}[1]{%
4998   \new@ifnextchar[%
4999     {\@ordinal@string{#1}}%
5000   {\@ordinal@string{#1}[m]}%
5001 }
```

`@ordinal@string` Display ordinal as a string according to gender.

```
5002 \def\@ordinal@string#1[#2]{%
5003   {%
5004     \ifthenelse{\equal{#2}{f}}%
5005     {%
5006       \protect\@ordinalstringF{#1}{\@fc@ordstr}%
5007     }%
5008     {%
5009       \ifthenelse{\equal{#2}{n}}%
5010       {%
5011         \protect\@ordinalstringN{#1}{\@fc@ordstr}%
5012       }%
5013       {%
5014         \ifthenelse{\equal{#2}{m}}%
5015         {}%
5016         {%
5017           \PackageError{fmtcount}%
5018             {Invalid gender option ‘#2’ to \protect\ordinalstring}%
5019             {Available options are m, f or n}%
5020         }%
5021         \protect\@ordinalstringM{#1}{\@fc@ordstr}%
5022       }%
5023     }%
5024     \@fc@ordstr
5025   }%
5026 }
```

`reordinalstring` Store textual representation of number. First argument is identifying name, second argument is the counter set to the required number.

```
5027 \newcommand*{\storeordinalstring}[2]{%
```

```

5028 {%
5029   \toks0{\storeordinalstringnum{#1}}%
5030   \expandafter
5031 } \the\toks0\expandafter{\the\value{#2}}%
5032 }

```

`\storeordinalstringnum` Store textual representation of number. First argument is identifying name, second argument is a count register or number.

```

5033 \newrobustcmd*{\storeordinalstringnum}[2]{%
5034   \ifnextchar[%
5035     {\@store@ordinal@string{#1}{#2}}%
5036     {\@store@ordinal@string{#1}{#2}[m]}%
5037 }

```

`\@ordinal@string` Store textual representation of number according to gender.

```

5038 \def\@store@ordinal@string#1#2[#3]{%
5039   \ifthenelse{\equal{#3}{f}}%
5040   {%
5041     \protect\@ordinalstringF{#2}{\@fc@ordstr}%
5042   }%
5043   {%
5044     \ifthenelse{\equal{#3}{n}}%
5045     {%
5046       \protect\@ordinalstringN{#2}{\@fc@ordstr}%
5047     }%
5048     {%
5049       \ifthenelse{\equal{#3}{m}}%
5050       {}%
5051       {%
5052         \PackageError{fmtcount}%
5053           {Invalid gender option ‘#3’ to \protect\ordinalstring}%
5054           {Available options are m, f or n}%
5055       }%
5056       \protect\@ordinalstringM{#2}{\@fc@ordstr}%
5057     }%
5058   }%
5059   \expandafter\let\csname @fcs@#1\endcsname\@fc@ordstr
5060 }

```

`\Ordinalstring` Display ordinal as a string with initial letters in upper case (argument is a counter)

```

5061 \newcommand*{\Ordinalstring}[1]{%
5062   \Ordinalstringnum{\expandafter\expandafter\expandafter\the\value{#1}}%
5063 }

```

`\storeordinalstringnum` Display ordinal as a string with initial letters in upper case (argument is a number or count register)

```

5064 \newrobustcmd*{\Ordinalstringnum}[1]{%
5065   \new@ifnextchar[%

```

```

5066 {\@Ordinal@string{#1}}%
5067 {\@Ordinal@string{#1}[m]}%
5068 }

```

**@Ordinal@string** Display ordinal as a string with initial letters in upper case according to gender

```

5069 \def\@Ordinal@string#1[#2]{%
5070  {%
5071   \ifthenelse{\equal{#2}{f}}%
5072   {%
5073    \protect\@OrdinalstringF{#1}{\@fc@ordstr}%
5074   }%
5075   {%
5076    \ifthenelse{\equal{#2}{n}}%
5077    {%
5078     \protect\@OrdinalstringN{#1}{\@fc@ordstr}%
5079    }%
5080    {%
5081     \ifthenelse{\equal{#2}{m}}%
5082     {}%
5083     {%
5084      \PackageError{fmtcount}%
5085       {Invalid gender option ‘#2’}%
5086       {Available options are m, f or n}%
5087     }%
5088     \protect\@OrdinalstringM{#1}{\@fc@ordstr}%
5089    }%
5090   }%
5091   \@fc@ordstr
5092 }%
5093 }

```

**reOrdinalstring** Store textual representation of number, with initial letters in upper case. First argument is identifying name, second argument is the counter set to the required number.

