fmtcount.sty: Displaying the Values of LaTeX Counters

Nicola L.C. Talbot   Vincent Belaïche

www.dickimaw-books.com

2017-09-16 (version 3.04)

Contents

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: 3rd. 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 incompatiblity, if you want to use the fmtcount package with the memoir class you should use

\FCordinal

```
\FCordinal
```

2

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$^{\text{rd}}$! whereas \ordinal{section}[m] ! will produce: 3$^{\text{rd}}$!

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$^{\text{rd}}$.

\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[1].

\numberstringnum

```
\numberstringnum{⟨n⟩}[⟨gender⟩]
```

\Numberstringnum

```
\Numberstringnum{⟨n⟩}[⟨gender⟩]
```

\NUMBERstringnum

```
\NUMBERstringnum{⟨n⟩}[⟨gender⟩]
```

Theses 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.

---

[1]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⟩]`

rdinalstringnum	`\Ordinalstringnum{⟨n⟩}[⟨gender⟩]`

RDINALstringnum	`\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⟩]`

\storeordinalstring	`\storeordinalstring{⟨label⟩}{⟨counter⟩}[⟨gender⟩]`
\storeOrdinalstring	`\storeOrdinalstring{⟨label⟩}{⟨counter⟩}[⟨gender⟩]`
\storeORDINALstring	`\storeORDINALstring{⟨label⟩}{⟨counter⟩}[⟨gender⟩]`
\storenumberstring	`\storenumberstring{⟨label⟩}{⟨counter⟩}[⟨gender⟩]`
\storeNumberstring	`\storeNumberstring{⟨label⟩}{⟨counter⟩}[⟨gender⟩]`
\storeNUMBERstring	`\storeNUMBERstring{⟨label⟩}{⟨counter⟩}[⟨gender⟩]`
\storeordinalnum	`\storeordinalnum{⟨label⟩}{⟨number⟩}[⟨gender⟩]`
\storeordinalstringnum	`\storeordinalstring{⟨label⟩}{⟨number⟩}[⟨gender⟩]`
\storeOrdinalstringnum	`\storeOrdinalstringnum{⟨label⟩}{⟨number⟩}[⟨gender⟩]`
\storeORDINALstringnum	`\storeORDINALstringnum{⟨label⟩}{⟨number⟩}[⟨gender⟩]`
\storenumberstringnum	`\storenumberstring{⟨label⟩}{⟨number⟩}[⟨gender⟩]`
\storeNumberstringnum	`\storeNumberstring{⟨label⟩}{⟨number⟩}[⟨gender⟩]`
\storeNUMBERstringnum	`\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

5

\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 pro-
duce: 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 num-
ber 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.

\AAAlph

```
\AAAlph{⟨counter⟩}
```

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

\aaalphnum

```
\aaalphnum{⟨n⟩}
```

\AAAlphnum

```
\AAAlphnum{⟨n⟩}
```

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

countsetoptions | `\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.[2] Italian support was added in version 1.31.[3]

To ensure the language definitions are loaded correctly for document dialects, use

`\FCloadlang` | `\FCloadlang{⟨dialect⟩}`

in the preamble. The ⟨*dialect*⟩ should match the options passed to babel or polyglossia. fmtcount currently supports the following ⟨*dialect*⟩: `english`, `UKenglish`, `british`, `USenglish`, `american`, `spanish`, `portuges`, `french`, `frenchb`, `francais`, `german`, `germanb`, `ngerman`, `ngermanb`, and `italian`. If you don't use this, fmtcount will attempt to detect the required dialects, but this isn't guaranteed to work.

The commands `\ordinal`, `\ordinalstring` and `\numberstring` (and their variants) will be formatted in the currently selected language. If the current language hasn't been loaded (via `\FCloadlang` above) and fmtcount detects a definition file for that language it will attempt to load it, but this isn't robust and may cause problems, so it's best to use `\FCloadlang`.

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.

---

[2]Thanks to K. H. Fricke for supplying the information.
[3]Thanks to Edoardo Pasca for supplying the information.

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

```
\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[4] 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

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.

```
\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`.

---

[4]see § 3

9

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[5] pour 80, et nonante pour 90

Il est à noter que la variante `belgian` est parfaitement correcte pour les francophones français[6], 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$^{\text{e}}$ (par défaut), ou
`false` pour produire des ordinaux de la forme 2$^{\text{ème}}$

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

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

mil plural	`\fmtcountsetoptions{french={mil plural=⟨french plural control⟩}}`

n-illion plural	`\fmtcountsetoptions{french={n-illion plural=⟨french plural control⟩}}`

-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`

---

[5]voir Octante et huitante sur le site d'Alain Lassine
[6]je précise que l'auteur de ces lignes est français

`plural`, permettent de contrôler très finement l'accord en nombre des mots respectivement vingt, cent, mil, et des mots de la forme ⟨*n*⟩illion et ⟨*n*⟩illiard, où ⟨*n*⟩ 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 on les simplifie. Le paquetage `fmtcount` est déjà prêt à cette éventualité.

Le paramètre ⟨*french plural control*⟩ 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 `traditional` et `reformed` 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,
`traditional o`	pareil que `traditional` mais dans le cas des numéraux cardinaux, lorsqu'ils ont une valeur ordinale,
`reformed o`	pareil que `reformed` mais dans le cas des numéraux cardinaux, lorsqu'ils ont une valeur ordinale, de même que précédemment `reformed o` et `traditional o` ont exactement le même effet,

always	pour marquer toujours le pluriel, ceci n'est correct que pour « mil » vis à vis des règles en vigueur,
never	pour ne jamais marquer le pluriel, ceci est incorrect vis à vis des règles d'orthographe en vigueur,
multiple	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 ⟨*n*⟩illion et ⟨*n*⟩illiard lorsque le nombre a une valeur cardinale,
multiple g-last	pour marquer le pluriel lorsque le nombre considéré est multiplié par au moins 2 est est *globalement* 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,
multiple l-last	pour marquer le pluriel lorsque le nombre considéré est multiplié par au moins 2 et est *localement* en dernière position, où "localement" siginifie qu'on considère seulement la portion du nombre qui multiplie soit l'unité, soit un ⟨*n*⟩illion ou un ⟨*n*⟩illiard ; ceci est la convention en vigueur pour le pluriel de "vingt" et de "cent" lorsque le nombre formaté a une valeur cardinale,
multiple lng-last	pour marquer le pluriel lorsque le nombre considéré est multiplié par au moins 2 et est *localement* mais ***non globablement*** en dernière position, où "localement" et *globablement* on la même siginification que pour les options `multiple g-last` et `multiple l-last` ; ceci est la convention en vigueur pour le pluriel de "vingt" et de "cent" lorsque le nombre formaté a une valeur ordinale,
multiple ng-last	pour marquer le pluriel lorsque le nombre considéré est multiplié par au moins 2, et ***n'***est pas ***globalement*** en dernière position, où "globalement" a la même signification que pour l'option `multiple g-last` ; ceci est la règle que j'infère être en vigueur pour les nombres de la forme ⟨*n*⟩illion et ⟨*n*⟩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 :

⟨*x*⟩ dans "⟨*x*⟩ plural"	traditional	reformed	traditional o	reformed o
vingt	multiple l-last		multiple lng-last	
cent				
mil	always			
n-illion	multiple		multiple ng-last	
n-illiard				

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 $n \mod 10 = 1$, 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 ⟨*dash or space*⟩ 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 pise 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 ⟨*scale*⟩ 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 ⟨*n*⟩illion où ⟨*n*⟩ est remplacé par "bi" pour 2, "tri" pour 3, etc. et $10^{6\times n+3}$ donne un ⟨*n*⟩illiard avec la même convention pour ⟨*n*⟩. L'option long est correcte en Europe, par contre j'ignore l'usage au Québec.
short $10^{6\times n}$ donne un ⟨*n*⟩illion où ⟨*n*⟩ 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=⟨n-illiard upto⟩}}
```

Cette option n'a de sens que si scale vaut long. Certaines personnes préfèrent dire "mille ⟨*n*⟩illions" qu'un "⟨*n*⟩illiard". Mettre l'option n-illiard upto à :

infinity   pour que $10^{6\times n+3}$ donne ⟨*n*⟩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 ⟨*n*⟩illions" lorsque $n > k$, et donne "⟨*n*⟩illiard" sinon

mil plural mark	`\fmtcountsetoptions{french={mil plural mark=⟨any text⟩}}`

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{⟨counter⟩}[⟨prefix options⟩]`

latinnumeralstringnum	`\latinnumeralstringnum{⟨number⟩}[⟨prefix options⟩]`

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-british.def

British definitions

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

Load fc-english.def, if not already loaded

```
19 \FCloadlang{english}%
```

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

```
20 \global\let\@ordinalMbritish\@ordinalMenglish
21 \global\let\@ordinalFbritish\@ordinalMenglish
22 \global\let\@ordinalNbritish\@ordinalMenglish
23 \global\let\@numberstringMbritish\@numberstringMenglish
24 \global\let\@numberstringFbritish\@numberstringMenglish
25 \global\let\@numberstringNbritish\@numberstringMenglish
26 \global\let\@NumberstringMbritish\@NumberstringMenglish
27 \global\let\@NumberstringFbritish\@NumberstringMenglish
28 \global\let\@NumberstringNbritish\@NumberstringMenglish
29 \global\let\@ordinalstringMbritish\@ordinalstringMenglish
30 \global\let\@ordinalstringFbritish\@ordinalstringMenglish
31 \global\let\@ordinalstringNbritish\@ordinalstringMenglish
32 \global\let\@OrdinalstringMbritish\@OrdinalstringMenglish
33 \global\let\@OrdinalstringFbritish\@OrdinalstringMenglish
34 \global\let\@OrdinalstringNbritish\@OrdinalstringMenglish
```

9.1.3 fc-english.def

English definitions

```
35 \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.

```
36 \newcommand*\@ordinalMenglish[2]{%
37 \def\@fc@ord{}%
38 \@orgargctr=#1\relax
39 \@ordinalctr=#1%
40 \@FCmodulo{\@ordinalctr}{100}%
41 \ifnum\@ordinalctr=11\relax
42   \def\@fc@ord{th}%
43 \else
44   \ifnum\@ordinalctr=12\relax
45     \def\@fc@ord{th}%
46   \else
47     \ifnum\@ordinalctr=13\relax
48       \def\@fc@ord{th}%
49     \else
50       \@FCmodulo{\@ordinalctr}{10}%
51       \ifcase\@ordinalctr
52         \def\@fc@ord{th}%      case 0
53         \or \def\@fc@ord{st}%  case 1
54         \or \def\@fc@ord{nd}%  case 2
55         \or \def\@fc@ord{rd}%  case 3
56       \else
57         \def\@fc@ord{th}%      default case
58       \fi
59     \fi
60   \fi
```

16

```
61 \fi
62 \edef#2{\number#1\relax\noexpand\fmtord{\@fc@ord}}%
63 }%
64 \global\let\@ordinalMenglish\@ordinalMenglish
```

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

```
65 \global\let\@ordinalFenglish=\@ordinalMenglish
66 \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.

```
67 \newcommand*\@@unitstringenglish[1]{%
68   \ifcase#1\relax
69     zero%
70     \or one%
71     \or two%
72     \or three%
73     \or four%
74     \or five%
75     \or six%
76     \or seven%
77     \or eight%
78     \or nine%
79 \fi
80 }%
81 \global\let\@@unitstringenglish\@@unitstringenglish
```

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

```
82 \newcommand*\@@tenstringenglish[1]{%
83   \ifcase#1\relax
84     \or ten%
85     \or twenty%
86     \or thirty%
87     \or forty%
88     \or fifty%
89     \or sixty%
90     \or seventy%
91     \or eighty%
92     \or ninety%
93   \fi
94 }%
95 \global\let\@@tenstringenglish\@@tenstringenglish
```

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

```
96 \newcommand*\@@teenstringenglish[1]{%
97   \ifcase#1\relax
98     ten%
99     \or eleven%
100    \or twelve%
```

```
101     \or thirteen%
102     \or fourteen%
103     \or fifteen%
104     \or sixteen%
105     \or seventeen%
106     \or eighteen%
107     \or nineteen%
108   \fi
109 }%
110 \global\let\@@teenstringenglish\@@teenstringenglish
```

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

```
111 \newcommand*\@@Unitstringenglish[1]{%
112   \ifcase#1\relax
113     Zero%
114     \or One%
115     \or Two%
116     \or Three%
117     \or Four%
118     \or Five%
119     \or Six%
120     \or Seven%
121     \or Eight%
122     \or Nine%
123   \fi
124 }%
125 \global\let\@@Unitstringenglish\@@Unitstringenglish
```

The tens:

```
126 \newcommand*\@@Tenstringenglish[1]{%
127   \ifcase#1\relax
128     \or Ten%
129     \or Twenty%
130     \or Thirty%
131     \or Forty%
132     \or Fifty%
133     \or Sixty%
134     \or Seventy%
135     \or Eighty%
136     \or Ninety%
137   \fi
138 }%
139 \global\let\@@Tenstringenglish\@@Tenstringenglish
```

The teens:

```
140 \newcommand*\@@Teenstringenglish[1]{%
141   \ifcase#1\relax
142     Ten%
143     \or Eleven%
144     \or Twelve%
145     \or Thirteen%
```

```
146    \or Fourteen%
147    \or Fifteen%
148    \or Sixteen%
149    \or Seventeen%
150    \or Eighteen%
151    \or Nineteen%
152  \fi
153 }%
154 \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.)

```
155 \newcommand*\@@numberstringenglish[2]{%
156 \ifnum#1>99999
157 \PackageError{fmtcount}{Out of range}%
158 {This macro only works for values less than 100000}%
159 \else
160 \ifnum#1<0
161 \PackageError{fmtcount}{Negative numbers not permitted}%
162 {This macro does not work for negative numbers, however
163 you can try typing "minus" first, and then pass the modulus of
164 this number}%
165 \fi
166 \fi
167 \def#2{}%
168 \@strctr=#1\relax \divide\@strctr by 1000\relax
169 \ifnum\@strctr>9
170   \divide\@strctr by 10
171   \ifnum\@strctr>1\relax
172     \let\@@fc@numstr#2\relax
173     \edef#2{\@@fc@numstr\@tenstring{\@strctr}}%
174     \@strctr=#1 \divide\@strctr by 1000\relax
175     \@FCmodulo{\@strctr}{10}%
176     \ifnum\@strctr>0\relax
177       \let\@@fc@numstr#2\relax
178       \edef#2{\@@fc@numstr-\@unitstring{\@strctr}}%
179     \fi
180   \else
181     \@strctr=#1\relax
182     \divide\@strctr by 1000\relax
183     \@FCmodulo{\@strctr}{10}%
184     \let\@@fc@numstr#2\relax
185     \edef#2{\@@fc@numstr\@teenstring{\@strctr}}%
186   \fi
187   \let\@@fc@numstr#2\relax
188   \edef#2{\@@fc@numstr\ \@thousand}%
189 \else
190   \ifnum\@strctr>0\relax
191     \let\@@fc@numstr#2\relax
```

```
192       \edef#2{\@@fc@numstr\@unitstring{\@strctr}\ \@thousand}%
193   \fi
194 \fi
195 \@strctr=#1\relax \@FCmodulo{\@strctr}{1000}%
196 \divide\@strctr by 100
197 \ifnum\@strctr>0\relax
198    \ifnum#1>1000\relax
199       \let\@@fc@numstr#2\relax
200       \edef#2{\@@fc@numstr\ }%
201    \fi
202    \let\@@fc@numstr#2\relax
203    \edef#2{\@@fc@numstr\@unitstring{\@strctr}\ \@hundred}%
204 \fi
205 \@strctr=#1\relax \@FCmodulo{\@strctr}{100}%
206 \ifnum#1>100\relax
207   \ifnum\@strctr>0\relax
208      \let\@@fc@numstr#2\relax
209      \edef#2{\@@fc@numstr\ \@andname\ }%
210   \fi
211 \fi
212 \ifnum\@strctr>19\relax
213   \divide\@strctr by 10\relax
214   \let\@@fc@numstr#2\relax
215   \edef#2{\@@fc@numstr\@tenstring{\@strctr}}%
216   \@strctr=#1\relax \@FCmodulo{\@strctr}{10}%
217   \ifnum\@strctr>0\relax
218      \let\@@fc@numstr#2\relax
219      \edef#2{\@@fc@numstr-\@unitstring{\@strctr}}%
220   \fi
221 \else
222   \ifnum\@strctr<10\relax
223     \ifnum\@strctr=0\relax
224        \ifnum#1<100\relax
225           \let\@@fc@numstr#2\relax
226           \edef#2{\@@fc@numstr\@unitstring{\@strctr}}%
227        \fi
228     \else
229       \let\@@fc@numstr#2\relax
230       \edef#2{\@@fc@numstr\@unitstring{\@strctr}}%
231     \fi
232   \else
233     \@FCmodulo{\@strctr}{10}%
234     \let\@@fc@numstr#2\relax
235     \edef#2{\@@fc@numstr\@teenstring{\@strctr}}%
236   \fi
237 \fi
238 }%
239 \global\let\@@numberstringenglish\@@numberstringenglish
```

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

```
240 \newcommand*{\@numberstringMenglish}[2]{%
241   \let\@unitstring=\@@unitstringenglish
242   \let\@teenstring=\@@teenstringenglish
243   \let\@tenstring=\@@tenstringenglish
244   \def\@hundred{hundred}\def\@thousand{thousand}%
245   \def\@andname{and}%
246   \@@numberstringenglish{#1}{#2}%
247 }%
248 \global\let\@numberstringMenglish\@numberstringMenglish
```

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

```
249 \global\let\@numberstringFenglish=\@numberstringMenglish
250 \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.

```
251 \newcommand*\@NumberstringMenglish[2]{%
252   \let\@unitstring=\@@Unitstringenglish
253   \let\@teenstring=\@@Teenstringenglish
254   \let\@tenstring=\@@Tenstringenglish
255   \def\@hundred{Hundred}\def\@thousand{Thousand}%
256   \def\@andname{and}%
257   \@@numberstringenglish{#1}{#2}%
258 }%
259 \global\let\@NumberstringMenglish\@NumberstringMenglish
```

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

```
260 \global\let\@NumberstringFenglish=\@NumberstringMenglish
261 \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:

```
262 \newcommand*\@@unitthstringenglish[1]{%
263   \ifcase#1\relax
264     zeroth%
265     \or first%
266     \or second%
267     \or third%
268     \or fourth%
269     \or fifth%
270     \or sixth%
271     \or seventh%
272     \or eighth%
273     \or ninth%
274   \fi
275 }%
276 \global\let\@@unitthstringenglish\@@unitthstringenglish
```

Next the tens:

```
277 \newcommand*\@@tenthstringenglish[1]{%
278   \ifcase#1\relax
```

```
279    \or tenth%
280    \or twentieth%
281    \or thirtieth%
282    \or fortieth%
283    \or fiftieth%
284    \or sixtieth%
285    \or seventieth%
286    \or eightieth%
287    \or ninetieth%
288  \fi
289 }%
290 \global\let\@@tenthstringenglish\@@tenthstringenglish
```

The teens:

```
291 \newcommand*\@@teenthstringenglish[1]{%
292   \ifcase#1\relax
293     tenth%
294    \or eleventh%
295    \or twelfth%
296    \or thirteenth%
297    \or fourteenth%
298    \or fifteenth%
299    \or sixteenth%
300    \or seventeenth%
301    \or eighteenth%
302    \or nineteenth%
303   \fi
304 }%
305 \global\let\@@teenthstringenglish\@@teenthstringenglish
```

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

```
306 \newcommand*\@@Unitthstringenglish[1]{%
307   \ifcase#1\relax
308     Zeroth%
309    \or First%
310    \or Second%
311    \or Third%
312    \or Fourth%
313    \or Fifth%
314    \or Sixth%
315    \or Seventh%
316    \or Eighth%
317    \or Ninth%
318   \fi
319 }%
320 \global\let\@@Unitthstringenglish\@@Unitthstringenglish
```

The tens:

```
321 \newcommand*\@@Tenthstringenglish[1]{%
322   \ifcase#1\relax
323    \or Tenth%
```

```
324     \or Twentieth%
325     \or Thirtieth%
326     \or Fortieth%
327     \or Fiftieth%
328     \or Sixtieth%
329     \or Seventieth%
330     \or Eightieth%
331     \or Ninetieth%
332   \fi
333 }%
334 \global\let\@@Tenthstringenglish\@@Tenthstringenglish
```

The teens:

```
335 \newcommand*\@@Teenthstringenglish[1]{%
336   \ifcase#1\relax
337     Tenth%
338     \or Eleventh%
339     \or Twelfth%
340     \or Thirteenth%
341     \or Fourteenth%
342     \or Fifteenth%
343     \or Sixteenth%
344     \or Seventeenth%
345     \or Eighteenth%
346     \or Nineteenth%
347   \fi
348 }%
349 \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.

```
350 \newcommand*\@@ordinalstringenglish[2]{%
351 \@strctr=#1\relax
352 \ifnum#1>99999
353 \PackageError{fmtcount}{Out of range}%
354 {This macro only works for values less than 100000 (value given: \number\@strctr)}%
355 \else
356 \ifnum#1<0
357 \PackageError{fmtcount}{Negative numbers not permitted}%
358 {This macro does not work for negative numbers, however
359 you can try typing "minus" first, and then pass the modulus of
360 this number}%
361 \fi
362 \def#2{}%
363 \fi
364 \@strctr=#1\relax \divide\@strctr by 1000\relax
365 \ifnum\@strctr>9\relax
```

#1 is greater or equal to 10000

```
366    \divide\@strctr by 10
367    \ifnum\@strctr>1\relax
368      \let\@@fc@ordstr#2\relax
369      \edef#2{\@@fc@ordstr\@tenstring{\@strctr}}%
370      \@strctr=#1\relax
371      \divide\@strctr by 1000\relax
372      \@FCmodulo{\@strctr}{10}%
373      \ifnum\@strctr>0\relax
374        \let\@@fc@ordstr#2\relax
375        \edef#2{\@@fc@ordstr-\@unitstring{\@strctr}}%
376      \fi
377    \else
378      \@strctr=#1\relax \divide\@strctr by 1000\relax
379      \@FCmodulo{\@strctr}{10}%
380      \let\@@fc@ordstr#2\relax
381      \edef#2{\@@fc@ordstr\@teenstring{\@strctr}}%
382    \fi
383    \@strctr=#1\relax \@FCmodulo{\@strctr}{1000}%
384    \ifnum\@strctr=0\relax
385      \let\@@fc@ordstr#2\relax
386      \edef#2{\@@fc@ordstr\ \@thousandth}%
387    \else
388      \let\@@fc@ordstr#2\relax
389      \edef#2{\@@fc@ordstr\ \@thousand}%
390    \fi
391  \else
392    \ifnum\@strctr>0\relax
393      \let\@@fc@ordstr#2\relax
394      \edef#2{\@@fc@ordstr\@unitstring{\@strctr}}%
395      \@strctr=#1\relax \@FCmodulo{\@strctr}{1000}%
396      \let\@@fc@ordstr#2\relax
397      \ifnum\@strctr=0\relax
398        \edef#2{\@@fc@ordstr\ \@thousandth}%
399      \else
400        \edef#2{\@@fc@ordstr\ \@thousand}%
401      \fi
402    \fi
403  \fi
404  \@strctr=#1\relax \@FCmodulo{\@strctr}{1000}%
405  \divide\@strctr by 100
406  \ifnum\@strctr>0\relax
407    \ifnum#1>1000\relax
408      \let\@@fc@ordstr#2\relax
409      \edef#2{\@@fc@ordstr\ }%
410    \fi
411    \let\@@fc@ordstr#2\relax
412    \edef#2{\@@fc@ordstr\@unitstring{\@strctr}}%
413    \@strctr=#1\relax \@FCmodulo{\@strctr}{100}%
414    \let\@@fc@ordstr#2\relax
```

24

```
415  \ifnum\@strctr=0\relax
416     \edef#2{\@@fc@ordstr\ \@hundredth}%
417  \else
418     \edef#2{\@@fc@ordstr\ \@hundred}%
419  \fi
420 \fi
421 \@strctr=#1\relax \@FCmodulo{\@strctr}{100}%
422 \ifnum#1>100\relax
423  \ifnum\@strctr>0\relax
424     \let\@@fc@ordstr#2\relax
425     \edef#2{\@@fc@ordstr\ \@andname\ }%
426  \fi
427 \fi
428 \ifnum\@strctr>19\relax
429  \@tmpstrctr=\@strctr
430  \divide\@strctr by 10\relax
431  \@FCmodulo{\@tmpstrctr}{10}%
432  \let\@@fc@ordstr#2\relax
433  \ifnum\@tmpstrctr=0\relax
434     \edef#2{\@@fc@ordstr\@tenthstring{\@strctr}}%
435  \else
436     \edef#2{\@@fc@ordstr\@tenstring{\@strctr}}%
437  \fi
438  \@strctr=#1\relax \@FCmodulo{\@strctr}{10}%
439  \ifnum\@strctr>0\relax
440     \let\@@fc@ordstr#2\relax
441     \edef#2{\@@fc@ordstr-\@unitthstring{\@strctr}}%
442  \fi
443 \else
444  \ifnum\@strctr<10\relax
445     \ifnum\@strctr=0\relax
446        \ifnum#1<100\relax
447           \let\@@fc@ordstr#2\relax
448           \edef#2{\@@fc@ordstr\@unitthstring{\@strctr}}%
449        \fi
450     \else
451        \let\@@fc@ordstr#2\relax
452        \edef#2{\@@fc@ordstr\@unitthstring{\@strctr}}%
453     \fi
454  \else
455     \@FCmodulo{\@strctr}{10}%
456     \let\@@fc@ordstr#2\relax
457     \edef#2{\@@fc@ordstr\@teenthstring{\@strctr}}%
458  \fi
459 \fi
460 }%
461 \global\let\@@ordinalstringenglish\@@ordinalstringenglish
```

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

resulting text is stored.

```
462 \newcommand*{\@ordinalstringMenglish}[2]{%
463   \let\@unitthstring=\@@unitthstringenglish
464   \let\@teenthstring=\@@teenthstringenglish
465   \let\@tenthstring=\@@tenthstringenglish
466   \let\@unitstring=\@@unitstringenglish
467   \let\@teenstring=\@@teenstringenglish
468   \let\@tenstring=\@@tenstringenglish
469   \def\@andname{and}%
470   \def\@hundred{hundred}\def\@thousand{thousand}%
471   \def\@hundredth{hundredth}\def\@thousandth{thousandth}%
472   \@@ordinalstringenglish{#1}{#2}%
473 }%
474 \global\let\@ordinalstringMenglish\@ordinalstringMenglish
```

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

```
475 \global\let\@ordinalstringFenglish=\@ordinalstringMenglish
476 \global\let\@ordinalstringNenglish=\@ordinalstringMenglish
```

First letter of each word in upper case:

```
477 \newcommand*{\@OrdinalstringMenglish}[2]{%
478   \let\@unitthstring=\@@Unitthstringenglish
479   \let\@teenthstring=\@@Teenthstringenglish
480   \let\@tenthstring=\@@Tenthstringenglish
481   \let\@unitstring=\@@Unitstringenglish
482   \let\@teenstring=\@@Teenstringenglish
483   \let\@tenstring=\@@Tenstringenglish
484   \def\@andname{and}%
485   \def\@hundred{Hundred}\def\@thousand{Thousand}%
486   \def\@hundredth{Hundredth}\def\@thousandth{Thousandth}%
487   \@@ordinalstringenglish{#1}{#2}%
488 }%
489 \global\let\@OrdinalstringMenglish\@OrdinalstringMenglish
```

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

```
490 \global\let\@OrdinalstringFenglish=\@OrdinalstringMenglish
491 \global\let\@OrdinalstringNenglish=\@OrdinalstringMenglish
```

9.1.4 fc-francais.def

```
492 \ProvidesFCLanguage{francais}[2013/08/17]%
493 \FCloadlang{french}%
```

Set `francais` to be equivalent to `french`.

```
494 \global\let\@ordinalMfrancais=\@ordinalMfrench
495 \global\let\@ordinalFfrancais=\@ordinalFfrench
496 \global\let\@ordinalNfrancais=\@ordinalNfrench
497 \global\let\@numberstringMfrancais=\@numberstringMfrench
498 \global\let\@numberstringFfrancais=\@numberstringFfrench
499 \global\let\@numberstringNfrancais=\@numberstringNfrench
500 \global\let\@NumberstringMfrancais=\@NumberstringMfrench
```

```
501 \global\let\@NumberstringFfrancais=\@NumberstringFfrench
502 \global\let\@NumberstringNfrancais=\@NumberstringNfrench
503 \global\let\@ordinalstringMfrancais=\@ordinalstringMfrench
504 \global\let\@ordinalstringFfrancais=\@ordinalstringFfrench
505 \global\let\@ordinalstringNfrancais=\@ordinalstringNfrench
506 \global\let\@OrdinalstringMfrancais=\@OrdinalstringMfrench
507 \global\let\@OrdinalstringFfrancais=\@OrdinalstringFfrench
508 \global\let\@OrdinalstringNfrancais=\@OrdinalstringNfrench
```

9.1.5 fc-french.def

Definitions for French.

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

Package fcprefix is needed to format the prefix ⟨*n*⟩ in ⟨*n*⟩illion or ⟨*n*⟩illiard. Big numbers were developed based on reference: 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.

```
510 \ifcsundef{fc@gl@let}{\global\let\fc@gl@let\let}{\PackageError{fmtcount}{Command already define
511 \protect\fc@gl@let\space already defined.}}
512 \ifcsundef{fc@gl@def}{\global\let\fc@gl@def\def}{\PackageError{fmtcount}{Command already define
513 \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'.

```
514 \gdef\fc@french@set@plural#1#2#3#4#5#6{%
515   \ifthenelse{\equal{#2}{reformed}}{%
516     \expandafter\fc@gl@def\csname fc@frenchoptions@#1@plural\endcsname{#3}%
517   }{%
518     \ifthenelse{\equal{#2}{traditional}}{%
519       \expandafter\fc@gl@def\csname fc@frenchoptions@#1@plural\endcsname{#4}%
520     }{%
521       \ifthenelse{\equal{#2}{reformed o}}{%
522         \expandafter\fc@gl@def\csname fc@frenchoptions@#1@plural\endcsname{#5}%
523       }{%
524         \ifthenelse{\equal{#2}{traditional o}}{%
525           \expandafter\fc@gl@def\csname fc@frenchoptions@#1@plural\endcsname{#6}%
526         }{%
527           \ifthenelse{\equal{#2}{always}}{%
528             \expandafter\fc@gl@def\csname fc@frenchoptions@#1@plural\endcsname{0}%
```

27

```
529            }{%
530          \ifthenelse{\equal{#2}{never}}{%
531            \expandafter\fc@gl@def\csname fc@frenchoptions@#1@plural\endcsname{1}%
532          }{%
533            \ifthenelse{\equal{#2}{multiple}}{%
534              \expandafter\fc@gl@def\csname fc@frenchoptions@#1@plural\endcsname{2}%
535            }{%
536              \ifthenelse{\equal{#2}{multiple g-last}}{%
537                \expandafter\fc@gl@def\csname fc@frenchoptions@#1@plural\endcsname{3}%
538              }{%
539                \ifthenelse{\equal{#2}{multiple l-last}}{%
540                  \expandafter\fc@gl@def\csname fc@frenchoptions@#1@plural\endcsname{4}%
541                }{%
542                  \ifthenelse{\equal{#2}{multiple lng-last}}{%
543                    \expandafter\fc@gl@def\csname fc@frenchoptions@#1@plural\endcsname{5}%
544                  }{%
545                    \ifthenelse{\equal{#2}{multiple ng-last}}{%
546                      \expandafter\fc@gl@def\csname fc@frenchoptions@#1@plural\endcsname{6}%
547                    }{%
548                      \PackageError{fmtcount}{Unexpected argument}{%
549                        '#2' was unexpected: french option '#1 plural' expects 'reformed', 't
550                        'reformed o', 'traditional o', 'always', 'never', 'multiple', 'multip
551                        'multiple l-last', 'multiple lng-last', or 'multiple ng-last'.%
552                      }}}}}}}}}}}%
```

Now a shorthand `\@tempa` is defined just to define all the options controlling plural mark. This shorthand takes into account that 'reformed' and 'traditional' have the same effect, and so do 'reformed o' and 'traditional o'.

```
553 \def\@tempa#1#2#3{%
554   \define@key{fcfrench}{#1 plural}[reformed]{%
555     \fc@french@set@plural{#1}{##1}{#2}{#2}{#3}{#3}%
556   }%
```

Macro `\@tempb` takes a macro as argument, and makes its current definition global. Like here it is useful when the macro name contains non-letters, and we have to resort to the `\csname...\endcsname` construct.

```
557   \expandafter\@tempb\csname KV@fcfrench@#1 plural\endcsname
558 }%
559 \def\@tempb#1{%
560   \global\let#1#1
561 }%
562 \@tempa{vingt}{4}{5}
563 \@tempa{cent}{4}{5}
564 \@tempa{mil}{0}{0}
565 \@tempa{n-illion}{2}{6}
566 \@tempa{n-illiard}{2}{6}
```

For option 'all plural' we cannot use the `\@tempa` shorthand, because 'all plural' is just a multiplexer.

```
567 \define@key{fcfrench}{all plural}[reformed]{%
```

```
568   \csname KV@fcfrench@vingt plural\endcsname{#1}%
569   \csname KV@fcfrench@cent plural\endcsname{#1}%
570   \csname KV@fcfrench@mil plural\endcsname{#1}%
571   \csname KV@fcfrench@n-illion plural\endcsname{#1}%
572   \csname KV@fcfrench@n-illiard plural\endcsname{#1}%
573 }%
574 \expandafter\@tempb\csname KV@fcfrench@all plural\endcsname
```

Now options 'dash or space', we have three possible key values:

traditional	use dash for numbers below 100, except when 'et' is used, and space otherwise
reformed	reform of 1990, use dash except with million & milliard, and suchlikes, i.e. $\langle n\rangle$illion and $\langle n\rangle$illiard,
always	always use dashes to separate all words

```
575 \define@key{fcfrench}{dash or space}[reformed]{%
576   \ifthenelse{\equal{#1}{traditional}}{%
577     \let\fc@frenchoptions@supermillion@dos\space%
578     \let\fc@frenchoptions@submillion@dos\space
579   }{%
580     \ifthenelse{\equal{#1}{reformed}\or\equal{#1}{1990}}{%
581       \let\fc@frenchoptions@supermillion@dos\space
582       \def\fc@frenchoptions@submillion@dos{-}%
583     }{%
584       \ifthenelse{\equal{#1}{always}}{%
585         \def\fc@frenchoptions@supermillion@dos{-}%
586         \def\fc@frenchoptions@submillion@dos{-}%
587       }{%
588         \PackageError{fmtcount}{Unexpected argument}{%
589           French option 'dash or space' expects 'always', 'reformed' or 'traditional'
590         }
591       }%
592     }%
593   }%
594 }%
```

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

```
595 \define@key{fcfrench}{scale}[recursive]{%
596   \ifthenelse{\equal{#1}{long}}{%
597     \let\fc@poweroften\fc@@pot@longscalefrench
598   }{%
599     \ifthenelse{\equal{#1}{recursive}}{%
600       \let\fc@poweroften\fc@@pot@recursivefrench
601     }{%
602       \ifthenelse{\equal{#1}{short}}{%
603         \let\fc@poweroften\fc@@pot@shortscalefrench
```

```
604        }{%
605          \PackageError{fmtcount}{Unexpected argument}{%
606            French option 'scale' expects 'long', 'recursive' or 'short'
607          }
608        }%
609      }%
610    }%
611 }%
```

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

infinity in that case ⟨*n*⟩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 ⟨*n*⟩illions"

```
612 \define@key{fcfrench}{n-illiard upto}[infinity]{%
613   \ifthenelse{\equal{#1}{infinity}}{%
614       \def\fc@longscale@nilliard@upto{0}%
615   }{%
616     \ifthenelse{\equal{#1}{infty}}{%
617       \def\fc@longscale@nilliard@upto{0}%
618     }{%
619       \if Q\ifnum9<1#1Q\fi\else
620       \PackageError{fmtcount}{Unexpected argument}{%
621         French option 'milliard threshold' expects 'infinity', or equivalently 'infty', or a no
622         integer.}%
623       \fi
624       \def\fc@longscale@nilliard@upto{#1}%
625     }}%
626 }%
```

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.

```
627 \def\@tempa#1{%
628   \define@key{fcfrench}{#1}[]{%
629     \PackageError{fmtcount}{Unexpected argument}{French option with key '#1' does not take
630       any value}}%
631   \csgdef{KV@fcfrench@#1@default}{%
632     \fc@gl@def\fmtcount@french{#1}}%
633 }%
634 \@tempa{france}\@tempa{swiss}\@tempa{belgian}%
```

Make 'france' the default dialect for 'french' language

```
635 \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.

```
636 \define@key{fcfrench}{dialect}[france]{%
637   \ifthenelse{\equal{#1}{france}
638     \or\equal{#1}{swiss}
```

```
639      \or\equal{#1}{belgian}}{%
640      \def\fmtcount@french{#1}}{%
641      \PackageError{fmtcount}{Invalid value '#1' to french option dialect key}
642      {Option 'french' can only take the values 'france',
643        'belgian' or 'swiss'}}}%
644 \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'.

```
645 \define@key{fcfrench}{mil plural mark}[le]{%
646    \def\fc@frenchoptions@mil@plural@mark{#1}}
647 \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.

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

```
648 \gdef\fc@UpperCaseFirstLetter#1#2\@nil{%
649    \uppercase{#1}#2}
```

`\fc@CaseIden` The macro `\fc@CaseIden` is such that `\fc@CaseIden⟨word⟩\@nil` expands to `\word` unchanged.

```
650 \gdef\fc@CaseIden#1\@nil{%
651    #1%
652 }%
```

`\fc@UpperCaseAll` The macro `\fc@UpperCaseAll` is such that `\fc@UpperCaseAll⟨word⟩\@nil` expands to `\word` all capitalized.

```
653 \gdef\fc@UpperCaseAll#1\@nil{%
654    \uppercase{#1}%
655 }%
```

`\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.

```
656 \global\let\fc@wcase\fc@CaseIden
```

`\fc@gcase` The macro `\fc@gcase` is the capitalizing macro for global (the completed number) capitalization. By default we set it to identity, ie. no capitalization.