```

5094 \newcommand*\storeOrdinalstring}[2]{%
5095  {%
5096   \toks0{\storeOrdinalstringnum{#1}}%
5097   \expandafter
5098  }\the\toks0\expandafter{\the\value{#2}}%
5099 }

```

**rdinalstringnum** Store textual representation of number, with initial letters in upper case. First argument is identifying name, second argument is a count register or number.

```

5100 \newrobustcmd*\storeOrdinalstringnum}[2]{%
5101  \ifnextchar[%
5102  {\@store@Ordinal@string{#1}{#2}}%
5103  {\@store@Ordinal@string{#1}{#2}[m]}%
5104 }

```

**@Ordinal@string** Store textual representation of number according to gender, with initial letters in upper case.

```

5105 \def\@store@Ordinal@string#1#2[#3]{%
5106 \ifthenelse{\equal{#3}{f}}{%
5107 {%
5108   \protect\@OrdinalstringF{#2}{\@fc@ordstr}%
5109 }%
5110 {%
5111   \ifthenelse{\equal{#3}{n}}{%
5112     {%
5113       \protect\@OrdinalstringN{#2}{\@fc@ordstr}%
5114     }%
5115     {%
5116       \ifthenelse{\equal{#3}{m}}{%
5117         }%
5118         {%
5119           \PackageError{fmtcount}%
5120             {Invalid gender option ‘#3’}%
5121             {Available options are m or f}%
5122           }%
5123           \protect\@OrdinalstringM{#2}{\@fc@ordstr}%
5124         }%
5125       }%
5126     \expandafter\let\csname @fcs@#1\endcsname\@fc@ordstr
5127 }

```

`reORDINALstring` Store upper case textual representation of ordinal. The first argument is identifying name, the second argument is a counter.

```

5128 \newcommand*\@storeORDINALstring}[2]{%
5129   {%
5130     \toks0{\@storeORDINALstringnum{#1}}%
5131     \expandafter
5132   }\the\toks0\expandafter{\the\value{#2}}%
5133 }

```

`ORDINALstringnum` As above, but the second argument is a count register or a number.

```

5134 \newrobustcmd*\@storeORDINALstringnum}[2]{%
5135   \@ifnextchar[%
5136     {\@store@ORDINAL@string{#1}{#2}}%
5137     {\@store@ORDINAL@string{#1}{#2}[m]}%
5138 }

```

`@ORDINAL@string` Gender is specified as an optional argument at the end.

```

5139 \def\@store@ORDINAL@string#1#2[#3]{%
5140 \ifthenelse{\equal{#3}{f}}{%
5141 {%
5142   \protect\@ordinalstringF{#2}{\@fc@ordstr}%
5143 }%
5144 {%
5145   \ifthenelse{\equal{#3}{n}}{%
5146     {%

```

```

5147     \protect\@ordinalstringN{#2}{\@fc@ordstr}%
5148   }%
5149   {%
5150     \ifthenelse{\equal{#3}{m}}%
5151     {}%
5152     {%
5153       \PackageError{fmtcount}%
5154       {Invalid gender option ‘#3’}%
5155       {Available options are m or f}%
5156     }%
5157     \protect\@ordinalstringM{#2}{\@fc@ordstr}%
5158   }%
5159 }%

5160 \expandafter\protected@edef\csname @fcs@#1\endcsname{%
5161   \noexpand\MakeUppercase{\@fc@ordstr}%
5162 }%
5163 }

```

`\ORDINALstring` Display upper case textual representation of an ordinal. The argument must be a counter.

```

5164 \newcommand*{\ORDINALstring}[1]{%
5165   \ORDINALstringnum{\expandafter\expandafter\expandafter
5166     \the\value{#1}}%
5167 }%
5168 }

```

`\ORDINALstringnum` As above, but the argument is a count register or a number.

```

5169 \newrobustcmd*{\ORDINALstringnum}[1]{%
5170   \new@ifnextchar[%
5171     {\@ORDINAL@string{#1}}%
5172     {\@ORDINAL@string{#1}[m]}%
5173 }

```

`\@ORDINAL@string` Gender is specified as an optional argument at the end.