```
657 \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.

```
658 \gdef\fc@apply@gcase{%
```

First of all we expand whatever `\fc@wcase...\@nil` found within `\@tempa`.

```
659    \protected@edef\@tempa{\@tempa}%
660    \protected@edef\@tempa{\expandafter\fc@gcase\@tempa\@nil}%
661 }
```

`\@ordinalMfrench`

```
662 \newcommand*{\@ordinalMfrench}[2]{%
663 \iffmtord@abbrv
```

```
664   \ifnum#1=1 %
665     \edef#2{\number#1\relax\noexpand\fmtord{er}}%
666   \else
667     \edef#2{\number#1\relax\noexpand\fmtord{e}}%
668   \fi
669 \else
670   \PackageWarning{fmtcount}{Non abbreviated ordinal finals ('eme) are
671     considered incorrect in French.}%
672   \ifnum#1=1 %
673     \edef#2{\number#1\relax\noexpand\fmtord{er}}%
674   \else
675     \protected@edef#2{\number#1\relax\noexpand\fmtord{\protect\'eme}}%
676   \fi
677 \fi}
678 \global\let\@ordinalMfrench\@ordinalMfrench
```

@ordinalFfrench

```
679 \newcommand*{\@ordinalFfrench}[2]{%
680 \iffmtord@abbrv
681   \ifnum#1=1 %
682     \edef#2{\number#1\relax\noexpand\fmtord{re}}%
683   \else
684     \edef#2{\number#1\relax\noexpand\fmtord{e}}%
685   \fi
686 \else
687   \PackageWarning{fmtcount}{Non abbreviated ordinal finals ('eme) are
688     considered incorrect in French.}%
689   \ifnum#1=1 %
690     \protected@edef#2{\number#1\relax\noexpand\fmtord{\protect\'ere}}%
691   \else
692     \protected@edef#2{\number#1\relax\noexpand\fmtord{\protect\'eme}}%
693   \fi
694 \fi}
695 \global\let\@ordinalFfrench\@ordinalFfrench
```

In French neutral gender and masculine gender are formally identical.

```
696 \global\let\@ordinalNfrench\@ordinalMfrench
```

unitstringfrench

```
697 \newcommand*{\@@unitstringfrench}[1]{%
698 \noexpand\fc@wcase
699 \ifcase#1 %
700 z\'ero%
701 \or un%
702 \or deux%
703 \or trois%
704 \or quatre%
705 \or cinq%
706 \or six%
707 \or sept%
708 \or huit%
```

32

```
709 \or neuf%
710 \fi
711 \noexpand\@nil
712 }%
713 \global\let\@@unitstringfrench\@@unitstringfrench
```

```
714 \newcommand*{\@@tenstringfrench}[1]{%
715 \noexpand\fc@wcase
716 \ifcase#1 %
717 \or dix%
718 \or vingt%
719 \or trente%
720 \or quarante%
721 \or cinquante%
722 \or soixante%
723 \or septante%
724 \or huitante%
725 \or nonante%
726 \or cent%
727 \fi
728 \noexpand\@nil
729 }%
730 \global\let\@@tenstringfrench\@@tenstringfrench
```

```
731 \newcommand*{\@@teenstringfrench}[1]{%
732 \noexpand\fc@wcase
733 \ifcase#1 %
734     dix%
735 \or onze%
736 \or douze%
737 \or treize%
738 \or quatorze%
739 \or quinze%
740 \or seize%
741 \or dix\noexpand\@nil-\noexpand\fc@wcase sept%
742 \or dix\noexpand\@nil-\noexpand\fc@wcase huit%
743 \or dix\noexpand\@nil-\noexpand\fc@wcase neuf%
744 \fi
745 \noexpand\@nil
746 }%
747 \global\let\@@teenstringfrench\@@teenstringfrench
```

```
748 \newcommand*{\@@seventiesfrench}[1]{%
749 \@tenstring{6}%
750 \ifnum#1=1 %
751 \fc@frenchoptions@submillion@dos\@andname\fc@frenchoptions@submillion@dos
752 \else
753 -%
```

```
754 \fi
755 \@teenstring{#1}%
756 }%
757 \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".

```
758 \newcommand*\@@eightiesfrench[1]{%
759 \fc@wcase quatre\@nil-\noexpand\fc@wcase vingt%
760 \ifnum#1>0 %
761   \ifnum\fc@frenchoptions@vingt@plural=0 % vingt plural=always
762   s%
763   \fi
764   \noexpand\@nil
765   -\@unitstring{#1}%
766 \else
767   \ifcase\fc@frenchoptions@vingt@plural\space
768     s% 0: always
769   \or
770     % 1: never
771   \or
772     s% 2: multiple
773   \or
774     % 3: multiple g-last
775     \ifnum\count0=\count6\ifnum\count9=0 s\fi\fi
776   \or
777     % 4: multiple l-last
778     \ifnum\count9=1 %
779     \else
780       s%
781     \fi
782   \or
783     % 5: multiple lng-last
784     \ifnum\count9=1 %
785     \else
786       \ifnum\count0>0 %
787         s%
788       \fi
789     \fi
790   \or
791     % or 6: multiple ng-last
```

```
792    \ifnum\count0>0 %
793        s%
794      \fi
795    \fi
796    \noexpand\@nil
797 \fi
798 }%
799 \global\let\@@eightiesfrench\@@eightiesfrench
800 \newcommand*{\@@ninetiesfrench}[1]{%
801 \fc@wcase quatre\@nil-\noexpand\fc@wcase vingt%
802 \ifnum\fc@frenchoptions@vingt@plural=0 % vingt plural=always
803    s%
804 \fi
805 \noexpand\@nil
806 -\@teenstring{#1}%
807 }%
808 \global\let\@@ninetiesfrench\@@ninetiesfrench
809 \newcommand*{\@@seventiesfrenchswiss}[1]{%
810 \@tenstring{7}%
811 \ifnum#1=1\ \@andname\ \fi
812 \ifnum#1>1-\fi
813 \ifnum#1>0 \@unitstring{#1}\fi
814 }%
815 \global\let\@@seventiesfrenchswiss\@@seventiesfrenchswiss
816 \newcommand*{\@@eightiesfrenchswiss}[1]{%
817 \@tenstring{8}%
818 \ifnum#1=1\ \@andname\ \fi
819 \ifnum#1>1-\fi
820 \ifnum#1>0 \@unitstring{#1}\fi
821 }%
822 \global\let\@@eightiesfrenchswiss\@@eightiesfrenchswiss
823 \newcommand*{\@@ninetiesfrenchswiss}[1]{%
824 \@tenstring{9}%
825 \ifnum#1=1\ \@andname\ \fi
826 \ifnum#1>1-\fi
827 \ifnum#1>0 \@unitstring{#1}\fi
828 }%
829 \global\let\@@ninetiesfrenchswiss\@@ninetiesfrenchswiss
```

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

```
830 \newcommand*\fc@french@common{%
831    \let\fc@wcase\fc@CaseIden
832    \let\@unitstring=\@@unitstringfrench
833    \let\@teenstring=\@@teenstringfrench
834    \let\@tenstring=\@@tenstringfrench
835    \def\@hundred{cent}%
836    \def\@andname{et}%
837 }%
838 \global\let\fc@french@common\fc@french@common
```

```
839 \newcommand*{\@numberstringMfrenchswiss}[2]{%
840 \fc@french@common
841 \let\fc@gcase\fc@CaseIden
842 \let\@seventies=\@@seventiesfrenchswiss
843 \let\@eighties=\@@eightiesfrenchswiss
844 \let\@nineties=\@@ninetiesfrenchswiss
845 \let\fc@nbrstr@preamble\@empty
846 \let\fc@nbrstr@postamble\@empty
847 \@@numberstringfrench{#1}{#2}}
848 \global\let\@numberstringMfrenchswiss\@numberstringMfrenchswiss
849 \newcommand*{\@numberstringMfrenchfrance}[2]{%
850 \fc@french@common
851 \let\fc@gcase\fc@CaseIden
852 \let\@seventies=\@@seventiesfrench
853 \let\@eighties=\@@eightiesfrench
854 \let\@nineties=\@@ninetiesfrench
855 \let\fc@nbrstr@preamble\@empty
856 \let\fc@nbrstr@postamble\@empty
857 \@@numberstringfrench{#1}{#2}}
858 \global\let\@numberstringMfrenchfrance\@numberstringMfrenchfrance
859 \newcommand*{\@numberstringMfrenchbelgian}[2]{%
860 \fc@french@common
861 \let\fc@gcase\fc@CaseIden
862 \let\@seventies=\@@seventiesfrenchswiss
863 \let\@eighties=\@@eightiesfrench
864 \let\@nineties=\@@ninetiesfrench
865 \let\fc@nbrstr@preamble\@empty
866 \let\fc@nbrstr@postamble\@empty
867 \@@numberstringfrench{#1}{#2}}
868 \global\let\@numberstringMfrenchbelgian\@numberstringMfrenchbelgian
869 \let\@numberstringMfrench=\@numberstringMfrenchfrance
870 \newcommand*{\@numberstringFfrenchswiss}[2]{%
871 \fc@french@common
872 \let\fc@gcase\fc@CaseIden
873 \let\@seventies=\@@seventiesfrenchswiss
874 \let\@eighties=\@@eightiesfrenchswiss
875 \let\@nineties=\@@ninetiesfrenchswiss
876 \let\fc@nbrstr@preamble\fc@@nbrstr@Fpreamble
877 \let\fc@nbrstr@postamble\@empty
878 \@@numberstringfrench{#1}{#2}}
879 \global\let\@numberstringFfrenchswiss\@numberstringFfrenchswiss
880 \newcommand*{\@numberstringFfrenchfrance}[2]{%
881 \fc@french@common
882 \let\fc@gcase\fc@CaseIden
883 \let\@seventies=\@@seventiesfrench
884 \let\@eighties=\@@eightiesfrench
885 \let\@nineties=\@@ninetiesfrench
886 \let\fc@nbrstr@preamble\fc@@nbrstr@Fpreamble
887 \let\fc@nbrstr@postamble\@empty
```

```
888 \@@numberstringfrench{#1}{#2}}
889 \global\let\@numberstringFfrenchfrance\@numberstringFfrenchfrance
890 \newcommand*{\@numberstringFfrenchbelgian}[2]{%
891 \fc@french@common
892 \let\fc@gcase\fc@CaseIden
893 \let\@seventies=\@@seventiesfrenchswiss
894 \let\@eighties=\@@eightiesfrench
895 \let\@nineties=\@@ninetiesfrench
896 \let\fc@nbrstr@preamble\fc@@nbrstr@Fpreamble
897 \let\fc@nbrstr@postamble\@empty
898 \@@numberstringfrench{#1}{#2}}
899 \global\let\@numberstringFfrenchbelgian\@numberstringFfrenchbelgian
900 \global\let\@numberstringFfrench=\@numberstringFfrenchfrance
901 \global\let\@ordinalstringNfrench\@ordinalstringMfrench
902 \newcommand*{\@NumberstringMfrenchswiss}[2]{%
903 \fc@french@common
904 \let\fc@gcase\fc@UpperCaseFirstLetter
905 \let\@seventies=\@@seventiesfrenchswiss
906 \let\@eighties=\@@eightiesfrenchswiss
907 \let\@nineties=\@@ninetiesfrenchswiss
908 \let\fc@nbrstr@preamble\@empty
909 \let\fc@nbrstr@postamble\fc@apply@gcase
910 \@@numberstringfrench{#1}{#2}}
911 \global\let\@NumberstringMfrenchswiss\@NumberstringMfrenchswiss
912 \newcommand*{\@NumberstringMfrenchfrance}[2]{%
913 \fc@french@common
914 \let\fc@gcase\fc@UpperCaseFirstLetter
915 \let\@seventies=\@@seventiesfrench
916 \let\@eighties=\@@eightiesfrench
917 \let\@nineties=\@@ninetiesfrench
918 \let\fc@nbrstr@preamble\@empty
919 \let\fc@nbrstr@postamble\fc@apply@gcase
920 \@@numberstringfrench{#1}{#2}}
921 \global\let\@NumberstringMfrenchfrance\@NumberstringMfrenchfrance
922 \newcommand*{\@NumberstringMfrenchbelgian}[2]{%
923 \fc@french@common
924 \let\fc@gcase\fc@UpperCaseFirstLetter
925 \let\@seventies=\@@seventiesfrenchswiss
926 \let\@eighties=\@@eightiesfrench
927 \let\@nineties=\@@ninetiesfrench
928 \let\fc@nbrstr@preamble\@empty
929 \let\fc@nbrstr@postamble\fc@apply@gcase
930 \@@numberstringfrench{#1}{#2}}
931 \global\let\@NumberstringMfrenchbelgian\@NumberstringMfrenchbelgian
932 \global\let\@NumberstringMfrench=\@NumberstringMfrenchfrance
933 \newcommand*{\@NumberstringFfrenchswiss}[2]{%
934 \fc@french@common
935 \let\fc@gcase\fc@UpperCaseFirstLetter
936 \let\@seventies=\@@seventiesfrenchswiss
```

```
937 \let\@eighties=\@@eightiesfrenchswiss
938 \let\@nineties=\@@ninetiesfrenchswiss
939 \let\fc@nbrstr@preamble\fc@@nbrstr@Fpreamble
940 \let\fc@nbrstr@postamble\fc@apply@gcase
941 \@@numberstringfrench{#1}{#2}}
942 \global\let\@NumberstringFfrenchswiss\@NumberstringFfrenchswiss
943 \newcommand*{\@NumberstringFfrenchfrance}[2]{%
944 \fc@french@common
945 \let\fc@gcase\fc@UpperCaseFirstLetter
946 \let\@seventies=\@@seventiesfrench
947 \let\@eighties=\@@eightiesfrench
948 \let\@nineties=\@@ninetiesfrench
949 \let\fc@nbrstr@preamble\fc@@nbrstr@Fpreamble
950 \let\fc@nbrstr@postamble\fc@apply@gcase
951 \@@numberstringfrench{#1}{#2}}
952 \global\let\@NumberstringFfrenchfrance\@NumberstringFfrenchfrance
953 \newcommand*{\@NumberstringFfrenchbelgian}[2]{%
954 \fc@french@common
955 \let\fc@gcase\fc@UpperCaseFirstLetter
956 \let\@seventies=\@@seventiesfrenchswiss
957 \let\@eighties=\@@eightiesfrench
958 \let\@nineties=\@@ninetiesfrench
959 \let\fc@nbrstr@preamble\fc@@nbrstr@Fpreamble
960 \let\fc@nbrstr@postamble\fc@apply@gcase
961 \@@numberstringfrench{#1}{#2}}
962 \global\let\@NumberstringFfrenchbelgian\@NumberstringFfrenchbelgian
963 \global\let\@NumberstringFfrench=\@NumberstringFfrenchfrance
964 \global\let\@NumberstringNfrench\@NumberstringMfrench
965 \newcommand*{\@ordinalstringMfrenchswiss}[2]{%
966 \fc@french@common
967 \let\fc@gcase\fc@CaseIden
968 \let\fc@first\fc@@firstfrench
969 \let\@seventies=\@@seventiesfrenchswiss
970 \let\@eighties=\@@eightiesfrenchswiss
971 \let\@nineties=\@@ninetiesfrenchswiss
972 \@@ordinalstringfrench{#1}{#2}%
973 }%
974 \global\let\@ordinalstringMfrenchswiss\@ordinalstringMfrenchswiss
975 \newcommand*\fc@@firstfrench{premier}
976 \global\let\fc@@firstfrench\fc@@firstfrench

977 \newcommand*\fc@@firstFfrench{premi\protect\`ere}
978 \global\let\fc@@firstFfrench\fc@@firstFfrench
979 \newcommand*{\@ordinalstringMfrenchfrance}[2]{%
980 \fc@french@common
981 \let\fc@gcase\fc@CaseIden
982 \let\fc@first=\fc@@firstfrench
983 \let\@seventies=\@@seventiesfrench
984 \let\@eighties=\@@eightiesfrench
985 \let\@nineties=\@@ninetiesfrench
```

```
986 \@@ordinalstringfrench{#1}{#2}}
987 \global\let\@ordinalstringMfrenchfrance\@ordinalstringMfrenchfrance
988 \newcommand*{\@ordinalstringMfrenchbelgian}[2]{%
989 \fc@french@common
990 \let\fc@gcase\fc@CaseIden
991 \let\fc@first=\fc@@firstfrench
992 \let\@seventies=\@@seventiesfrench
993 \let\@eighties=\@@eightiesfrench
994 \let\@nineties=\@@ninetiesfrench
995 \@@ordinalstringfrench{#1}{#2}%
996 }%
997 \global\let\@ordinalstringMfrenchbelgian\@ordinalstringMfrenchbelgian
998 \global\let\@ordinalstringMfrench=\@ordinalstringMfrenchfrance
999 \newcommand*{\@ordinalstringFfrenchswiss}[2]{%
1000 \fc@french@common
1001 \let\fc@gcase\fc@CaseIden
1002 \let\fc@first\fc@@firstFfrench
1003 \let\@seventies=\@@seventiesfrenchswiss
1004 \let\@eighties=\@@eightiesfrenchswiss
1005 \let\@nineties=\@@ninetiesfrenchswiss
1006 \@@ordinalstringfrench{#1}{#2}%
1007 }%
1008 \global\let\@ordinalstringFfrenchswiss\@ordinalstringFfrenchswiss
1009 \newcommand*{\@ordinalstringFfrenchfrance}[2]{%
1010 \fc@french@common
1011 \let\fc@gcase\fc@CaseIden
1012 \let\fc@first=\fc@@firstFfrench
1013 \let\@seventies=\@@seventiesfrench
1014 \let\@eighties=\@@eightiesfrench
1015 \let\@nineties=\@@ninetiesfrench
1016 \@@ordinalstringfrench{#1}{#2}%
1017 }%
1018 \global\let\@ordinalstringFfrenchfrance\@ordinalstringFfrenchfrance
1019 \newcommand*{\@ordinalstringFfrenchbelgian}[2]{%
1020 \fc@french@common
1021 \let\fc@gcase\fc@CaseIden
1022 \let\fc@first=\fc@@firstFfrench
1023 \let\@seventies=\@@seventiesfrench
1024 \let\@eighties=\@@eightiesfrench
1025 \let\@nineties=\@@ninetiesfrench
1026 \@@ordinalstringfrench{#1}{#2}%
1027 }%
1028 \global\let\@ordinalstringFfrenchbelgian\@ordinalstringFfrenchbelgian
1029 \global\let\@ordinalstringFfrench=\@ordinalstringFfrenchfrance
1030 \global\let\@ordinalstringNfrench\@ordinalstringMfrench
1031 \newcommand*{\@OrdinalstringMfrenchswiss}[2]{%
1032 \fc@french@common
1033 \let\fc@gcase\fc@UpperCaseFirstLetter
1034 \let\fc@first=\fc@@firstfrench
```

```
1035 \let\@seventies=\@@seventiesfrenchswiss
1036 \let\@eighties=\@@eightiesfrenchswiss
1037 \let\@nineties=\@@ninetiesfrenchswiss
1038 \@@ordinalstringfrench{#1}{#2}%
1039 }%
1040 \global\let\@OrdinalstringMfrenchswiss\@OrdinalstringMfrenchswiss
1041 \newcommand*{\@OrdinalstringMfrenchfrance}[2]{%
1042 \fc@french@common
1043 \let\fc@gcase\fc@UpperCaseFirstLetter
1044 \let\fc@first\fc@@firstfrench
1045 \let\@seventies=\@@seventiesfrench
1046 \let\@eighties=\@@eightiesfrench
1047 \let\@nineties=\@@ninetiesfrench
1048 \@@ordinalstringfrench{#1}{#2}%
1049 }%
1050 \global\let\@OrdinalstringMfrenchfrance\@OrdinalstringMfrenchfrance
1051 \newcommand*{\@OrdinalstringMfrenchbelgian}[2]{%
1052 \fc@french@common
1053 \let\fc@gcase\fc@UpperCaseFirstLetter
1054 \let\fc@first\fc@@firstfrench
1055 \let\@seventies=\@@seventiesfrench
1056 \let\@eighties=\@@eightiesfrench
1057 \let\@nineties=\@@ninetiesfrench
1058 \@@ordinalstringfrench{#1}{#2}%
1059 }%
1060 \global\let\@OrdinalstringMfrenchbelgian\@OrdinalstringMfrenchbelgian
1061 \global\let\@OrdinalstringMfrench=\@OrdinalstringMfrenchfrance
1062 \newcommand*{\@OrdinalstringFfrenchswiss}[2]{%
1063 \fc@french@common
1064 \let\fc@gcase\fc@UpperCaseFirstLetter
1065 \let\fc@first\fc@@firstfrench
1066 \let\@seventies=\@@seventiesfrenchswiss
1067 \let\@eighties=\@@eightiesfrenchswiss
1068 \let\@nineties=\@@ninetiesfrenchswiss
1069 \@@ordinalstringfrench{#1}{#2}%
1070 }%
1071 \global\let\@OrdinalstringFfrenchswiss\@OrdinalstringFfrenchswiss
1072 \newcommand*{\@OrdinalstringFfrenchfrance}[2]{%
1073 \fc@french@common
1074 \let\fc@gcase\fc@UpperCaseFirstLetter
1075 \let\fc@first\fc@@firstFfrench
1076 \let\@seventies=\@@seventiesfrench
1077 \let\@eighties=\@@eightiesfrench
1078 \let\@nineties=\@@ninetiesfrench
1079 \@@ordinalstringfrench{#1}{#2}%
1080 }%
1081 \global\let\@OrdinalstringFfrenchfrance\@OrdinalstringFfrenchfrance
1082 \newcommand*{\@OrdinalstringFfrenchbelgian}[2]{%
1083 \fc@french@common
```

```
1084 \let\fc@gcase\fc@UpperCaseFirstLetter
1085 \let\fc@first\fc@@firstFfrench
1086 \let\@seventies=\@@seventiesfrench
1087 \let\@eighties=\@@eightiesfrench
1088 \let\@nineties=\@@ninetiesfrench
1089 \@@ordinalstringfrench{#1}{#2}%
1090 }%
1091 \global\let\@OrdinalstringFfrenchbelgian\@OrdinalstringFfrenchbelgian
1092 \global\let\@OrdinalstringFfrench=\@OrdinalstringFfrenchfrance
1093 \global\let\@OrdinalstringNfrench\@OrdinalstringMfrench
```

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

```
1094 \ifcsundef{fc@@do@plural@mark}{}%
1095 {\PackageError{fmtcount}{Duplicate definition}{Redefinition of macro
1096    '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 ⟨*n*⟩illiard, ⟨*n*⟩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.

```
1097 \def\fc@@do@plural@mark#1{%
1098   \ifcase\count10 %
1099     #1% 0=always
1100   \or% 1=never
1101   \or% 2=multiple
1102     \ifnum\count1>1 %
1103       #1%
1104     \fi
1105   \or% 3= multiple g-last
1106     \ifnum\count1>1 %
1107       \ifnum\count0=\count6 %
1108         #1%
1109       \fi
1110     \fi
1111   \or% 4= multiple l-last
1112     \ifnum\count1>1 %
1113       \ifnum\count9=1 %
1114       \else
1115         #1%
1116       \fi
1117     \fi
1118   \or% 5= multiple lng-last
```

```
1119        \ifnum\count1>1 %
1120          \ifnum\count9=1 %
1121          \else
1122            \if\count0>\count6 %
1123              #1%
1124            \fi
1125          \fi
1126        \fi
1127  \or% 6= multiple ng-last
1128        \ifnum\count1>1 %
1129          \ifnum\count0>\count6 %
1130            #1%
1131          \fi
1132        \fi
1133  \fi
1134 }%
1135 \global\let\fc@@do@plural@mark\fc@@do@plural@mark
```

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

```
1136 \ifcsundef{fc@@nbrstr@Fpreamble}{}{%
1137   \PackageError{fmtcount}{Duplicate definition}{Redefinition of macro
1138     'fc@@nbrstr@Fpreamble'}}
```

@@nbrstr@Fpreamble

```
1139 \def\fc@@nbrstr@Fpreamble{%
1140   \fc@read@unit{\count1}{0}%
1141   \ifnum\count1=1 %
1142       \let\fc@wcase@save\fc@wcase
1143       \def\fc@wcase{\noexpand\fc@wcase}%
1144       \def\@nil{\noexpand\@nil}%
1145     \let\fc@nbrstr@postamble\fc@@nbrstr@Fpostamble
1146   \fi
1147 }%
1148 \global\let\fc@@nbrstr@Fpreamble\fc@@nbrstr@Fpreamble
```

@@nbrstr@Fpostamble

```
1149 \def\fc@@nbrstr@Fpostamble{%
1150   \let\fc@wcase\fc@wcase@save
1151   \expandafter\fc@get@last@word\expandafter{\@tempa}\@tempb\@tempc
1152   \def\@tempd{un}%
1153   \ifx\@tempc\@tempd
1154     \let\@tempc\@tempa
1155     \edef\@tempa{\@tempb\fc@wcase une\@nil}%
1156   \fi
1157 }%
1158 \global\let\fc@@nbrstr@Fpostamble\fc@@nbrstr@Fpostamble
```

@@pot@longscalefrenchMacro \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.

```
1159 \ifcsundef{fc@@pot@longscalefrench}{}{%
1160   \PackageError{fmtcount}{Duplicate definition}{Redefinition of macro
1161     '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

```
1162 \def\fc@@pot@longscalefrench#1#2#3{%
1163   {%
```

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

```
1164     \edef\@tempb{\number#1}%
```

Let \count1 be the plural value.

```
1165     \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 $0 \le r < 6$, then \count2 is set to *n* and \count3 is set to *r*.

```
1166     \count2\count0 %
1167     \divide\count2 by 6 %
1168     \count3\count2 %
1169     \multiply\count3 by 6 %
1170     \count3-\count3 %
1171     \advance\count3 by \count0 %
1172     \ifnum\count0>0 %
```

If weight *w* (a.k.a. \count0) is such that $w > 0$, then $w \ge 3$ 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)".

```
1173       \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 \@temph to '1' for "mil(le)", and to '2' otherwise.

```
1174         \edef\@temph{%
1175           \ifnum\count2=0 % weight=3
```

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

```
1176             1%
1177           \else
1178             \ifnum\count3>2 %
```

Here we are in the case of $3 \le r < 6$, 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.

43

```
1179                \ifnum\fc@longscale@nilliard@upto=0 %
```

Here option 'n-illiard upto' is 'infinity', so we always use "⟨*n*⟩illiard(s)".

```
1180                    2%
1181                \else
```

Here option 'n-illiard upto' indicate some threshold to which to compare *n* (a.k.a. \count2).

```
1182                    \ifnum\count2>\fc@longscale@nilliard@upto
1183                      1%
1184                    \else
1185                      2%
1186                    \fi
1187                \fi
1188              \else
1189                2%
1190              \fi
1191            \fi
1192          }%
1193          \ifnum\@temph=1 %
```

Here 10^w is formatted as "mil(le)".

```
1194              \count10=\fc@frenchoptions@mil@plural\space
1195              \edef\@tempe{%
1196                \noexpand\fc@wcase
1197                mil%
1198                \fc@@do@plural@mark\fc@frenchoptions@mil@plural@mark
1199                \noexpand\@nil
1200              }%
1201          \else
1202          % weight >= 6
1203          \expandafter\fc@@latin@cardinal@pefix\expandafter{\the\count2}\@tempg
1204          % now form the xxx-illion(s) or xxx-illiard(s) word
1205          \ifnum\count3>2 %
1206              \toks10{illiard}%
1207              \count10=\csname fc@frenchoptions@n-illiard@plural\endcsname\space
1208          \else
1209              \toks10{illion}%
1210              \count10=\csname fc@frenchoptions@n-illion@plural\endcsname\space
1211          \fi
1212          \edef\@tempe{%
1213            \noexpand\fc@wcase
1214            \@tempg
1215            \the\toks10 %
1216            \fc@@do@plural@mark s%
1217            \noexpand\@nil
1218          }%
1219        \fi
1220      \else
```

Here plural indicator of *d* indicates that $d = 0$, so we have 0×10^w, and it is not worth to format

10^w, because there are none of them.

```
1221            \let\@tempe\@empty
1222            \def\@temph{0}%
1223         \fi
1224      \else
```

Case of $w = 0$.

```
1225            \let\@tempe\@empty
1226            \def\@temph{0}%
1227      \fi
```

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

```
1228      \expandafter\toks\expandafter1\expandafter{\@tempe}%
1229      \toks0{#2}%
1230      \edef\@tempa{\the\toks0 \@temph \def\noexpand#3{\the\toks1}}%
1231      \expandafter
1232   }\@tempa
1233 }%
1234 \global\let\fc@@pot@longscalefrench\fc@@pot@longscalefrench
```

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.

```
1235 \ifcsundef{fc@@pot@shortscalefrench}{}{%
1236   \PackageError{fmtcount}{Duplicate definition}{Redefinition of macro
1237    '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

```
1238 \def\fc@@pot@shortscalefrench#1#2#3{%
1239   {%
```

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

```
1240      \edef\@tempb{\number#1}%
```

And let \count1 be the plural value.

```
1241      \count1=\@tempb
```

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

```
1242      \count2\count0 %
1243      \divide\count2 by 3 %
1244      \advance\count2 by -1 %
```

45

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 "⟨*n*⟩illion(s)|⟨*n*⟩illiard(s)".

```
1245     \ifnum\count0>0 % If weight>=3, i.e we do have to append thousand or n-illion(s)/n-illiard(
1246       \ifnum\count1>0 % we have at least one thousand/n-illion/n-illiard
1247         \ifnum\count2=0 %
1248           \def\@temph{1}%
1249           \count1=\fc@frenchoptions@mil@plural\space
1250           \edef\@tempe{%
1251             mil%
1252             \fc@@do@plural@mark\fc@frenchoptions@mil@plural@mark
1253           }%
1254         \else
1255           \def\@temph{2}%
1256           % weight >= 6
1257           \expandafter\fc@@latin@cardinal@pefix\expandafter{\the\count2}\@tempg
1258           \count10=\csname fc@frenchoptions@n-illion@plural\endcsname\space
1259           \edef\@tempe{%
1260             \noexpand\fc@wcase
1261             \@tempg
1262             illion%
1263             \fc@@do@plural@mark s%
1264             \noexpand\@nil
1265           }%
1266         \fi
1267       \else
```

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

```
1268           \def\@temph{0}%
1269           \let\@tempe\@empty
1270       \fi
1271     \else
```

Here $w = 0$.

```
1272       \def\@temph{0}%
1273       \let\@tempe\@empty
1274     \fi
1275 % now place into \@cs{@tempa} the assignment of results \cs{@temph} and \cs{@tempe} to to \text
1276 % \texttt{\#3} for further propagation after closing brace.
1277 %     \begin{macrocode}
1278     \expandafter\toks\expandafter1\expandafter{\@tempe}%
1279     \toks0{#2}%
1280     \edef\@tempa{\the\toks0 \@temph \def\noexpand#3{\the\toks1}}%
1281     \expandafter
1282   }\@tempa
1283 }%
1284 \global\let\fc@@pot@shortscalefrench\fc@@pot@shortscalefrench
```

`@@pot@recursivefrench` 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.

```
1285 \ifcsundef{fc@@pot@recursivefrench}{}{%
1286   \PackageError{fmtcount}{Duplicate definition}{Redefinition of macro
1287     '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

```
1288 \def\fc@@pot@recursivefrench#1#2#3{%
1289   {%
```

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

```
1290     \edef\@tempb{\number#1}%
1291     \let\@tempa\@@tempa
```

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

```
1292     \count1=\@tempb\space
```

Now compute into \count2 how many times "de milliards" has to be repeated.

```
1293     \ifnum\count1>0 %
1294       \count2\count0 %
1295       \divide\count2 by 9 %
1296       \advance\count2 by -1 %
1297       \let\@tempe\@empty
1298       \edef\@tempf{\fc@frenchoptions@supermillion@dos
1299         de\fc@frenchoptions@supermillion@dos\fc@wcase milliards\@nil}%
1300       \count11\count0 %
1301       \ifnum\count2>0 %
1302         \count3\count2 %
1303         \count3-\count3 %
1304         \multiply\count3 by 9 %
1305         \advance\count11 by \count3 %
1306         \loop
1307           % (\count2, \count3) <- (\count2 div 2, \count2 mod 2)
1308           \count3\count2 %
1309           \divide\count3 by 2 %
1310           \multiply\count3 by 2 %
1311           \count3-\count3 %
1312           \advance\count3 by \count2 %
1313           \divide\count2 by 2 %
1314           \ifnum\count3=1 %
1315             \let\@tempg\@tempe
1316             \edef\@tempe{\@tempg\@tempf}%
1317           \fi
1318           \let\@tempg\@tempf
```

```
1319        \edef\@tempf{\@tempg\@tempg}%
1320        \ifnum\count2>0 %
1321     \repeat
1322    \fi
1323    \divide\count11 by 3 %
1324    \ifcase\count11 % 0 .. 5
1325     % 0 => d milliard(s) (de milliards)*
1326     \def\@temph{2}%
1327     \count10=\csname fc@frenchoptions@n-illiard@plural\endcsname\space
1328    \or  % 1 => d mille milliard(s) (de milliards)*
1329     \def\@temph{1}%
1330     \count10=\fc@frenchoptions@mil@plural\space
1331    \or % 2 => d million(s) (de milliards)*
1332     \def\@temph{2}%
1333     \count10=\csname fc@frenchoptions@n-illion@plural\endcsname\space
1334    \or % 3 => d milliard(s) (de milliards)*
1335     \def\@temph{2}%
1336     \count10=\csname fc@frenchoptions@n-illiard@plural\endcsname\space
1337    \or % 4 => d mille milliards (de milliards)*
1338     \def\@temph{1}%
1339     \count10=\fc@frenchoptions@mil@plural\space
1340    \else % 5 => d million(s) (de milliards)*
1341     \def\@temph{2}%
1342     \count10=\csname fc@frenchoptions@n-illion@plural\endcsname\space
1343    \fi
1344    \let\@tempg\@tempe
1345    \edef\@tempf{%
1346     \ifcase\count11 % 0 .. 5
1347     \or
1348       mil\fc@@do@plural@mark \fc@frenchoptions@mil@plural@mark
1349     \or
1350       million\fc@@do@plural@mark s%
1351     \or
1352       milliard\fc@@do@plural@mark s%
1353     \or
1354       mil\fc@@do@plural@mark\fc@frenchoptions@mil@plural@mark
1355       \noexpand\@nil\fc@frenchoptions@supermillion@dos
1356       \noexpand\fc@wcase milliards% 4
1357     \or
1358       million\fc@@do@plural@mark s%
1359       \noexpand\@nil\fc@frenchoptions@supermillion@dos
1360       de\fc@frenchoptions@supermillion@dos\noexpand\fc@wcase  milliards% 5
1361     \fi
1362    }%
1363    \edef\@tempe{%
1364     \ifx\@tempf\@empty\else
1365     \expandafter\fc@wcase\@tempf\@nil
1366     \fi
1367     \@tempg
```

```
1368          }%
1369      \else
1370          \def\@temph{0}%
1371          \let\@tempe\@empty
1372      \fi
```

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

```
1373      \expandafter\toks\expandafter1\expandafter{\@tempe}%
1374      \toks0{#2}%
1375      \edef\@tempa{\the\toks0 \@temph \def\noexpand#3{\the\toks1}}%
1376      \expandafter
1377  }\@tempa
1378 }%
1379 \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.

```
1380 \ifcsundef{fc@muladdfrench}{}{%
1381    \PackageError{fmtcount}{Duplicate definition}{Redefinition of macro
1382      '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

```
1383 \def\fc@muladdfrench#1#2#3{%
1384    {%
```

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

```
1385      \edef\@@tempc{#1}%
1386      \edef\@@tempd{#2}%
1387      \edef\@tempf{#3}%
1388      \let\@tempc\@@tempc
1389      \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".

```
1390      \ifnum\@tempc=1 %
1391        \ifnum\count9=1 %
1392          \let\@tempd\@empty
1393        \fi
1394      \fi
```

Now we do the "multiplication" of $d = $ \@tempd and of $10^w = $ \@tempf, and place the result into \@tempg.

```
1395      \edef\@tempg{%
1396        \@tempd
1397        \ifx\@tempd\@empty\else
1398          \ifx\@tempf\@empty\else
1399            \ifcase\count9 %
1400            \or
1401              \fc@frenchoptions@submillion@dos
1402            \or
1403              \fc@frenchoptions@supermillion@dos
1404            \fi
1405          \fi
1406        \fi
1407        \@tempf
1408      }%
```

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

```
1409      \edef\@temph{%
1410        \@tempa
1411        \ifx\@tempa\@empty\else
1412          \ifx\@tempg\@empty\else
1413            \ifcase\count8 %
1414            \or
1415              \fc@frenchoptions@submillion@dos
1416            \or
1417              \fc@frenchoptions@supermillion@dos
1418            \fi
1419          \fi
1420        \fi
1421        \@tempg
1422      }%
```

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

```
1423      \def\@tempb##1{\def\@tempa{\def\@tempa{##1}}}%
1424      \expandafter\@tempb\expandafter{\@temph}%
1425      \expandafter
1426  }\@tempa
1427 }%
1428 \global\let\fc@muladdfrench\fc@muladdfrench
```

\fc@lthundredstringfrench Macro \fc@lthundredstringfrench is used to format a number in interval $[0..99]$. First we

check that it is not already defined.

```
1429 \ifcsundef{fc@lthundredstringfrench}{}{%
1430    \PackageError{fmtcount}{Duplicate definition}{Redefinition of macro
1431      'fc@lthundredstringfrench'}}
```

The number to format is not passed as an argument to this macro, instead each digits of it is in a `\fc@digit@⟨w⟩` macro after this number has been parsed. So the only thing that `\fc@lthundredstringfrench` needs is to know $⟨w⟩$ which is passed as `\count0` for the less significant digit.

#1   intput/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.

```
1432 \def\fc@lthundredstringfrench#1{%
1433    {%
```

First save arguments into local temporary macro.

```
1434      \let\@tempc#1%
```

Read units d_w to `\count1`.

```
1435      \fc@read@unit{\count1}{\count0}%
```

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

```
1436      \count3\count0 %
1437      \advance\count3 1 %
1438      \fc@read@unit{\count2}{\count3}%
```

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

```
1439      \edef\@tempa{%
1440        \@tempc
1441        \ifnum\count2>1 %
1442          % 20 .. 99
1443          \ifnum\count2>6 %
1444            % 70 .. 99
1445            \ifnum\count2<8 %
1446              % 70 .. 79
1447              \@seventies{\count1}%
1448          \else
1449            % 80..99
1450            \ifnum\count2<9 %
1451              % 80 .. 89
1452              \@eighties{\count1}%
1453            \else
1454              % 90 .. 99
1455              \@nineties{\count1}%
1456            \fi
1457          \fi
1458        \else
1459          % 20..69
1460          \@tenstring{\count2}%
```

```
1461          \ifnum\count1>0 %
1462              % x1 .. x0
1463              \ifnum\count1=1 %
1464                % x1
1465                \fc@frenchoptions@submillion@dos\@andname\fc@frenchoptions@submillion@dos
1466              \else
1467                % x2 .. x9
1468                -%
1469              \fi
1470              \@unitstring{\count1}%
1471            \fi
1472          \fi
1473        \else
1474          % 0 .. 19
1475          \ifnum\count2=0 % when tens = 0
1476            % 0 .. 9
1477            \ifnum\count1=0 % when units = 0
1478              % \count3=1 when #1 = 0, i.e. only for the unit of the top level number
1479              \ifnum\count3=1 %
1480                \ifnum\fc@max@weight=0 %
1481                  \@unitstring{0}%
1482                \fi
1483              \fi
1484            \else
1485              % 1 .. 9
1486              \@unitstring{\count1}%
1487            \fi
1488          \else
1489            % 10 .. 19
1490            \@teenstring{\count1}%
1491          \fi
1492        \fi
1493      }%
```

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

```
1494      \def\@tempb##1{\def\@tempa{\def#1{##1}}}%
1495      \expandafter\@tempb\expandafter{\@tempa}%
1496      \expandafter
1497    }\@tempa
1498 }%
1499 \global\let\fc@lthundredstringfrench\fc@lthundredstringfrench
```

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

```
1500 \ifcsundef{fc@ltthousandstringfrench}{}{%
1501   \PackageError{fmtcount}{Duplicate definition}{Redefinition of macro
1502     'fc@ltthousandstringfrench'}}
```

Output is empty for 0. Arguments as follows:

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

52

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 $0 \Rightarrow \varnothing$, $1 \Rightarrow$ "mil(le)", $2 \Rightarrow$ $\langle n\rangle$illion(s)$|\langle n\rangle$illiard(s)

```
1503 \def\fc@ltthousandstringfrench#1{%
1504   {%
```

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

```
1505     \count4\count0 %
1506     \advance\count4 by 2 %
1507     \fc@read@unit{\count2 }{\count4 }%
```

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

```
1508     \advance\count4 by -1 %
1509     \count3\count4 %
1510     \advance\count3 by -1 %
1511     \fc@check@nonzeros{\count3 }{\count4 }\@tempa
```

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

```
1512     \edef\@temps{%
1513       \ifcase\fc@frenchoptions@cent@plural\space
1514       % 0 => always
1515       s%
1516       \or
1517       % 1 => never
1518       \or
1519       % 2 => multiple
1520       \ifnum\count2>1s\fi
1521       \or
1522       % 3 => multiple g-last
1523         \ifnum\count2>1 \ifnum\@tempa=0 \ifnum\count0=\count6s\fi\fi\fi
1524       \or
1525       % 4 => multiple l-last
1526         \ifnum\count2>1 \ifnum\@tempa=0 \ifnum\count9=0s\else\ifnum\count9=2s\fi\fi\fi\fi
1527       \fi
1528     }%
1529     % compute spacing after cent(s?) into \@tempb
1530     \expandafter\let\expandafter\@tempb
1531       \ifnum\@tempa>0 \fc@frenchoptions@submillion@dos\else\@empty\fi
1532     % now place into \@tempa the hundreds
1533     \edef\@tempa{%
1534       \ifnum\count2=0 %
1535       \else
1536         \ifnum\count2=1 %
1537           \expandafter\fc@wcase\@hundred\@nil
1538         \else
1539           \@unitstring{\count2}\fc@frenchoptions@submillion@dos
1540           \noexpand\fc@wcase\@hundred\@temps\noexpand\@nil
1541         \fi
```

```
1542            \@tempb
1543          \fi
1544      }%
1545      % now append to \@tempa the ten and unit
1546      \fc@lthundredstringfrench\@tempa
```

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

```
1547      \def\@tempb##1{\def\@tempa{\def#1{##1}}}%
1548      \expandafter\@tempb\expandafter{\@tempa}%
1549      \expandafter
1550    }\@tempa
1551 }%
1552 \global\let\fc@ltthousandstringfrench\fc@ltthousandstringfrench
```

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

```
1553 \ifcsundef{@@numberstringfrench}{}{%
1554    \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

```
1555 \def\@@numberstringfrench#1#2{%
1556    {%
```

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

```
1557        \edef\@tempa{#1}%
1558        \expandafter\fc@number@parser\expandafter{\@tempa}%
1559        \ifnum\fc@min@weight<0 %
1560            \PackageError{fmtcount}{Out of range}%
1561                {This macro does not work with fractional numbers}%
1562        \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'.

```
1563        \edef\@tempa{\ifcase\fc@sign@case\space\or\fc@wcase plus\@nil\or\fc@wcase moins\@nil\fi}%
1564        \fc@nbrstr@preamble
1565        \fc@@nbrstrfrench@inner
1566        \fc@nbrstr@postamble
```

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

```
1567        \def\@tempb##1{\def\@tempa{\def#2{##1}}}%
1568        \expandafter\@tempb\expandafter{\@tempa}%
1569        \expandafter
1570    }\@tempa
1571 }%
1572 \global\let\@@numberstringfrench\@@numberstringfrench
```

@@nbrstrfrench@inner Common 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.