```

5174 \def\@ORDINAL@string#1[#2]{%
5175   {%
5176     \ifthenelse{\equal{#2}{f}}%
5177     {%
5178       \protect\@ordinalstringF{#1}{\@fc@ordstr}%
5179     }%
5180     {%
5181       \ifthenelse{\equal{#2}{n}}%
5182       {%
5183         \protect\@ordinalstringN{#1}{\@fc@ordstr}%
5184       }%
5185       {%
5186         \ifthenelse{\equal{#2}{m}}%
5187         {}%
5188         {%

```

```

5189         \PackageError{fmtcount}%
5190         {Invalid gender option ‘#2’}%
5191         {Available options are m, f or n}%
5192     }%
5193     \protect\@ordinalstringM{#1}{\@fc@ordstr}%
5194 }%
5195 }%
5196 \MakeUppercase{\@fc@ordstr}%
5197 }%
5198 }

```

`\storenumberstring` Convert number to textual representation, and store. First argument is the identifying name, second argument is a counter containing the number.

```

5199 \newcommand*\storenumberstring}[2]{%
5200   \expandafter\protect\expandafter\storenumberstringnum{#1}{%
5201     \expandafter\the\value{#2}}%
5202 }

```

`\numberstringnum` As above, but second argument is a number or count register.

```

5203 \newcommand*\storenumberstringnum}[2]{%
5204   \@ifnextchar[%
5205     {\@store@number@string{#1}{#2}}%
5206     {\@store@number@string{#1}{#2}[m]}%
5207 }

```

`\@store@number@string` Gender is given as optional argument, *at the end*.

```

5208 \def\@store@number@string#1#2[#3]{%
5209   \ifthenelse{\equal{#3}{f}}{%
5210     {%
5211       \protect\@numberstringF{#2}{\@fc@numstr}%
5212     }%
5213     {%
5214       \ifthenelse{\equal{#3}{n}}{%
5215         {%
5216           \protect\@numberstringN{#2}{\@fc@numstr}%
5217         }%
5218         {%
5219           \ifthenelse{\equal{#3}{m}}{%
5220             {}%
5221             {%
5222               \PackageError{fmtcount}
5223               {Invalid gender option ‘#3’}%
5224               {Available options are m, f or n}%
5225             }%
5226             \protect\@numberstringM{#2}{\@fc@numstr}%
5227           }%
5228         }%
5229       \expandafter\let\csname @fcs@#1\endcsname\@fc@numstr
5230 }

```

`\numberstring` Display textual representation of a number. The argument must be a counter.

```
5231 \newcommand*{\numberstring}[1]{%
5232   \numberstringnum{\expandafter\expandafter\expandafter
5233     \the\value{#1}}%
5234 }
```

`numberstringnum` As above, but the argument is a count register or a number.

```
5235 \newrobustcmd*{\numberstringnum}[1]{%
5236   \new@ifnextchar[%
5237     {\@number@string{#1}}%
5238     {\@number@string{#1}[m]}%
5239 }
```

`\@number@string` Gender is specified as an optional argument *at the end*.

```
5240 \def\@number@string#1[#2]{%
5241   {%
5242     \ifthenelse{\equal{#2}{f}}%
5243     {%
5244       \protect\@numberstringF{#1}{\@fc@numstr}%
5245     }%
5246     {%
5247       \ifthenelse{\equal{#2}{n}}%
5248       {%
5249         \protect\@numberstringN{#1}{\@fc@numstr}%
5250       }%
5251       {%
5252         \ifthenelse{\equal{#2}{m}}%
5253         {}%
5254         {%
5255           \PackageError{fmtcount}%
5256             {Invalid gender option ‘#2’}%
5257             {Available options are m, f or n}%
5258         }%
5259         \protect\@numberstringM{#1}{\@fc@numstr}%
5260       }%
5261     }%
5262     \@fc@numstr
5263   }%
5264 }
```

`storeNumberstring` Store textual representation of number. First argument is identifying name, second argument is a counter.

```
5265 \newcommand*{\storeNumberstring}[2]{%
5266   {%
5267     \toks0{\storeNumberstringnum{#1}}%
5268     \expandafter
5269   }\the\toks0\expandafter{\the\value{#2}}%
5270 }
```

`Numberstringnum` As above, but second argument is a count register or number.