1573 `\def\fc@@nbrstrfrench@inner{%`

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

1574   `\count0=\fc@max@weight`
1575   `\divide\count0 by 3 %`
1576   `\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`.

1577   `\fc@intpart@find@last{\count5 }%`
1578   `\count6\count5 %`
1579   `\divide\count6 3 %`
1580   `\multiply\count6 3 %`
1581   `\count8=0 %`
1582   `\loop`

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

1583   `\count1\count0 %`
1584   `\advance\count1 by 2 %`
1585   `\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`.

1586   `\fc@poweroften\@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`

1587   `\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.

1588   `\fc@muladdfrench\@tempt\@tempd\@tempb`

Then iterate.

1589   `\count8\count9 %`
1590   `\advance\count0 by -3 %`
1591   `\ifnum\count6>\count0 \else`
1592   `\repeat`
1593 `}%`
1594 `\global\let\fc@@nbrstrfrench@inner\fc@@nbrstrfrench@inner`

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

1595 `\ifcsundef{@@ordinalstringfrench}{}{%`
1596 `\PackageError{fmtcount}{Duplicate definition}{Redefinition of macro`
1597 `'@@ordinalstringfrench'}}`

Arguments are as follows:

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

```
1598 \def\@@ordinalstringfrench#1#2{%
1599   {%
```

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

```
1600     \edef\@tempa{#1}%
1601     \expandafter\fc@number@parser\expandafter{\@tempa}%
1602     \ifnum\fc@min@weight<0 %
1603       \PackageError{fmtcount}{Out of range}%
1604         {This macro does not work with fractional numbers}%
1605     \fi
1606     \ifnum\fc@sign@case>0 %
1607       \PackageError{fmtcount}{Out of range}%
1608         {This macro does with negative or explicitly marked as positive numbers}%
1609     \fi
```

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

```
1610     \ifnum\fc@max@weight=0 %
1611       \ifnum\csname fc@digit@0\endcsname=1 %
1612         \count0=1 %
1613       \else
1614         \count0=0 %
1615       \fi
1616     \else
1617       \count0=0 %
1618     \fi
1619     \ifnum\count0=1 %

1620       \protected@edef\@tempa{\expandafter\fc@wcase\fc@first\@nil}%
1621     \else
```

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

```
1622       \def\@tempa##1{%
1623         \expandafter\edef\csname fc@frenchoptions@##1@plural\endcsname{%
1624           \ifcase\csname fc@frenchoptions@##1@plural\endcsname\space
1625           0% 0: always => always
1626           \or
1627           1% 1: never => never
1628           \or
1629           6% 2: multiple => multiple  ng-last
1630           \or
1631           1% 3: multiple g-last => never
1632           \or
1633           5% 4: multiple l-last => multiple lng-last
1634           \or
1635           5% 5: multiple lng-last => multiple lng-last
```

56

```
1636            \or
1637            6% 6: multiple ng-last => multiple ng-last
1638            \fi
1639          }%
1640        }%
1641        \@tempa{vingt}%
1642        \@tempa{cent}%
1643        \@tempa{mil}%
1644        \@tempa{n-illion}%
1645        \@tempa{n-illiard}%
```

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

```
1646        \let\fc@wcase@save\fc@wcase
1647        \def\fc@wcase{\noexpand\fc@wcase}%
1648        \def\@nil{\noexpand\@nil}%
```

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

```
1649        \let\@tempa\@empty
1650        \fc@@nbrstrfrench@inner
```

Now restore `\fc@wcase`

```
1651      \let\fc@wcase\fc@wcase@save
```

Now we add the "ième" ending

```
1652      \expandafter\fc@get@last@word\expandafter{\@tempa}\@tempb\@tempc
1653      \expandafter\fc@get@last@letter\expandafter{\@tempc}\@tempd\@tempe
1654      \def\@tempf{e}%
1655      \ifx\@tempe\@tempf
1656        \protected@edef\@tempa{\@tempb\expandafter\fc@wcase\@tempd i\protect\`eme\@nil}%
1657      \else
1658        \def\@tempf{q}%
1659        \ifx\@tempe\@tempf
1660          \protected@edef\@tempa{\@tempb\expandafter\fc@wcase\@tempd qui\protect\`eme\@nil}%
1661        \else
1662          \def\@tempf{f}%
1663          \ifx\@tempe\@tempf
1664            \protected@edef\@tempa{\@tempb\expandafter\fc@wcase\@tempd vi\protect\`eme\@nil}%
1665          \else
1666            \protected@edef\@tempa{\@tempb\expandafter\fc@wcase\@tempc i\protect\`eme\@nil}%
1667          \fi
1668        \fi
1669      \fi
1670    \fi
```

Apply `\fc@gcase` to the result.

```
1671    \fc@apply@gcase
```

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

```
1672    \def\@tempb##1{\def\@tempa{\def#2{##1}}}%
1673    \expandafter\@tempb\expandafter{\@tempa}%
1674    \expandafter
1675  }\@tempa
```

```
1676 }%
1677 \global\let\@@ordinalstringfrench\@@ordinalstringfrench
```

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

```
1678 \newcommand*\fc@frenchoptions@setdefaults{%
1679   \csname KV@fcfrench@all plural\endcsname{reformed}%
1680   \fc@gl@def\fc@frenchoptions@submillion@dos{-}%
1681   \fc@gl@let\fc@frenchoptions@supermillion@dos\space
1682   \fc@gl@let\fc@u@in@duo\@empty% Could be 'u'
1683   % \fc@gl@let\fc@poweroften\fc@@pot@longscalefrench
1684   \fc@gl@let\fc@poweroften\fc@@pot@recursivefrench
1685   \fc@gl@def\fc@longscale@nilliard@upto{0}% infinity
1686   \fc@gl@def\fc@frenchoptions@mil@plural@mark{le}%
1687 }%
1688 \global\let\fc@frenchoptions@setdefaults\fc@frenchoptions@setdefaults
1689 {%
1690   \let\fc@gl@def\gdef
1691   \def\fc@gl@let{\global\let}%
1692   \fc@frenchoptions@setdefaults
1693 }%
```

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

```
1694 \gdef\@ordinalstringMfrench{\csuse{@ordinalstringMfrench\fmtcount@french}}%
1695 \gdef\@ordinalstringFfrench{\csuse{@ordinalstringFfrench\fmtcount@french}}%
1696 \gdef\@OrdinalstringMfrench{\csuse{@OrdinalstringMfrench\fmtcount@french}}%
1697 \gdef\@OrdinalstringFfrench{\csuse{@OrdinalstringFfrench\fmtcount@french}}%
1698 \gdef\@numberstringMfrench{\csuse{@numberstringMfrench\fmtcount@french}}%
1699 \gdef\@numberstringFfrench{\csuse{@numberstringFfrench\fmtcount@french}}%
1700 \gdef\@NumberstringMfrench{\csuse{@NumberstringMfrench\fmtcount@french}}%
1701 \gdef\@NumberstringFfrench{\csuse{@NumberstringFfrench\fmtcount@french}}%
```

9.1.6 fc-frenchb.def

```
1702 \ProvidesFCLanguage{frenchb}[2013/08/17]%
1703 \FCloadlang{french}%
```

Set frenchb to be equivalent to french.

```
1704 \global\let\@ordinalMfrenchb=\@ordinalMfrench
1705 \global\let\@ordinalFfrenchb=\@ordinalFfrench
1706 \global\let\@ordinalNfrenchb=\@ordinalNfrench
1707 \global\let\@numberstringMfrenchb=\@numberstringMfrench
1708 \global\let\@numberstringFfrenchb=\@numberstringFfrench
1709 \global\let\@numberstringNfrenchb=\@numberstringNfrench
1710 \global\let\@NumberstringMfrenchb=\@NumberstringMfrench
1711 \global\let\@NumberstringFfrenchb=\@NumberstringFfrench
1712 \global\let\@NumberstringNfrenchb=\@NumberstringNfrench
1713 \global\let\@ordinalstringMfrenchb=\@ordinalstringMfrench
1714 \global\let\@ordinalstringFfrenchb=\@ordinalstringFfrench
1715 \global\let\@ordinalstringNfrenchb=\@ordinalstringNfrench
1716 \global\let\@OrdinalstringMfrenchb=\@OrdinalstringMfrench
1717 \global\let\@OrdinalstringFfrenchb=\@OrdinalstringFfrench
```

```
1718 \global\let\@OrdinalstringNfrenchb=\@OrdinalstringNfrench
```

9.1.7 fc-german.def

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

```
1719 \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:

```
1720 \newcommand{\@ordinalMgerman}[2]{%
1721   \edef#2{\number#1\relax.}%
1722 }%
1723 \global\let\@ordinalMgerman\@ordinalMgerman
```

Feminine:

```
1724 \newcommand{\@ordinalFgerman}[2]{%
1725   \edef#2{\number#1\relax.}%
1726 }%
1727 \global\let\@ordinalFgerman\@ordinalFgerman
```

Neuter:

```
1728 \newcommand{\@ordinalNgerman}[2]{%
1729   \edef#2{\number#1\relax.}%
1730 }%
1731 \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):

```
1732 \newcommand*\@@unitstringgerman[1]{%
1733   \ifcase#1%
1734     null%
1735     \or ein%
1736     \or zwei%
1737     \or drei%
1738     \or vier%
1739     \or f\"unf%
1740     \or sechs%
1741     \or sieben%
1742     \or acht%
1743     \or neun%
1744   \fi
1745 }%
1746 \global\let\@@unitstringgerman\@@unitstringgerman
```

Tens (argument must go from 1 to 10):

```
1747 \newcommand*\@@tenstringgerman[1]{%
1748   \ifcase#1%
1749     \or zehn%
1750     \or zwanzig%
1751     \or drei{\ss}ig%
1752     \or vierzig%
1753     \or f\"unfzig%
```

```
1754     \or sechzig%
1755     \or siebzig%
1756     \or achtzig%
1757     \or neunzig%
1758     \or einhundert%
1759   \fi
1760 }%
1761 \global\let\@@tenstringgerman\@@tenstringgerman
```

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

```
1762 \providecommand*{\einhundert}{einhundert}%
1763 \providecommand*{\eintausend}{eintausend}%
1764 \global\let\einhundert\einhundert
1765 \global\let\eintausend\eintausend
```

Teens:

```
1766 \newcommand*\@@teenstringgerman[1]{%
1767   \ifcase#1%
1768     zehn%
1769     \or elf%
1770     \or zw\"olf%
1771     \or dreizehn%
1772     \or vierzehn%
1773     \or f\"unfzehn%
1774     \or sechzehn%
1775     \or siebzehn%
1776     \or achtzehn%
1777     \or neunzehn%
1778   \fi
1779 }%
1780 \global\let\@@teenstringgerman\@@teenstringgerman
```

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

```
1781 \newcommand*{\@numberstringMgerman}[2]{%
1782   \let\@unitstring=\@@unitstringgerman
1783   \let\@teenstring=\@@teenstringgerman
1784   \let\@tenstring=\@@tenstringgerman
1785   \@@numberstringgerman{#1}{#2}%
1786 }%
1787 \global\let\@numberstringMgerman\@numberstringMgerman
```

Feminine and neuter forms:

```
1788 \global\let\@numberstringFgerman=\@numberstringMgerman
1789 \global\let\@numberstringNgerman=\@numberstringMgerman
```

As above, but initial letters in upper case:

```
1790 \newcommand*{\@NumberstringMgerman}[2]{%
1791   \@numberstringMgerman{#1}{\@@num@str}%
1792   \edef#2{\noexpand\MakeUppercase\expandonce\@@num@str}%
1793 }%
1794 \global\let\@NumberstringMgerman\@NumberstringMgerman
```

Feminine and neuter form:

```
1795 \global\let\@NumberstringFgerman=\@NumberstringMgerman
1796 \global\let\@NumberstringNgerman=\@NumberstringMgerman
```

As above, but for ordinals.

```
1797 \newcommand*{\@ordinalstringMgerman}[2]{%
1798   \let\@unitthstring=\@@unitthstringMgerman
1799   \let\@teenthstring=\@@teenthstringMgerman
1800   \let\@tenthstring=\@@tenthstringMgerman
1801   \let\@unitstring=\@@unitstringgerman
1802   \let\@teenstring=\@@teenstringgerman
1803   \let\@tenstring=\@@tenstringgerman
1804   \def\@thousandth{tausendster}%
1805   \def\@hundredth{hundertster}%
1806   \@@ordinalstringgerman{#1}{#2}%
1807 }%
1808 \global\let\@ordinalstringMgerman\@ordinalstringMgerman
```

Feminine form:

```
1809 \newcommand*{\@ordinalstringFgerman}[2]{%
1810   \let\@unitthstring=\@@unitthstringFgerman
1811   \let\@teenthstring=\@@teenthstringFgerman
1812   \let\@tenthstring=\@@tenthstringFgerman
1813   \let\@unitstring=\@@unitstringgerman
1814   \let\@teenstring=\@@teenstringgerman
1815   \let\@tenstring=\@@tenstringgerman
1816   \def\@thousandth{tausendste}%
1817   \def\@hundredth{hundertste}%
1818   \@@ordinalstringgerman{#1}{#2}%
1819 }%
1820 \global\let\@ordinalstringFgerman\@ordinalstringFgerman
```

Neuter form:

```
1821 \newcommand*{\@ordinalstringNgerman}[2]{%
1822   \let\@unitthstring=\@@unitthstringNgerman
1823   \let\@teenthstring=\@@teenthstringNgerman
1824   \let\@tenthstring=\@@tenthstringNgerman
1825   \let\@unitstring=\@@unitstringgerman
1826   \let\@teenstring=\@@teenstringgerman
1827   \let\@tenstring=\@@tenstringgerman
1828   \def\@thousandth{tausendstes}%
1829   \def\@hundredth{hunderstes}%
1830   \@@ordinalstringgerman{#1}{#2}%
1831 }%
1832 \global\let\@ordinalstringNgerman\@ordinalstringNgerman
```

As above, but with initial letters in upper case.

```
1833 \newcommand*{\@OrdinalstringMgerman}[2]{%
1834 \@ordinalstringMgerman{#1}{\@@num@str}%
1835 \edef#2{\noexpand\MakeUppercase\expandonce\@@num@str}%
1836 }%
```

```
1837 \global\let\@OrdinalstringMgerman\@OrdinalstringMgerman
```

Feminine form:

```
1838 \newcommand*{\@OrdinalstringFgerman}[2]{%
1839   \@ordinalstringFgerman{#1}{\@@num@str}%
1840   \edef#2{\noexpand\MakeUppercase\expandonce\@@num@str}%
1841 }%
1842 \global\let\@OrdinalstringFgerman\@OrdinalstringFgerman
```

Neuter form:

```
1843 \newcommand*{\@OrdinalstringNgerman}[2]{%
1844   \@ordinalstringNgerman{#1}{\@@num@str}%
1845   \edef#2{\noexpand\MakeUppercase\expandonce\@@num@str}%
1846 }%
1847 \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:

```
1848 \newcommand*\@@unitthstringMgerman[1]{%
1849   \ifcase#1%
1850     nullter%
1851     \or erster%
1852     \or zweiter%
1853     \or dritter%
1854     \or vierter%
1855     \or f\"unfter%
1856     \or sechster%
1857     \or siebter%
1858     \or achter%
1859     \or neunter%
1860   \fi
1861 }%
1862 \global\let\@@unitthstringMgerman\@@unitthstringMgerman
```

Tens:

```
1863 \newcommand*\@@tenthstringMgerman[1]{%
1864   \ifcase#1%
1865     \or zehnter%
1866     \or zwanzigster%
1867     \or drei{\ss}igster%
1868     \or vierzigster%
1869     \or f\"unfzigster%
1870     \or sechzigster%
1871     \or siebzigster%
1872     \or achtzigster%
1873     \or neunzigster%
1874   \fi
1875 }%
1876 \global\let\@@tenthstringMgerman\@@tenthstringMgerman
```

Teens:

```
1877 \newcommand*\@@teenthstringMgerman[1]{%
1878   \ifcase#1%
1879     zehnter%
1880     \or elfter%
1881     \or zw\"olfter%
1882     \or dreizehnter%
1883     \or vierzehnter%
1884     \or f\"unfzehnter%
1885     \or sechzehnter%
1886     \or siebzehnter%
1887     \or achtzehnter%
1888     \or neunzehnter%
1889   \fi
1890 }%
1891 \global\let\@@teenthstringMgerman\@@teenthstringMgerman
```

Units (feminine):

```
1892 \newcommand*\@@unitthstringFgerman[1]{%
1893   \ifcase#1%
1894     nullte%
1895     \or erste%
1896     \or zweite%
1897     \or dritte%
1898     \or vierte%
1899     \or f\"unfte%
1900     \or sechste%
1901     \or siebte%
1902     \or achte%
1903     \or neunte%
1904   \fi
1905 }%
1906 \global\let\@@unitthstringFgerman\@@unitthstringFgerman
```

Tens (feminine):

```
1907 \newcommand*\@@tenthstringFgerman[1]{%
1908   \ifcase#1%
1909     \or zehnte%
1910     \or zwanzigste%
1911     \or drei{\ss}igste%
1912     \or vierzigste%
1913     \or f\"unfzigste%
1914     \or sechzigste%
1915     \or siebzigste%
1916     \or achtzigste%
1917     \or neunzigste%
1918   \fi
1919 }%
1920 \global\let\@@tenthstringFgerman\@@tenthstringFgerman
```

Teens (feminine)

```
1921 \newcommand*\@@teenthstringFgerman[1]{%
```

```
1922  \ifcase#1%
1923    zehnte%
1924    \or elfte%
1925    \or zw\"olfte%
1926    \or dreizehnte%
1927    \or vierzehnte%
1928    \or f\"unfzehnte%
1929    \or sechzehnte%
1930    \or siebzehnte%
1931    \or achtzehnte%
1932    \or neunzehnte%
1933  \fi
1934 }%
1935 \global\let\@@teenthstringFgerman\@@teenthstringFgerman
```

Units (neuter):