```
5271 \newcommand{\storeNumberstringnum}[2]{%
5272   \@ifnextchar[%
5273     {\@store@Number@string{#1}{#2}}%
5274     {\@store@Number@string{#1}{#2}[m]}}%
5275 }
```

`e@Number@string` Gender is specified as an optional argument *at the end*:

```
5276 \def\@store@Number@string#1#2[#3]{%
5277   \ifthenelse{\equal{#3}{f}}%
5278     {%
5279       \protect\@NumberstringF{#2}{\@fc@numstr}%
5280     }%
5281     {%
5282       \ifthenelse{\equal{#3}{n}}%
5283         {%
5284           \protect\@NumberstringN{#2}{\@fc@numstr}%
5285         }%
5286         {%
5287           \ifthenelse{\equal{#3}{m}}%
5288             {}%
5289             {%
5290               \PackageError{fmtcount}%
5291                 {Invalid gender option ‘#3’}%
5292                 {Available options are m, f or n}%
5293             }%
5294           \protect\@NumberstringM{#2}{\@fc@numstr}%
5295         }%
5296     }%
5297   \expandafter\let\csname @fcs@#1\endcsname\@fc@numstr
5298 }
```

`\Numberstring` Display textual representation of number. The argument must be a counter.

```
5299 \newcommand*{\Numberstring}[1]{%
5300   \Numberstringnum{\expandafter\expandafter\expandafter
5301     \the\value{#1}}%
5302 }
```

`Numberstringnum` As above, but the argument is a count register or number.

```
5303 \newrobustcmd*{\Numberstringnum}[1]{%
5304   \new@ifnextchar[%
5305     {\@Number@string{#1}}%
5306     {\@Number@string{#1}[m]}}%
5307 }
```

`\@Number@string` Gender is specified as an optional argument at the end.

```
5308 \def\@Number@string#1[#2]{%
5309   {%
5310     \ifthenelse{\equal{#2}{f}}%
```

```

5311   {%
5312     \protect\@NumberstringF{#1}{\@fc@numstr}%
5313   }%
5314   {%
5315     \ifthenelse{\equal{#2}{n}}%
5316     {%
5317       \protect\@NumberstringN{#1}{\@fc@numstr}%
5318     }%
5319     {%
5320       \ifthenelse{\equal{#2}{m}}%
5321       {}%
5322       {%
5323         \PackageError{fmtcount}%
5324         {Invalid gender option ‘#2’}%
5325         {Available options are m, f or n}%
5326       }%
5327       \protect\@NumberstringM{#1}{\@fc@numstr}%
5328     }%
5329   }%
5330   \@fc@numstr
5331 }%
5332 }

```

`\storeNUMBERstring` Store upper case textual representation of number. The first argument is identifying name, the second argument is a counter.

```

5333 \newcommand{\storeNUMBERstring}[2]{%
5334   {%
5335     \toks0{\storeNUMBERstringnum{#1}}%
5336     \expandafter
5337     }\the\toks0\expandafter{\the\value{#2}}%
5338 }

```

`\NUMBERstringnum` As above, but the second argument is a count register or a number.

```

5339 \newcommand{\storeNUMBERstringnum}[2]{%
5340   \@ifnextchar[%
5341   {\@store@NUMBER@string{#1}{#2}}%
5342   {\@store@NUMBER@string{#1}{#2}[m]}%
5343 }

```

`\e@NUMBER@string` Gender is specified as an optional argument at the end.

```

5344 \def\@store@NUMBER@string#1#2[#3]{%
5345   \ifthenelse{\equal{#3}{f}}%
5346   {%
5347     \protect\@numberstringF{#2}{\@fc@numstr}%
5348   }%
5349   {%
5350     \ifthenelse{\equal{#3}{n}}%
5351     {%
5352       \protect\@numberstringN{#2}{\@fc@numstr}%

```

```

5353 }%
5354 {%
5355   \ifthenelse{\equal{#3}{m}}%
5356   {}%
5357   {%
5358     \PackageError{fmtcount}%
5359     {Invalid gender option ‘#3’}%
5360     {Available options are m or f}%
5361   }%
5362   \protect\@numberstringM{#2}{\@fc@numstr}%
5363 }%
5364 }%
5365 \expandafter\edef\csname @fcs@#1\endcsname{%
5366   \noexpand\MakeUppercase{\@fc@numstr}%
5367 }%
5368 }

```

`\NUMBERstring` Display upper case textual representation of a number. The argument must be a counter.

```

5369 \newcommand*{\NUMBERstring}[1]{%
5370   \NUMBERstringnum\expandafter\expandafter\expandafter
5371   \the\value{#1}}%
5372 }

```

`NUMBERstringnum` As above, but the argument is a count register or a number.

```

5373 \newrobustcmd*{\NUMBERstringnum}[1]{%
5374   \new@ifnextchar[%
5375   {\@NUMBER@string{#1}}%
5376   {\@NUMBER@string{#1}[m]}%
5377 }

```

`\@NUMBER@string` Gender is specified as an optional argument at the end.

```

5378 \def\@NUMBER@string#1[#2]{%
5379   {%
5380     \ifthenelse{\equal{#2}{f}}%
5381     {%
5382       \protect\@numberstringF{#1}{\@fc@numstr}%
5383     }%
5384     {%
5385       \ifthenelse{\equal{#2}{n}}%
5386       {%
5387         \protect\@numberstringN{#1}{\@fc@numstr}%
5388       }%
5389       {%
5390         \ifthenelse{\equal{#2}{m}}%
5391         {}%
5392         {%
5393           \PackageError{fmtcount}%
5394           {Invalid gender option ‘#2’}%
5395           {Available options are m, f or n}%

```

```

5396     }%
5397     \protect\@numberstringM{#1}{\@fc@numstr}%
5398     }%
5399     }%
5400     \MakeUppercase{\@fc@numstr}%
5401     }%
5402 }

```

`\binary` Number representations in other bases. Binary:

```

5403 \providecommand*\binary}[1]{%
5404   \@binary{\expandafter\expandafter\expandafter
5405     \the\value{#1}}%
5406 }

```

`\aaalph` Like `\alph`, but goes beyond 26. (a ... z aa ... zz ...)

```

5407 \providecommand*\aaalph}[1]{%
5408   \@aaalph{\expandafter\expandafter\expandafter
5409     \the\value{#1}}%
5410 }

```

`\AAAlph` As before, but upper case.

```

5411 \providecommand*\AAAlph}[1]{%
5412   \@AAAlph{\expandafter\expandafter\expandafter
5413     \the\value{#1}}%
5414 }

```

`\abalph` Like `\alph`, but goes beyond 26. (a ... z ab ... az ...)

```

5415 \providecommand*\abalph}[1]{%
5416   \@abalph{\expandafter\expandafter\expandafter
5417     \the\value{#1}}%
5418 }

```

`\ABAlph` As above, but upper case.

```

5419 \providecommand*\ABAlph}[1]{%
5420   \@ABAlph{\expandafter\expandafter\expandafter
5421     \the\value{#1}}%
5422 }

```

`\hexadecimal` Hexadecimal:

```

5423 \providecommand*\hexadecimal}[1]{%
5424   \@hexadecimal{\expandafter\expandafter\expandafter
5425     \the\value{#1}}%
5426 }

```

`\Hexadecimal` As above, but in upper case.

```

5427 \providecommand*\Hexadecimal}[1]{%
5428   \@Hexadecimal{\expandafter\expandafter\expandafter
5429     \the\value{#1}}%
5430 }

```

`\octal` Octal:

```
5431 \providecommand*{\octal}[1]{%
5432   \@octal{\expandafter\expandafter\expandafter
5433     \the\value{#1}}%
5434 }
```

`\decimal` Decimal:

```
5435 \providecommand*{\decimal}[1]{%
5436   \@decimal{\expandafter\expandafter\expandafter
5437     \the\value{#1}}%
5438 }
```

### 9.4.1 Multilanguage Definitions

Flag `\fc@language@detected` allows to stop scanning for multilingual mode trigger conditions. It is initialized to `false` as no such scanning as taken place yet.

```
5439 \newif\iffc@language@detected
5440 \fc@language@detectedfalse
```

`def@ultfmtcount` If multilingual support is provided, make `\@numberstring` etc use the correct language (if defined). Otherwise use English definitions. `\@setdef@ultfmtcount` sets the macros to use English.