```
1936 \newcommand*\@@unitthstringNgerman[1]{%
1937  \ifcase#1%
1938    nulltes%
1939    \or erstes%
1940    \or zweites%
1941    \or drittes%
1942    \or viertes%
1943    \or f\"unftes%
1944    \or sechstes%
1945    \or siebtes%
1946    \or achtes%
1947    \or neuntes%
1948  \fi
1949 }%
1950 \global\let\@@unitthstringNgerman\@@unitthstringNgerman
```

Tens (neuter):

```
1951 \newcommand*\@@tenthstringNgerman[1]{%
1952  \ifcase#1%
1953    \or zehntes%
1954    \or zwanzigstes%
1955    \or drei{\ss}igstes%
1956    \or vierzigstes%
1957    \or f\"unfzigstes%
1958    \or sechzigstes%
1959    \or siebzigstes%
1960    \or achtzigstes%
1961    \or neunzigstes%
1962  \fi
1963 }%
1964 \global\let\@@tenthstringNgerman\@@tenthstringNgerman
```

Teens (neuter)

```
1965 \newcommand*\@@teenthstringNgerman[1]{%
1966  \ifcase#1%
```

```
1967     zehntes%
1968     \or elftes%
1969     \or zw\"olftes%
1970     \or dreizehntes%
1971     \or vierzehntes%
1972     \or f\"unfzehntes%
1973     \or sechzehntes%
1974     \or siebzehntes%
1975     \or achtzehntes%
1976     \or neunzehntes%
1977   \fi
1978 }%
1979 \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.

```
1980 \newcommand*\@@numberunderhundredgerman[2]{%
1981 \ifnum#1<10\relax
1982   \ifnum#1>0\relax
1983     \eappto#2{\@unitstring{#1}}%
1984   \fi
1985 \else
1986   \@tmpstrctr=#1\relax
1987   \@FCmodulo{\@tmpstrctr}{10}%
1988   \ifnum#1<20\relax
1989     \eappto#2{\@teenstring{\@tmpstrctr}}%
1990   \else
1991     \ifnum\@tmpstrctr=0\relax
1992     \else
1993       \eappto#2{\@unitstring{\@tmpstrctr}und}%
1994     \fi
1995     \@tmpstrctr=#1\relax
1996     \divide\@tmpstrctr by 10\relax
1997     \eappto#2{\@tenstring{\@tmpstrctr}}%
1998   \fi
1999 \fi
2000 }%
2001 \global\let\@@numberunderhundredgerman\@@numberunderhundredgerman
```

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

```
2002 \newcommand*\@@numberstringgerman[2]{%
2003 \ifnum#1>99999\relax
2004   \PackageError{fmtcount}{Out of range}%
2005   {This macro only works for values less than 100000}%
2006 \else
2007   \ifnum#1<0\relax
2008     \PackageError{fmtcount}{Negative numbers not permitted}%
2009     {This macro does not work for negative numbers, however
2010      you can try typing "minus" first, and then pass the modulus of
```

```
2011      this number}%
2012   \fi
2013 \fi
2014 \def#2{}%
2015 \@strctr=#1\relax \divide\@strctr by 1000\relax
2016 \ifnum\@strctr>1\relax
```

#1 is ≥ 2000, `\@strctr` now contains the number of thousands

```
2017   \@@numberunderhundredgerman{\@strctr}{#2}%
2018   \appto#2{tausend}%
2019 \else
```

#1 lies in range [1000,1999]

```
2020   \ifnum\@strctr=1\relax
2021     \eappto#2{\eintausend}%
2022   \fi
2023 \fi
2024 \@strctr=#1\relax
2025 \@FCmodulo{\@strctr}{1000}%
2026 \divide\@strctr by 100\relax
2027 \ifnum\@strctr>1\relax
```

now dealing with number in range [200,999]

```
2028   \eappto#2{\@unitstring{\@strctr}hundert}%
2029 \else
2030   \ifnum\@strctr=1\relax
```

dealing with number in range [100,199]

```
2031     \ifnum#1>1000\relax
```

if original number > 1000, use einhundert

```
2032       \appto#2{einhundert}%
2033     \else
```

otherwise use `\einhundert`

```
2034       \eappto#2{\einhundert}%
2035     \fi
2036   \fi
2037 \fi
2038 \@strctr=#1\relax
2039 \@FCmodulo{\@strctr}{100}%
2040 \ifnum#1=0\relax
2041   \def#2{null}%
2042 \else
2043   \ifnum\@strctr=1\relax
2044     \appto#2{eins}%
2045   \else
2046     \@@numberunderhundredgerman{\@strctr}{#2}%
2047   \fi
2048 \fi
2049 }%
2050 \global\let\@@numberstringgerman\@@numberstringgerman
```

As above, but for ordinals

```
2051 \newcommand*\@@numberunderhundredthgerman[2]{%
2052 \ifnum#1<10\relax
2053   \eappto#2{\@unitthstring{#1}}%
2054 \else
2055   \@tmpstrctr=#1\relax
2056   \@FCmodulo{\@tmpstrctr}{10}%
2057   \ifnum#1<20\relax
2058     \eappto#2{\@teenthstring{\@tmpstrctr}}%
2059   \else
2060     \ifnum\@tmpstrctr=0\relax
2061     \else
2062       \eappto#2{\@unitstring{\@tmpstrctr}und}%
2063     \fi
2064     \@tmpstrctr=#1\relax
2065     \divide\@tmpstrctr by 10\relax
2066     \eappto#2{\@tenthstring{\@tmpstrctr}}%
2067   \fi
2068 \fi
2069 }%
2070 \global\let\@@numberunderhundredthgerman\@@numberunderhundredthgerman

2071 \newcommand*\@@ordinalstringgerman[2]{%
2072 \ifnum#1>99999\relax
2073   \PackageError{fmtcount}{Out of range}%
2074   {This macro only works for values less than 100000}%
2075 \else
2076   \ifnum#1<0\relax
2077     \PackageError{fmtcount}{Negative numbers not permitted}%
2078     {This macro does not work for negative numbers, however
2079     you can try typing "minus" first, and then pass the modulus of
2080     this number}%
2081   \fi
2082 \fi
2083 \def#2{}%
2084 \@strctr=#1\relax \divide\@strctr by 1000\relax
2085 \ifnum\@strctr>1\relax
```

#1 is ≥ 2000, \@strctr now contains the number of thousands

```
2086 \@@numberunderhundredgerman{\@strctr}{#2}%
```

is that it, or is there more?

```
2087   \@tmpstrctr=#1\relax \@FCmodulo{\@tmpstrctr}{1000}%
2088   \ifnum\@tmpstrctr=0\relax
2089     \eappto#2{\@thousandth}%
2090   \else
2091     \appto#2{tausend}%
2092   \fi
2093 \else
```

#1 lies in range [1000,1999]

```
2094  \ifnum\@strctr=1\relax
2095    \ifnum#1=1000\relax
2096      \eappto#2{\@thousandth}%
2097    \else
2098      \eappto#2{\eintausend}%
2099    \fi
2100  \fi
2101 \fi
2102 \@strctr=#1\relax
2103 \@FCmodulo{\@strctr}{1000}%
2104 \divide\@strctr by 100\relax
2105 \ifnum\@strctr>1\relax
```

now dealing with number in range [200,999] is that it, or is there more?

```
2106    \@tmpstrctr=#1\relax \@FCmodulo{\@tmpstrctr}{100}%
2107    \ifnum\@tmpstrctr=0\relax
2108      \ifnum\@strctr=1\relax
2109        \eappto#2{\@hundredth}%
2110      \else
2111        \eappto#2{\@unitstring{\@strctr}\@hundredth}%
2112      \fi
2113    \else
2114      \eappto#2{\@unitstring{\@strctr}hundert}%
2115    \fi
2116 \else
2117    \ifnum\@strctr=1\relax
```

dealing with number in range [100,199] is that it, or is there more?

```
2118      \@tmpstrctr=#1\relax \@FCmodulo{\@tmpstrctr}{100}%
2119      \ifnum\@tmpstrctr=0\relax
2120        \eappto#2{\@hundredth}%
2121      \else
2122      \ifnum#1>1000\relax
2123        \appto#2{einhundert}%
2124      \else
2125        \eappto#2{\einhundert}%
2126      \fi
2127      \fi
2128    \fi
2129 \fi
2130 \@strctr=#1\relax
2131 \@FCmodulo{\@strctr}{100}%
2132 \ifthenelse{\@strctr=0 \and #1>0}{}{%
2133 \@@numberunderhundredthgerman{\@strctr}{#2}%
2134 }%
2135 }%
2136 \global\let\@@ordinalstringgerman\@@ordinalstringgerman
```

Load fc-germanb.def if not already loaded

```
2137 \FCloadlang{germanb}%
```

9.1.8 fc-germanb.def

2138 `\ProvidesFCLanguage{germanb}[2013/08/17]%`

Load fc-german.def if not already loaded

2139 `\FCloadlang{german}%`

Set germanb to be equivalent to german.

2140 `\global\let\@ordinalMgermanb=\@ordinalMgerman`
2141 `\global\let\@ordinalFgermanb=\@ordinalFgerman`
2142 `\global\let\@ordinalNgermanb=\@ordinalNgerman`
2143 `\global\let\@numberstringMgermanb=\@numberstringMgerman`
2144 `\global\let\@numberstringFgermanb=\@numberstringFgerman`
2145 `\global\let\@numberstringNgermanb=\@numberstringNgerman`
2146 `\global\let\@NumberstringMgermanb=\@NumberstringMgerman`
2147 `\global\let\@NumberstringFgermanb=\@NumberstringFgerman`
2148 `\global\let\@NumberstringNgermanb=\@NumberstringNgerman`
2149 `\global\let\@ordinalstringMgermanb=\@ordinalstringMgerman`
2150 `\global\let\@ordinalstringFgermanb=\@ordinalstringFgerman`
2151 `\global\let\@ordinalstringNgermanb=\@ordinalstringNgerman`
2152 `\global\let\@OrdinalstringMgermanb=\@OrdinalstringMgerman`
2153 `\global\let\@OrdinalstringFgermanb=\@OrdinalstringFgerman`
2154 `\global\let\@OrdinalstringNgermanb=\@OrdinalstringNgerman`

9.1.9 fc-italian

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

2155 `\ProvidesFCLanguage{italian}[2013/08/17]`
2156
2157 `\RequirePackage{itnumpar}`
2158
2159 `\newcommand{\@numberstringMitalian}[2]{%`
2160 `  \edef#2{\noexpand\printnumeroinparole{#1}}%`
2161 `}`
2162 `\global\let\@numberstringMitalian\@numberstringMitalian`
2163
2164 `\newcommand{\@numberstringFitalian}[2]{%`
2165 `  \edef#2{\noexpand\printnumeroinparole{#1}}}`
2166
2167 `\global\let\@numberstringFitalian\@numberstringFitalian`
2168
2169 `\newcommand{\@NumberstringMitalian}[2]{%`
2170 `  \edef#2{\noexpand\printNumeroinparole{#1}}%`
2171 `}`
2172 `\global\let\@NumberstringMitalian\@NumberstringMitalian`
2173
2174 `\newcommand{\@NumberstringFitalian}[2]{%`
2175 `  \edef#2{\noexpand\printNumeroinparole{#1}}%`
2176 `}`
2177 `\global\let\@NumberstringFitalian\@NumberstringFitalian`
2178

```
2179 \newcommand{\@ordinalstringMitalian}[2]{%
2180   \edef#2{\noexpand\printordinalem{#1}}%
2181 }
2182 \global\let\@ordinalstringMitalian\@ordinalstringMitalian
2183
2184 \newcommand{\@ordinalstringFitalian}[2]{%
2185   \edef#2{\noexpand\printordinalef{#1}}%
2186 }
2187 \global\let\@ordinalstringFitalian\@ordinalstringFitalian
2188
2189 \newcommand{\@OrdinalstringMitalian}[2]{%
2190   \edef#2{\noexpand\printOrdinalem{#1}}%
2191 }
2192 \global\let\@OrdinalstringMitalian\@OrdinalstringMitalian
2193
2194 \newcommand{\@OrdinalstringFitalian}[2]{%
2195   \edef#2{\noexpand\printOrdinalef{#1}}%
2196 }
2197 \global\let\@OrdinalstringFitalian\@OrdinalstringFitalian
2198
2199 \newcommand{\@ordinalMitalian}[2]{%
2200   \edef#2{#1\relax\noexpand\fmtord{o}}}
2201
2202 \global\let\@ordinalMitalian\@ordinalMitalian
2203
2204 \newcommand{\@ordinalFitalian}[2]{%
2205   \edef#2{#1\relax\noexpand\fmtord{a}}}
2206 \global\let\@ordinalFitalian\@ordinalFitalian
```

9.1.10 fc-ngerman.def

```
2207 \ProvidesFCLanguage{ngerman}[2012/06/18]%
2208 \FCloadlang{german}%
2209 \FCloadlang{ngermanb}%
```

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

```
2210 \global\let\@ordinalMngerman=\@ordinalMgerman
2211 \global\let\@ordinalFngerman=\@ordinalFgerman
2212 \global\let\@ordinalNngerman=\@ordinalNgerman
2213 \global\let\@numberstringMngerman=\@numberstringMgerman
2214 \global\let\@numberstringFngerman=\@numberstringFgerman
2215 \global\let\@numberstringNngerman=\@numberstringNgerman
2216 \global\let\@NumberstringMngerman=\@NumberstringMgerman
2217 \global\let\@NumberstringFngerman=\@NumberstringFgerman
2218 \global\let\@NumberstringNngerman=\@NumberstringNgerman
2219 \global\let\@ordinalstringMngerman=\@ordinalstringMgerman
2220 \global\let\@ordinalstringFngerman=\@ordinalstringFgerman
2221 \global\let\@ordinalstringNngerman=\@ordinalstringNgerman
2222 \global\let\@OrdinalstringMngerman=\@OrdinalstringMgerman
```

```
2223 \global\let\@OrdinalstringFngerman=\@OrdinalstringFgerman
2224 \global\let\@OrdinalstringNngerman=\@OrdinalstringNgerman
```

9.1.11 fc-ngermanb.def

```
2225 \ProvidesFCLanguage{ngermanb}[2013/08/17]%
2226 \FCloadlang{german}%
```

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

```
2227 \global\let\@ordinalMngermanb=\@ordinalMgerman
2228 \global\let\@ordinalFngermanb=\@ordinalFgerman
2229 \global\let\@ordinalNngermanb=\@ordinalNgerman
2230 \global\let\@numberstringMngermanb=\@numberstringMgerman
2231 \global\let\@numberstringFngermanb=\@numberstringFgerman
2232 \global\let\@numberstringNngermanb=\@numberstringNgerman
2233 \global\let\@NumberstringMngermanb=\@NumberstringMgerman
2234 \global\let\@NumberstringFngermanb=\@NumberstringFgerman
2235 \global\let\@NumberstringNngermanb=\@NumberstringNgerman
2236 \global\let\@ordinalstringMngermanb=\@ordinalstringMgerman
2237 \global\let\@ordinalstringFngermanb=\@ordinalstringFgerman
2238 \global\let\@ordinalstringNngermanb=\@ordinalstringNgerman
2239 \global\let\@OrdinalstringMngermanb=\@OrdinalstringMgerman
2240 \global\let\@OrdinalstringFngermanb=\@OrdinalstringFgerman
2241 \global\let\@OrdinalstringNngermanb=\@OrdinalstringNgerman
```

Load fc-ngerman.def if not already loaded

```
2242 \FCloadlang{ngerman}%
```

9.1.12 fc-portuges.def

Portuguse definitions

```
2243 \ProvidesFCLanguage{portuges}[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. Masculine:

```
2244 \newcommand*\@ordinalMportuges[2]{%
2245   \ifnum#1=0\relax
2246     \edef#2{\number#1}%
2247   \else
2248     \edef#2{\number#1\relax\noexpand\fmtord{o}}%
2249   \fi
2250 }%
2251 \global\let\@ordinalMportuges\@ordinalMportuges
```

Feminine:

```
2252 \newcommand*\@ordinalFportuges[2]{%
2253   \ifnum#1=0\relax
2254     \edef#2{\number#1}%
2255   \else
2256     \edef#2{\number#1\relax\noexpand\fmtord{a}}%
2257   \fi
2258 }%
```

```
2259 \global\let\@ordinalFportuges\@ordinalFportuges
```

Make neuter same as masculine:

```
2260 \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):

```
2261 \newcommand*\@@unitstringportuges[1]{%
2262   \ifcase#1\relax
2263     zero%
2264     \or um%
2265     \or dois%
2266     \or tr\^es%
2267     \or quatro%
2268     \or cinco%
2269     \or seis%
2270     \or sete%
2271     \or oito%
2272     \or nove%
2273   \fi
2274 }%
2275 \global\let\@@unitstringportuges\@@unitstringportuges
2276 %   \end{macrocode}
2277 % As above, but for feminine:
2278 %   \begin{macrocode}
2279 \newcommand*\@@unitstringFportuges[1]{%
2280   \ifcase#1\relax
2281     zero%
2282     \or uma%
2283     \or duas%
2284     \or tr\^es%
2285     \or quatro%
2286     \or cinco%
2287     \or seis%
2288     \or sete%
2289     \or oito%
2290     \or nove%
2291   \fi
2292 }%
2293 \global\let\@@unitstringFportuges\@@unitstringFportuges
```

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

```
2294 \newcommand*\@@tenstringportuges[1]{%
2295   \ifcase#1\relax
2296     \or dez%
2297     \or vinte%
2298     \or trinta%
2299     \or quarenta%
2300     \or cinq\"uenta%
2301     \or sessenta%
2302     \or setenta%
```

```
2303      \or oitenta%
2304      \or noventa%
2305      \or cem%
2306    \fi
2307 }%
2308 \global\let\@@tenstringportuges\@@tenstringportuges
```

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

```
2309 \newcommand*\@@teenstringportuges[1]{%
2310    \ifcase#1\relax
2311      dez%
2312    \or onze%
2313    \or doze%
2314    \or treze%
2315    \or quatorze%
2316    \or quinze%
2317    \or dezesseis%
2318    \or dezessete%
2319    \or dezoito%
2320    \or dezenove%
2321    \fi
2322 }%
2323 \global\let\@@teenstringportuges\@@teenstringportuges
```

Hundreds:

```
2324 \newcommand*\@@hundredstringportuges[1]{%
2325    \ifcase#1\relax
2326    \or cento%
2327    \or duzentos%
2328    \or trezentos%
2329    \or quatrocentos%
2330    \or quinhentos%
2331    \or seiscentos%
2332    \or setecentos%
2333    \or oitocentos%
2334    \or novecentos%
2335    \fi
2336 }%
2337 \global\let\@@hundredstringportuges\@@hundredstringportuges
```

Hundreds (feminine):

```
2338 \newcommand*\@@hundredstringFportuges[1]{%
2339    \ifcase#1\relax
2340    \or cento%
2341    \or duzentas%
2342    \or trezentas%
2343    \or quatrocentas%
2344    \or quinhentas%
2345    \or seiscentas%
2346    \or setecentas%
2347    \or oitocentas%
```

```
2348      \or novecentas%
2349   \fi
2350 }%
2351 \global\let\@@hundredstringFportuges\@@hundredstringFportuges
```

Units (initial letter in upper case):

```
2352 \newcommand*\@@Unitstringportuges[1]{%
2353   \ifcase#1\relax
2354     Zero%
2355     \or Um%
2356     \or Dois%
2357     \or Tr\^es%
2358     \or Quatro%
2359     \or Cinco%
2360     \or Seis%
2361     \or Sete%
2362     \or Oito%
2363     \or Nove%
2364   \fi
2365 }%
2366 \global\let\@@Unitstringportuges\@@Unitstringportuges
```

As above, but feminine:

```
2367 \newcommand*\@@UnitstringFportuges[1]{%
2368   \ifcase#1\relax
2369     Zera%
2370     \or Uma%
2371     \or Duas%
2372     \or Tr\^es%
2373     \or Quatro%
2374     \or Cinco%
2375     \or Seis%
2376     \or Sete%
2377     \or Oito%
2378     \or Nove%
2379   \fi
2380 }%
2381 \global\let\@@UnitstringFportuges\@@UnitstringFportuges
```

Tens (with initial letter in upper case):

```
2382 \newcommand*\@@Tenstringportuges[1]{%
2383   \ifcase#1\relax
2384     \or Dez%
2385     \or Vinte%
2386     \or Trinta%
2387     \or Quarenta%
2388     \or Cinq\"uenta%
2389     \or Sessenta%
2390     \or Setenta%
2391     \or Oitenta%
2392     \or Noventa%
```

```
2393        \or Cem%
2394     \fi
2395 }%
2396 \global\let\@@Tenstringportuges\@@Tenstringportuges
```

Teens (with initial letter in upper case):

```
2397 \newcommand*\@@Teenstringportuges[1]{%
2398     \ifcase#1\relax
2399        Dez%
2400        \or Onze%
2401        \or Doze%
2402        \or Treze%
2403        \or Quatorze%
2404        \or Quinze%
2405        \or Dezesseis%
2406        \or Dezessete%
2407        \or Dezoito%
2408        \or Dezenove%
2409     \fi
2410 }%
2411 \global\let\@@Teenstringportuges\@@Teenstringportuges
```

Hundreds (with initial letter in upper case):

```
2412 \newcommand*\@@Hundredstringportuges[1]{%
2413     \ifcase#1\relax
2414        \or Cento%
2415        \or Duzentos%
2416        \or Trezentos%
2417        \or Quatrocentos%
2418        \or Quinhentos%
2419        \or Seiscentos%
2420        \or Setecentos%
2421        \or Oitocentos%
2422        \or Novecentos%
2423     \fi
2424 }%
2425 \global\let\@@Hundredstringportuges\@@Hundredstringportuges
```

As above, but feminine:

```
2426 \newcommand*\@@HundredstringFportuges[1]{%
2427     \ifcase#1\relax
2428        \or Cento%
2429        \or Duzentas%
2430        \or Trezentas%
2431        \or Quatrocentas%
2432        \or Quinhentas%
2433        \or Seiscentas%
2434        \or Setecentas%
2435        \or Oitocentas%
2436        \or Novecentas%
2437     \fi
```

```
2438 }%
2439 \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.)