```
5441 \def\@setdef@ultfmtcount{%
5442   \fc@language@detectedtrue
5443   \ifcsundef{@ordinalMenglish}{\FCloadlang{english}}{}%
5444   \def\@ordinalstringM{\@ordinalstringMenglish}%
5445   \let\@ordinalstringF=\@ordinalstringMenglish
5446   \let\@ordinalstringN=\@ordinalstringMenglish
5447   \def\@OrdinalstringM{\@OrdinalstringMenglish}%
5448   \let\@OrdinalstringF=\@OrdinalstringMenglish
5449   \let\@OrdinalstringN=\@OrdinalstringMenglish
5450   \def\@numberstringM{\@numberstringMenglish}%
5451   \let\@numberstringF=\@numberstringMenglish
5452   \let\@numberstringN=\@numberstringMenglish
5453   \def\@NumberstringM{\@NumberstringMenglish}%
5454   \let\@NumberstringF=\@NumberstringMenglish
5455   \let\@NumberstringN=\@NumberstringMenglish
5456   \def\@ordinalM{\@ordinalMenglish}%
5457   \let\@ordinalF=\@ordinalM
5458   \let\@ordinalN=\@ordinalM
5459   \let\fmtord\fc@orddef@ult
5460 }
```

`\fc@multiling` `\fc@multiling{<name>}{<gender>}`

```
5461 \newcommand*{\fc@multiling}[2]{%
5462   \ifcsundef{@#1#2\languagename}%
5463   {% try loading it
5464     \FCloadlang{\languagename}%
```

```

5465 }%
5466 {%
5467 }%
5468 \ifcsundef{@#1#2\languagename}%
5469 {%
5470   \PackageWarning{fmtcount}%
5471   {No support for \expandafter\protect\csname #1\endcsname\space for
5472   language '\languagename'}%
5473   \ifthenelse{\equal{\languagename}{\fc@mainlang}}%
5474   {%
5475     \FCloadlang{english}%
5476   }%
5477   {%
5478   }%
5479   \ifcsdef{@#1#2\fc@mainlang}%
5480   {%
5481     \csuse{@#1#2\fc@mainlang}%
5482   }%
5483   {%
5484     \PackageWarningNoLine{fmtcount}%
5485     {No languages loaded at all! Loading english definitions}%
5486     \FCloadlang{english}%
5487     \def\fc@mainlang{english}%
5488     \csuse{@#1#2english}%
5489   }%
5490 }%
5491 {%
5492   \csuse{@#1#2\languagename}%
5493 }%
5494 }

```

multiling@fmtcount This defines the number and ordinal string macros to use \languagename:

```

5495 \def\@set@multiling@fmtcount{%
5496   \fc@languagemode@detectedtrue

```

The masculine version of \numberstring:

```

5497   \def\@numberstringM{%
5498     \fc@multiling{numberstring}{M}%
5499   }%

```

The feminine version of \numberstring:

```

5500   \def\@numberstringF{%
5501     \fc@multiling{numberstring}{F}%
5502   }%

```

The neuter version of \numberstring:

```

5503   \def\@numberstringN{%
5504     \fc@multiling{numberstring}{N}%
5505   }%

```

The masculine version of \Numberstring:

```

5506 \def\@NumberstringM{%
5507   \fc@multiling{Numberstring}{M}%
5508 }%

  The feminine version of \Numberstring:
5509 \def\@NumberstringF{%
5510   \fc@multiling{Numberstring}{F}%
5511 }%

  The neuter version of \Numberstring:
5512 \def\@NumberstringN{%
5513   \fc@multiling{Numberstring}{N}%
5514 }%

  The masculine version of \ordinal:
5515 \def\@ordinalM{%
5516   \fc@multiling{ordinal}{M}%
5517 }%

  The feminine version of \ordinal:
5518 \def\@ordinalF{%
5519   \fc@multiling{ordinal}{F}%
5520 }%

  The neuter version of \ordinal:
5521 \def\@ordinalN{%
5522   \fc@multiling{ordinal}{N}%
5523 }%

  The masculine version of \ordinalstring:
5524 \def\@ordinalstringM{%
5525   \fc@multiling{ordinalstring}{M}%
5526 }%

  The feminine version of \ordinalstring:
5527 \def\@ordinalstringF{%
5528   \fc@multiling{ordinalstring}{F}%
5529 }%

  The neuter version of \ordinalstring:
5530 \def\@ordinalstringN{%
5531   \fc@multiling{ordinalstring}{N}%
5532 }%

  The masculine version of \Ordinalstring:
5533 \def\@OrdinalstringM{%
5534   \fc@multiling{Ordinalstring}{M}%
5535 }%

  The feminine version of \Ordinalstring:
5536 \def\@OrdinalstringF{%
5537   \fc@multiling{Ordinalstring}{F}%
5538 }%

```

The neuter version of \Ordinalstring:

```
5539 \def\@OrdinalstringN{%
5540   \fc@multiling{Ordinalstring}{N}%
5541 }%
```

Make \fmtord language dependent:

```
5542 \let\fmtord\fc@ord@multiling
5543 }
```

Check to see if babel, polyglossia, mlp, or ngerman packages have been loaded, and if yes set fmtcount in multiling. First we define some \fc@check@for@multiling macro to do such action where #1 is the package name, and #2 is a callback.

```
5544 \def\fc@check@for@multiling#1:#2\@nil{%
5545   \ifpackageloaded{#1}{%
5546     #2\@set@multiling@fmtcount
5547   }{}%
5548 }
```

Now we define \fc@loop@on@multiling@pkg as an iterator to scan whether any of babel, polyglossia, mlp, or ngerman packages has been loaded, and if so set multilingual mode.

```
5549 \def\fc@loop@on@multiling@pkg#1,{%
5550   \def\@tempb{#1}%
5551   \ifx\@tempb\@nnil
```

We have reached the end of the loop, so stop here.

```
5552   \let\fc@loop@on@multiling@pkg\@empty
5553   \else
```

Make the \@ifpackageloaded test and break the loop if it was positive.

```
5554     \fc@check@for@multiling#1\@nil
5555     \iffc@languagemode@detected
5556     \def\fc@loop@on@multiling@pkg##1\@nil,{}%
5557     \fi
5558   \fi
5559   \fc@loop@on@multiling@pkg
5560 }
```

Now, do the loop itself, we do this at beginning of document not to constrain the order of loading fmtcount and the multilingual package babel, polyglossia, etc.:

```
5561 \AtBeginDocument{%
5562   \fc@loop@on@multiling@pkg babel:,polyglossia:,ngerman:\FCloadlang{ngerman},\@nil,
```

In the case that no multilingual package (such as babel/polyglossia/ngerman) has been loaded, then we go to multiling if a language has been loaded by package option.

```
5563   \unless\iffc@languagemode@detected\iffmtcount@language@option
```

If the multilingual mode has not been yet activated, but a language option has been passed to fmtcount, we should go to multilingual mode. However, first of, we do some sanity check, as this may help the end user understand what is wrong: we check that macro \language is defined, and activate the multilingual mode only then, and otherwise fall back to default legacy mode.

```

5564 \ifcsundef{languagename}%
5565 {%
5566 \PackageWarning{fmtcount}{%
5567 '\protect{languagename}' is undefined, you should use a language package such as bab
5568 when loading a language via package option. Reverting to default language.
5569 }%
5570 \@setdef@ultfmtcount
5571 }{%
5572 \@set@multiling@fmtcount
5573

```

Now, some more checking, having activated multilingual mode after a language option has been passed to `fmtcount`, we check that the `fmtcount` language definitions corresponding to `\languagename` have been loaded, and otherwise fall `\languagename` back to the latest `fmtcount` language definition loaded.

```

5574 \@FC@iflangloaded{languagename}{-}{%

```

The current `\languagename` is not a `fmtcount` language that has been previously loaded. The correction is to have `\languagename` let to `\fc@mainlang`. Please note that, as `\iffmtcount@language@option` is true, we know that `fmtcount` has loaded some language.

```

5575 \PackageWarning{fmtcount}{%
5576 Setting '\protect{languagename}' to '\fc@mainlang'.\MessageBreak
5577 Reason is that '\protect{languagename}' was '\languagename',\MessageBreak
5578 but '\languagename' was not loaded by fmtcount,\MessageBreak
5579 whereas '\fc@mainlang' was the last language loaded by fmtcount ;
5580 }%
5581 \let{languagename}\fc@mainlang
5582 }%
5583 }%
5584 \else
5585 \@setdef@ultfmtcount
5586 \fi\fi
5587 }

```

```

5588 \AtBeginDocument{%
5589 \ifcsundef{FBsupR}{\let\fc@textsuperscript\textsuperscript}{\let\fc@textsuperscript\fup}%
5590 }

```

Backwards compatibility:

```

5591 \let\@ordinal=\@ordinalM
5592 \let\@ordinalstring=\@ordinalstringM
5593 \let\@Ordinalstring=\@OrdinalstringM
5594 \let\@numberstring=\@numberstringM
5595 \let\@Numberstring=\@NumberstringM

```