```
2440 \newcommand*{\@numberstringMportuges}[2]{%
2441   \let\@unitstring=\@@unitstringportuges
2442   \let\@teenstring=\@@teenstringportuges
2443   \let\@tenstring=\@@tenstringportuges
2444   \let\@hundredstring=\@@hundredstringportuges
2445   \def\@hundred{cem}\def\@thousand{mil}%
2446   \def\@andname{e}%
2447   \@@numberstringportuges{#1}{#2}%
2448 }%
2449 \global\let\@numberstringMportuges\@numberstringMportuges
```

As above, but feminine form:

```
2450 \newcommand*{\@numberstringFportuges}[2]{%
2451   \let\@unitstring=\@@unitstringFportuges
2452   \let\@teenstring=\@@teenstringportuges
2453   \let\@tenstring=\@@tenstringportuges
2454   \let\@hundredstring=\@@hundredstringFportuges
2455   \def\@hundred{cem}\def\@thousand{mil}%
2456   \def\@andname{e}%
2457   \@@numberstringportuges{#1}{#2}%
2458 }%
2459 \global\let\@numberstringFportuges\@numberstringFportuges
```

Make neuter same as masculine:

```
2460 \global\let\@numberstringNportuges\@numberstringMportuges
```

As above, but initial letters in upper case:

```
2461 \newcommand*{\@NumberstringMportuges}[2]{%
2462   \let\@unitstring=\@@Unitstringportuges
2463   \let\@teenstring=\@@Teenstringportuges
2464   \let\@tenstring=\@@Tenstringportuges
2465   \let\@hundredstring=\@@Hundredstringportuges
2466   \def\@hundred{Cem}\def\@thousand{Mil}%
2467   \def\@andname{e}%
2468   \@@numberstringportuges{#1}{#2}%
2469 }%
2470 \global\let\@NumberstringMportuges\@NumberstringMportuges
```

As above, but feminine form:

```
2471 \newcommand*{\@NumberstringFportuges}[2]{%
2472   \let\@unitstring=\@@UnitstringFportuges
2473   \let\@teenstring=\@@Teenstringportuges
2474   \let\@tenstring=\@@Tenstringportuges
2475   \let\@hundredstring=\@@HundredstringFportuges
2476   \def\@hundred{Cem}\def\@thousand{Mil}%
```

```
2477    \def\@andname{e}%
2478    \@@numberstringportuges{#1}{#2}%
2479 }%
2480 \global\let\@NumberstringFportuges\@NumberstringFportuges
```

Make neuter same as masculine:

```
2481 \global\let\@NumberstringNportuges\@NumberstringMportuges
```

As above, but for ordinals.

```
2482 \newcommand*{\@ordinalstringMportuges}[2]{%
2483    \let\@unitthstring=\@@unitthstringportuges
2484    \let\@unitstring=\@@unitstringportuges
2485    \let\@teenthstring=\@@teenthstringportuges
2486    \let\@tenthstring=\@@tenthstringportuges
2487    \let\@hundredthstring=\@@hundredthstringportuges
2488    \def\@thousandth{mil\'esimo}%
2489    \@@ordinalstringportuges{#1}{#2}%
2490 }%
2491 \global\let\@ordinalstringMportuges\@ordinalstringMportuges
```

Feminine form:

```
2492 \newcommand*{\@ordinalstringFportuges}[2]{%
2493    \let\@unitthstring=\@@unitthstringFportuges
2494    \let\@unitstring=\@@unitstringFportuges
2495    \let\@teenthstring=\@@teenthstringportuges
2496    \let\@tenthstring=\@@tenthstringFportuges
2497    \let\@hundredthstring=\@@hundredthstringFportuges
2498    \def\@thousandth{mil\'esima}%
2499    \@@ordinalstringportuges{#1}{#2}%
2500 }%
2501 \global\let\@ordinalstringFportuges\@ordinalstringFportuges
```

Make neuter same as masculine:

```
2502 \global\let\@ordinalstringNportuges\@ordinalstringMportuges
```

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

```
2503 \newcommand*{\@OrdinalstringMportuges}[2]{%
2504    \let\@unitthstring=\@@Unitthstringportuges
2505    \let\@unitstring=\@@Unitstringportuges
2506    \let\@teenthstring=\@@teenthstringportuges
2507    \let\@tenthstring=\@@Tenthstringportuges
2508    \let\@hundredthstring=\@@Hundredthstringportuges
2509    \def\@thousandth{Mil\'esimo}%
2510    \@@ordinalstringportuges{#1}{#2}%
2511 }%
2512 \global\let\@OrdinalstringMportuges\@OrdinalstringMportuges
```

Feminine form:

```
2513 \newcommand*{\@OrdinalstringFportuges}[2]{%
2514    \let\@unitthstring=\@@UnitthstringFportuges
2515    \let\@unitstring=\@@UnitstringFportuges
2516    \let\@teenthstring=\@@teenthstringportuges
```

```
2517    \let\@tenthstring=\@@TenthstringFportuges
2518    \let\@hundredthstring=\@@HundredthstringFportuges
2519    \def\@thousandth{Mil\'esima}%
2520    \@@ordinalstringportuges{#1}{#2}%
2521 }%
2522 \global\let\@OrdinalstringFportuges\@OrdinalstringFportuges
```

Make neuter same as masculine:

```
2523 \global\let\@OrdinalstringNportuges\@OrdinalstringMportuges
```

In order to do the ordinals, split into units, teens, tens and hundreds. Units:

```
2524 \newcommand*\@@unitthstringportuges[1]{%
2525    \ifcase#1\relax
2526      zero%
2527      \or primeiro%
2528      \or segundo%
2529      \or terceiro%
2530      \or quarto%
2531      \or quinto%
2532      \or sexto%
2533      \or s\'etimo%
2534      \or oitavo%
2535      \or nono%
2536    \fi
2537 }%
2538 \global\let\@@unitthstringportuges\@@unitthstringportuges
```

Tens:

```
2539 \newcommand*\@@tenthstringportuges[1]{%
2540    \ifcase#1\relax
2541      \or d\'ecimo%
2542      \or vig\'esimo%
2543      \or trig\'esimo%
2544      \or quadrag\'esimo%
2545      \or q\"uinquag\'esimo%
2546      \or sexag\'esimo%
2547      \or setuag\'esimo%
2548      \or octog\'esimo%
2549      \or nonag\'esimo%
2550    \fi
2551 }%
2552 \global\let\@@tenthstringportuges\@@tenthstringportuges
```

Teens:

```
2553 \newcommand*\@@teenthstringportuges[1]{%
2554    \@tenthstring{1}%
2555    \ifnum#1>0\relax
2556      -\@unitthstring{#1}%
2557    \fi
2558 }%
2559 \global\let\@@teenthstringportuges\@@teenthstringportuges
```

Hundreds:

```
2560 \newcommand*\@@hundredthstringportuges[1]{%
2561   \ifcase#1\relax
2562     \or cent\'esimo%
2563     \or ducent\'esimo%
2564     \or trecent\'esimo%
2565     \or quadringent\'esimo%
2566     \or q\"uingent\'esimo%
2567     \or seiscent\'esimo%
2568     \or setingent\'esimo%
2569     \or octingent\'esimo%
2570     \or nongent\'esimo%
2571   \fi
2572 }%
2573 \global\let\@@hundredthstringportuges\@@hundredthstringportuges
```

Units (feminine):

```
2574 \newcommand*\@@unitthstringFportuges[1]{%
2575   \ifcase#1\relax
2576     zero%
2577     \or primeira%
2578     \or segunda%
2579     \or terceira%
2580     \or quarta%
2581     \or quinta%
2582     \or sexta%
2583     \or s\'etima%
2584     \or oitava%
2585     \or nona%
2586   \fi
2587 }%
2588 \global\let\@@unitthstringFportuges\@@unitthstringFportuges
```

Tens (feminine):

```
2589 \newcommand*\@@tenthstringFportuges[1]{%
2590   \ifcase#1\relax
2591     \or d\'ecima%
2592     \or vig\'esima%
2593     \or trig\'esima%
2594     \or quadrag\'esima%
2595     \or q\"uinquag\'esima%
2596     \or sexag\'esima%
2597     \or setuag\'esima%
2598     \or octog\'esima%
2599     \or nonag\'esima%
2600   \fi
2601 }%
2602 \global\let\@@tenthstringFportuges\@@tenthstringFportuges
```

Hundreds (feminine):

```
2603 \newcommand*\@@hundredthstringFportuges[1]{%
2604   \ifcase#1\relax
2605     \or cent\'esima%
2606     \or ducent\'esima%
2607     \or trecent\'esima%
2608     \or quadringent\'esima%
2609     \or q\"uingent\'esima%
2610     \or seiscent\'esima%
2611     \or setingent\'esima%
2612     \or octingent\'esima%
2613     \or nongent\'esima%
2614   \fi
2615 }%
2616 \global\let\@@hundredthstringFportuges\@@hundredthstringFportuges
```

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

```
2617 \newcommand*\@@Unitthstringportuges[1]{%
2618   \ifcase#1\relax
2619     Zero%
2620     \or Primeiro%
2621     \or Segundo%
2622     \or Terceiro%
2623     \or Quarto%
2624     \or Quinto%
2625     \or Sexto%
2626     \or S\'etimo%
2627     \or Oitavo%
2628     \or Nono%
2629   \fi
2630 }%
2631 \global\let\@@Unitthstringportuges\@@Unitthstringportuges
```

Tens:

```
2632 \newcommand*\@@Tenthstringportuges[1]{%
2633   \ifcase#1\relax
2634     \or D\'ecimo%
2635     \or Vig\'esimo%
2636     \or Trig\'esimo%
2637     \or Quadrag\'esimo%
2638     \or Q\"uinquag\'esimo%
2639     \or Sexag\'esimo%
2640     \or Setuag\'esimo%
2641     \or Octog\'esimo%
2642     \or Nonag\'esimo%
2643   \fi
2644 }%
2645 \global\let\@@Tenthstringportuges\@@Tenthstringportuges
```

Hundreds:

```
2646 \newcommand*\@@Hundredthstringportuges[1]{%
2647   \ifcase#1\relax
```

```
2648      \or Cent\'esimo%
2649      \or Ducent\'esimo%
2650      \or Trecent\'esimo%
2651      \or Quadringent\'esimo%
2652      \or Q\"uingent\'esimo%
2653      \or Seiscent\'esimo%
2654      \or Setingent\'esimo%
2655      \or Octingent\'esimo%
2656      \or Nongent\'esimo%
2657   \fi
2658 }%
2659 \global\let\@@Hundredthstringportuges\@@Hundredthstringportuges
```

As above, but feminine. Units:

```
2660 \newcommand*\@@UnitthstringFportuges[1]{%
2661   \ifcase#1\relax
2662     Zera%
2663     \or Primeira%
2664     \or Segunda%
2665     \or Terceira%
2666     \or Quarta%
2667     \or Quinta%
2668     \or Sexta%
2669     \or S\'etima%
2670     \or Oitava%
2671     \or Nona%
2672   \fi
2673 }%
2674 \global\let\@@UnitthstringFportuges\@@UnitthstringFportuges
```

Tens (feminine);

```
2675 \newcommand*\@@TenthstringFportuges[1]{%
2676   \ifcase#1\relax
2677     \or D\'ecima%
2678     \or Vig\'esima%
2679     \or Trig\'esima%
2680     \or Quadrag\'esima%
2681     \or Q\"uinquag\'esima%
2682     \or Sexag\'esima%
2683     \or Setuag\'esima%
2684     \or Octog\'esima%
2685     \or Nonag\'esima%
2686   \fi
2687 }%
2688 \global\let\@@TenthstringFportuges\@@TenthstringFportuges
```

Hundreds (feminine):

```
2689 \newcommand*\@@HundredthstringFportuges[1]{%
2690   \ifcase#1\relax
2691     \or Cent\'esima%
2692     \or Ducent\'esima%
```

```
2693    \or Trecent\'esima%
2694    \or Quadringent\'esima%
2695    \or Q\"uingent\'esima%
2696    \or Seiscent\'esima%
2697    \or Setingent\'esima%
2698    \or Octingent\'esima%
2699    \or Nongent\'esima%
2700  \fi
2701 }%
2702 \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.)

```
2703 \newcommand*\@@numberstringportuges[2]{%
2704 \ifnum#1>99999\relax
2705   \PackageError{fmtcount}{Out of range}%
2706   {This macro only works for values less than 100000}%
2707 \else
2708   \ifnum#1<0\relax
2709     \PackageError{fmtcount}{Negative numbers not permitted}%
2710     {This macro does not work for negative numbers, however
2711     you can try typing "minus" first, and then pass the modulus of
2712     this number}%
2713   \fi
2714 \fi
2715 \def#2{}%
2716 \@strctr=#1\relax \divide\@strctr by 1000\relax
2717 \ifnum\@strctr>9\relax
```

#1 is greater or equal to 10000

```
2718   \divide\@strctr by 10\relax
2719   \ifnum\@strctr>1\relax
2720     \let\@@fc@numstr#2\relax
2721     \protected@edef#2{\@@fc@numstr\@tenstring{\@strctr}}%
2722     \@strctr=#1 \divide\@strctr by 1000\relax
2723     \@FCmodulo{\@strctr}{10}%
2724     \ifnum\@strctr>0
2725       \ifnum\@strctr=1\relax
2726         \let\@@fc@numstr#2\relax
2727         \protected@edef#2{\@@fc@numstr\ \@andname}%
2728       \fi
2729       \let\@@fc@numstr#2\relax
2730       \protected@edef#2{\@@fc@numstr\ \@unitstring{\@strctr}}%
2731     \fi
2732   \else
2733     \@strctr=#1\relax
2734     \divide\@strctr by 1000\relax
2735     \@FCmodulo{\@strctr}{10}%
```

```
2736    \let\@@fc@numstr#2\relax
2737    \protected@edef#2{\@@fc@numstr\@teenstring{\@strctr}}%
2738  \fi
2739  \let\@@fc@numstr#2\relax
2740  \protected@edef#2{\@@fc@numstr\ \@thousand}%
2741 \else
2742  \ifnum\@strctr>0\relax
2743    \ifnum\@strctr>1\relax
2744      \let\@@fc@numstr#2\relax
2745      \protected@edef#2{\@@fc@numstr\@unitstring{\@strctr}\ }%
2746    \fi
2747    \let\@@fc@numstr#2\relax
2748    \protected@edef#2{\@@fc@numstr\@thousand}%
2749  \fi
2750 \fi
2751 \@strctr=#1\relax \@FCmodulo{\@strctr}{1000}%
2752 \divide\@strctr by 100\relax
2753 \ifnum\@strctr>0\relax
2754  \ifnum#1>1000 \relax
2755    \let\@@fc@numstr#2\relax
2756    \protected@edef#2{\@@fc@numstr\ }%
2757  \fi
2758  \@tmpstrctr=#1\relax
2759  \@FCmodulo{\@tmpstrctr}{1000}%
2760  \let\@@fc@numstr#2\relax
2761  \ifnum\@tmpstrctr=100\relax
2762    \protected@edef#2{\@@fc@numstr\@tenstring{10}}%
2763  \else
2764    \protected@edef#2{\@@fc@numstr\@hundredstring{\@strctr}}%
2765  \fi%
2766 \fi
2767 \@strctr=#1\relax \@FCmodulo{\@strctr}{100}%
2768 \ifnum#1>100\relax
2769  \ifnum\@strctr>0\relax
2770    \let\@@fc@numstr#2\relax
2771    \protected@edef#2{\@@fc@numstr\ \@andname\ }%
2772  \fi
2773 \fi
2774 \ifnum\@strctr>19\relax
2775  \divide\@strctr by 10\relax
2776  \let\@@fc@numstr#2\relax
2777  \protected@edef#2{\@@fc@numstr\@tenstring{\@strctr}}%
2778  \@strctr=#1\relax \@FCmodulo{\@strctr}{10}%
2779  \ifnum\@strctr>0
2780    \ifnum\@strctr=1\relax
2781      \let\@@fc@numstr#2\relax
2782      \protected@edef#2{\@@fc@numstr\ \@andname}%
2783    \else
2784      \ifnum#1>100\relax
```

```
2785        \let\@@fc@numstr#2\relax
2786        \protected@edef#2{\@@fc@numstr\ \@andname}%
2787      \fi
2788    \fi
2789    \let\@@fc@numstr#2\relax
2790    \protected@edef#2{\@@fc@numstr\ \@unitstring{\@strctr}}%
2791  \fi
2792 \else
2793  \ifnum\@strctr<10\relax
2794    \ifnum\@strctr=0\relax
2795      \ifnum#1<100\relax
2796        \let\@@fc@numstr#2\relax
2797        \protected@edef#2{\@@fc@numstr\@unitstring{\@strctr}}%
2798      \fi
2799    \else %(>0,<10)
2800      \let\@@fc@numstr#2\relax
2801      \protected@edef#2{\@@fc@numstr\@unitstring{\@strctr}}%
2802    \fi
2803  \else%>10
2804    \@FCmodulo{\@strctr}{10}%
2805    \let\@@fc@numstr#2\relax
2806    \protected@edef#2{\@@fc@numstr\@teenstring{\@strctr}}%
2807  \fi
2808 \fi
2809 }%
2810 \global\let\@@numberstringportuges\@@numberstringportuges
```

As above, but for ordinals.

```
2811 \newcommand*\@@ordinalstringportuges[2]{%
2812 \@strctr=#1\relax
2813 \ifnum#1>99999
2814 \PackageError{fmtcount}{Out of range}%
2815 {This macro only works for values less than 100000}%
2816 \else
2817 \ifnum#1<0
2818 \PackageError{fmtcount}{Negative numbers not permitted}%
2819 {This macro does not work for negative numbers, however
2820 you can try typing "minus" first, and then pass the modulus of
2821 this number}%
2822 \else
2823 \def#2{}%
2824 \ifnum\@strctr>999\relax
2825   \divide\@strctr by 1000\relax
2826   \ifnum\@strctr>1\relax
2827     \ifnum\@strctr>9\relax
2828       \@tmpstrctr=\@strctr
2829       \ifnum\@strctr<20
2830         \@FCmodulo{\@tmpstrctr}{10}%
2831         \let\@@fc@ordstr#2\relax
2832         \protected@edef#2{\@@fc@ordstr\@teenthstring{\@tmpstrctr}}%
```

```
2833      \else
2834        \divide\@tmpstrctr by 10\relax
2835        \let\@@fc@ordstr#2\relax
2836        \protected@edef#2{\@@fc@ordstr\@tenthstring{\@tmpstrctr}}%
2837        \@tmpstrctr=\@strctr
2838        \@FCmodulo{\@tmpstrctr}{10}%
2839        \ifnum\@tmpstrctr>0\relax
2840          \let\@@fc@ordstr#2\relax
2841          \protected@edef#2{\@@fc@ordstr\@unitthstring{\@tmpstrctr}}%
2842        \fi
2843      \fi
2844    \else
2845      \let\@@fc@ordstr#2\relax
2846      \protected@edef#2{\@@fc@ordstr\@unitstring{\@strctr}}%
2847    \fi
2848  \fi
2849  \let\@@fc@ordstr#2\relax
2850  \protected@edef#2{\@@fc@ordstr\@thousandth}%
2851 \fi
2852 \@strctr=#1\relax
2853 \@FCmodulo{\@strctr}{1000}%
2854 \ifnum\@strctr>99\relax
2855   \@tmpstrctr=\@strctr
2856   \divide\@tmpstrctr by 100\relax
2857   \ifnum#1>1000\relax
2858     \let\@@fc@ordstr#2\relax
2859     \protected@edef#2{\@@fc@ordstr-}%
2860   \fi
2861   \let\@@fc@ordstr#2\relax
2862   \protected@edef#2{\@@fc@ordstr\@hundredthstring{\@tmpstrctr}}%
2863 \fi
2864 \@FCmodulo{\@strctr}{100}%
2865 \ifnum#1>99\relax
2866   \ifnum\@strctr>0\relax
2867     \let\@@fc@ordstr#2\relax
2868     \protected@edef#2{\@@fc@ordstr-}%
2869   \fi
2870 \fi
2871 \ifnum\@strctr>9\relax
2872   \@tmpstrctr=\@strctr
2873   \divide\@tmpstrctr by 10\relax
2874   \let\@@fc@ordstr#2\relax
2875   \protected@edef#2{\@@fc@ordstr\@tenthstring{\@tmpstrctr}}%
2876   \@tmpstrctr=\@strctr
2877   \@FCmodulo{\@tmpstrctr}{10}%
2878   \ifnum\@tmpstrctr>0\relax
2879     \let\@@fc@ordstr#2\relax
2880     \protected@edef#2{\@@fc@ordstr-\@unitthstring{\@tmpstrctr}}%
2881   \fi
```

```
2882 \else
2883   \ifnum\@strctr=0\relax
2884     \ifnum#1=0\relax
2885       \let\@@fc@ordstr#2\relax
2886       \protected@edef#2{\@@fc@ordstr\@unitstring{0}}%
2887     \fi
2888   \else
2889     \let\@@fc@ordstr#2\relax
2890     \protected@edef#2{\@@fc@ordstr\@unitthstring{\@strctr}}%
2891   \fi
2892 \fi
2893 \fi
2894 \fi
2895 }%
2896 \global\let\@@ordinalstringportuges\@@ordinalstringportuges
```

9.1.13 fc-portuguese.def

```
2897 \ProvidesFCLanguage{portuguese}[2014/06/09]%
```

Load fc-portuges.def if not already loaded
```
2898 \FCloadlang{portuges}%
```

Set portuguese to be equivalent to portuges.
```
2899 \global\let\@ordinalMportuguese=\@ordinalMportuges
2900 \global\let\@ordinalFportuguese=\@ordinalFportuges
2901 \global\let\@ordinalNportuguese=\@ordinalNportuges
2902 \global\let\@numberstringMportuguese=\@numberstringMportuges
2903 \global\let\@numberstringFportuguese=\@numberstringFportuges
2904 \global\let\@numberstringNportuguese=\@numberstringNportuges
2905 \global\let\@NumberstringMportuguese=\@NumberstringMportuges
2906 \global\let\@NumberstringFportuguese=\@NumberstringFportuges
2907 \global\let\@NumberstringNportuguese=\@NumberstringNportuges
2908 \global\let\@ordinalstringMportuguese=\@ordinalstringMportuges
2909 \global\let\@ordinalstringFportuguese=\@ordinalstringFportuges
2910 \global\let\@ordinalstringNportuguese=\@ordinalstringNportuges
2911 \global\let\@OrdinalstringMportuguese=\@OrdinalstringMportuges
2912 \global\let\@OrdinalstringFportuguese=\@OrdinalstringFportuges
2913 \global\let\@OrdinalstringNportuguese=\@OrdinalstringNportuges
```
9.1.14 fc-spanish.def

Spanish definitions
```
2914 \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:
```
2915 \newcommand*\@ordinalMspanish[2]{%
2916   \edef#2{\number#1\relax\noexpand\fmtord{o}}%
2917 }%
2918 \global\let\@ordinalMspanish\@ordinalMspanish
```

Feminine:

```
2919 \newcommand{\@ordinalFspanish}[2]{%
2920   \edef#2{\number#1\relax\noexpand\fmtord{a}}%
2921 }%
2922 \global\let\@ordinalFspanish\@ordinalFspanish
```

Make neuter same as masculine:

```
2923 \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):

```
2924 \newcommand*\@@unitstringspanish[1]{%
2925   \ifcase#1\relax
2926     cero%
2927     \or uno%
2928     \or dos%
2929     \or tres%
2930     \or cuatro%
2931     \or cinco%
2932     \or seis%
2933     \or siete%
2934     \or ocho%
2935     \or nueve%
2936   \fi
2937 }%
2938 \global\let\@@unitstringspanish\@@unitstringspanish
```

Feminine:

```
2939 \newcommand*\@@unitstringFspanish[1]{%
2940   \ifcase#1\relax
2941     cera%
2942     \or una%
2943     \or dos%
2944     \or tres%
2945     \or cuatro%
2946     \or cinco%
2947     \or seis%
2948     \or siete%
2949     \or ocho%
2950     \or nueve%
2951   \fi
2952 }%
2953 \global\let\@@unitstringFspanish\@@unitstringFspanish
```

Tens (argument must go from 1 to 10):

```
2954 \newcommand*\@@tenstringspanish[1]{%
2955   \ifcase#1\relax
2956     \or diez%
2957     \or veinte%
2958     \or treinta%
2959     \or cuarenta%
```

```
2960      \or cincuenta%
2961      \or sesenta%
2962      \or setenta%
2963      \or ochenta%
2964      \or noventa%
2965      \or cien%
2966   \fi
2967 }%
2968 \global\let\@@tenstringspanish\@@tenstringspanish
```

Teens:

```
2969 \newcommand*\@@teenstringspanish[1]{%
2970   \ifcase#1\relax
2971      diez%
2972      \or once%
2973      \or doce%
2974      \or trece%
2975      \or catorce%
2976      \or quince%
2977      \or diecis\'eis%
2978      \or diecisiete%
2979      \or dieciocho%
2980      \or diecinueve%
2981   \fi
2982 }%
2983 \global\let\@@teenstringspanish\@@teenstringspanish
```

Twenties:

```
2984 \newcommand*\@@twentystringspanish[1]{%
2985   \ifcase#1\relax
2986      veinte%
2987      \or veintiuno%
2988      \or veintid\'os%
2989      \or veintitr\'es%
2990      \or veinticuatro%
2991      \or veinticinco%
2992      \or veintis\'eis%
2993      \or veintisiete%
2994      \or veintiocho%
2995      \or veintinueve%
2996   \fi
2997 }%
2998 \global\let\@@twentystringspanish\@@twentystringspanish
```

Feminine form:

```
2999 \newcommand*\@@twentystringFspanish[1]{%
3000   \ifcase#1\relax
3001      veinte%
3002      \or veintiuna%
3003      \or veintid\'os%
3004      \or veintitr\'es%
```

```
3005      \or veinticuatro%
3006      \or veinticinco%
3007      \or veintis\'eis%
3008      \or veintisiete%
3009      \or veintiocho%
3010      \or veintinueve%
3011   \fi
3012 }%
3013 \global\let\@@twentystringFspanish\@@twentystringFspanish
```

Hundreds:

```
3014 \newcommand*\@@hundredstringspanish[1]{%
3015   \ifcase#1\relax
3016      \or ciento%
3017      \or doscientos%
3018      \or trescientos%
3019      \or cuatrocientos%
3020      \or quinientos%
3021      \or seiscientos%
3022      \or setecientos%
3023      \or ochocientos%
3024      \or novecientos%
3025   \fi
3026 }%
3027 \global\let\@@hundredstringspanish\@@hundredstringspanish
```

Feminine form:

```
3028 \newcommand*\@@hundredstringFspanish[1]{%
3029   \ifcase#1\relax
3030      \or cienta%
3031      \or doscientas%
3032      \or trescientas%
3033      \or cuatrocientas%
3034      \or quinientas%
3035      \or seiscientas%
3036      \or setecientas%
3037      \or ochocientas%
3038      \or novecientas%
3039   \fi
3040 }%
3041 \global\let\@@hundredstringFspanish\@@hundredstringFspanish
```

As above, but with initial letter uppercase:

```
3042 \newcommand*\@@Unitstringspanish[1]{%
3043   \ifcase#1\relax
3044      Cero%
3045      \or Uno%
3046      \or Dos%
3047      \or Tres%
3048      \or Cuatro%
3049      \or Cinco%
```

89

```
3050     \or Seis%
3051     \or Siete%
3052     \or Ocho%
3053     \or Nueve%
3054   \fi
3055 }%
3056 \global\let\@@Unitstringspanish\@@Unitstringspanish
```

Feminine form:

```
3057 \newcommand*\@@UnitstringFspanish[1]{%
3058   \ifcase#1\relax
3059     Cera%
3060     \or Una%
3061     \or Dos%
3062     \or Tres%
3063     \or Cuatro%
3064     \or Cinco%
3065     \or Seis%
3066     \or Siete%
3067     \or Ocho%
3068     \or Nueve%
3069   \fi
3070 }%
3071 \global\let\@@UnitstringFspanish\@@UnitstringFspanish
```

Tens:

```
3072 %\changes{2.0}{2012-06-18}{fixed spelling mistake (correction
3073 %provided by Fernando Maldonado)}
3074 \newcommand*\@@Tenstringspanish[1]{%
3075   \ifcase#1\relax
3076     \or Diez%
3077     \or Veinte%
3078     \or Treinta%
3079     \or Cuarenta%
3080     \or Cincuenta%
3081     \or Sesenta%
3082     \or Setenta%
3083     \or Ochenta%
3084     \or Noventa%
3085     \or Cien%
3086   \fi
3087 }%
3088 \global\let\@@Tenstringspanish\@@Tenstringspanish
```

Teens:

```
3089 \newcommand*\@@Teenstringspanish[1]{%
3090   \ifcase#1\relax
3091     Diez%
3092     \or Once%
3093     \or Doce%
3094     \or Trece%
```

```
3095     \or Catorce%
3096     \or Quince%
3097     \or Diecis\'eis%
3098     \or Diecisiete%
3099     \or Dieciocho%
3100     \or Diecinueve%
3101   \fi
3102 }%
3103 \global\let\@@Teenstringspanish\@@Teenstringspanish
```

Twenties:

```
3104 \newcommand*\@@Twentystringspanish[1]{%
3105   \ifcase#1\relax
3106     Veinte%
3107     \or Veintiuno%
3108     \or Veintid\'os%
3109     \or Veintitr\'es%
3110     \or Veinticuatro%
3111     \or Veinticinco%
3112     \or Veintis\'eis%
3113     \or Veintisiete%
3114     \or Veintiocho%
3115     \or Veintinueve%
3116   \fi
3117 }%
3118 \global\let\@@Twentystringspanish\@@Twentystringspanish
```

Feminine form:

```
3119 \newcommand*\@@TwentystringFspanish[1]{%
3120   \ifcase#1\relax
3121     Veinte%
3122     \or Veintiuna%
3123     \or Veintid\'os%
3124     \or Veintitr\'es%
3125     \or Veinticuatro%
3126     \or Veinticinco%
3127     \or Veintis\'eis%
3128     \or Veintisiete%
3129     \or Veintiocho%
3130     \or Veintinueve%
3131   \fi
3132 }%
3133 \global\let\@@TwentystringFspanish\@@TwentystringFspanish
```

Hundreds:

```
3134 \newcommand*\@@Hundredstringspanish[1]{%
3135   \ifcase#1\relax
3136     \or Ciento%
3137     \or Doscientos%
3138     \or Trescientos%
3139     \or Cuatrocientos%
```

```
3140      \or Quinientos%
3141      \or Seiscientos%
3142      \or Setecientos%
3143      \or Ochocientos%
3144      \or Novecientos%
3145   \fi
3146 }%
3147 \global\let\@@Hundredstringspanish\@@Hundredstringspanish
```

Feminine form:

```
3148 \newcommand*\@@HundredstringFspanish[1]{%
3149   \ifcase#1\relax
3150      \or Cienta%
3151      \or Doscientas%
3152      \or Trescientas%
3153      \or Cuatrocientas%
3154      \or Quinientas%
3155      \or Seiscientas%
3156      \or Setecientas%
3157      \or Ochocientas%
3158      \or Novecientas%
3159   \fi
3160 }%
3161 \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.)

```
3162 \newcommand*{\@numberstringMspanish}[2]{%
3163   \let\@unitstring=\@@unitstringspanish
3164   \let\@teenstring=\@@teenstringspanish
3165   \let\@tenstring=\@@tenstringspanish
3166   \let\@twentystring=\@@twentystringspanish
3167   \let\@hundredstring=\@@hundredstringspanish
3168   \def\@hundred{cien}\def\@thousand{mil}%
3169   \def\@andname{y}%
3170   \@@numberstringspanish{#1}{#2}%
3171 }%
3172 \global\let\@numberstringMspanish\@numberstringMspanish
```

Feminine form:

```
3173 \newcommand*{\@numberstringFspanish}[2]{%
3174   \let\@unitstring=\@@unitstringFspanish
3175   \let\@teenstring=\@@teenstringspanish
3176   \let\@tenstring=\@@tenstringspanish
3177   \let\@twentystring=\@@twentystringFspanish
3178   \let\@hundredstring=\@@hundredstringFspanish
3179   \def\@hundred{cien}\def\@thousand{mil}%
3180   \def\@andname{b}%
3181   \@@numberstringspanish{#1}{#2}%
```

```
3182 }%
3183 \global\let\@numberstringFspanish\@numberstringFspanish
```

Make neuter same as masculine:

```
3184 \global\let\@numberstringNspanish\@numberstringMspanish
```

As above, but initial letters in upper case:

```
3185 \newcommand*{\@NumberstringMspanish}[2]{%
3186   \let\@unitstring=\@@Unitstringspanish
3187   \let\@teenstring=\@@Teenstringspanish
3188   \let\@tenstring=\@@Tenstringspanish
3189   \let\@twentystring=\@@Twentystringspanish
3190   \let\@hundredstring=\@@Hundredstringspanish
3191   \def\@andname{y}%
3192   \def\@hundred{Cien}\def\@thousand{Mil}%
3193   \@@numberstringspanish{#1}{#2}%
3194 }%
3195 \global\let\@NumberstringMspanish\@NumberstringMspanish
```

Feminine form:

```
3196 \newcommand*{\@NumberstringFspanish}[2]{%
3197   \let\@unitstring=\@@UnitstringFspanish
3198   \let\@teenstring=\@@Teenstringspanish
3199   \let\@tenstring=\@@Tenstringspanish
3200   \let\@twentystring=\@@TwentystringFspanish
3201   \let\@hundredstring=\@@HundredstringFspanish
3202   \def\@andname{b}%
3203   \def\@hundred{Cien}\def\@thousand{Mil}%
3204   \@@numberstringspanish{#1}{#2}%
3205 }%
3206 \global\let\@NumberstringFspanish\@NumberstringFspanish
```

Make neuter same as masculine:

```
3207 \global\let\@NumberstringNspanish\@NumberstringMspanish
```

As above, but for ordinals.

```
3208 \newcommand*{\@ordinalstringMspanish}[2]{%
3209   \let\@unitthstring=\@@unitthstringspanish
3210   \let\@unitstring=\@@unitstringspanish
3211   \let\@teenthstring=\@@teenthstringspanish
3212   \let\@tenthstring=\@@tenthstringspanish
3213   \let\@hundredthstring=\@@hundredthstringspanish
3214   \def\@thousandth{mil\'esimo}%
3215   \@@ordinalstringspanish{#1}{#2}%
3216 }%
3217 \global\let\@ordinalstringMspanish\@ordinalstringMspanish
```

Feminine form:

```
3218 \newcommand*{\@ordinalstringFspanish}[2]{%
3219   \let\@unitthstring=\@@unitthstringFspanish
3220   \let\@unitstring=\@@unitstringFspanish
3221   \let\@teenthstring=\@@teenthstringFspanish
```

```
3222    \let\@tenthstring=\@@tenthstringFspanish
3223    \let\@hundredthstring=\@@hundredthstringFspanish
3224    \def\@thousandth{mil\'esima}%
3225    \@@ordinalstringspanish{#1}{#2}%
3226 }%
3227 \global\let\@ordinalstringFspanish\@ordinalstringFspanish
```

Make neuter same as masculine:

```
3228 \global\let\@ordinalstringNspanish\@ordinalstringMspanish
```

As above, but with initial letters in upper case.

```
3229 \newcommand*{\@OrdinalstringMspanish}[2]{%
3230    \let\@unitthstring=\@@Unitthstringspanish
3231    \let\@unitstring=\@@Unitstringspanish
3232    \let\@teenthstring=\@@Teenthstringspanish
3233    \let\@tenthstring=\@@Tenthstringspanish
3234    \let\@hundredthstring=\@@Hundredthstringspanish
3235    \def\@thousandth{Mil\'esimo}%
3236    \@@ordinalstringspanish{#1}{#2}%
3237 }
3238 \global\let\@OrdinalstringMspanish\@OrdinalstringMspanish
```

Feminine form:

```
3239 \newcommand*{\@OrdinalstringFspanish}[2]{%
3240    \let\@unitthstring=\@@UnitthstringFspanish
3241    \let\@unitstring=\@@UnitstringFspanish
3242    \let\@teenthstring=\@@TeenthstringFspanish
3243    \let\@tenthstring=\@@TenthstringFspanish
3244    \let\@hundredthstring=\@@HundredthstringFspanish
3245    \def\@thousandth{Mil\'esima}%
3246    \@@ordinalstringspanish{#1}{#2}%
3247 }%
3248 \global\let\@OrdinalstringFspanish\@OrdinalstringFspanish
```

Make neuter same as masculine:

```
3249 \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:

```
3250 \newcommand*\@@unitthstringspanish[1]{%
3251    \ifcase#1\relax
3252       cero%
3253       \or primero%
3254       \or segundo%
3255       \or tercero%
3256       \or cuarto%
3257       \or quinto%
3258       \or sexto%
3259       \or s\'eptimo%
3260       \or octavo%
3261       \or noveno%
```

```
3262    \fi
3263 }%
3264 \global\let\@@unitthstringspanish\@@unitthstringspanish
```
   Tens:
```
3265 \newcommand*\@@tenthstringspanish[1]{%
3266    \ifcase#1\relax
3267      \or d\'ecimo%
3268      \or vig\'esimo%
3269      \or trig\'esimo%
3270      \or cuadrag\'esimo%
3271      \or quincuag\'esimo%
3272      \or sexag\'esimo%
3273      \or septuag\'esimo%
3274      \or octog\'esimo%
3275      \or nonag\'esimo%
3276    \fi
3277 }%
3278 \global\let\@@tenthstringspanish\@@tenthstringspanish
```
   Teens:
```
3279 \newcommand*\@@teenthstringspanish[1]{%
3280    \ifcase#1\relax
3281      d\'ecimo%
3282      \or und\'ecimo%
3283      \or duod\'ecimo%
3284      \or decimotercero%
3285      \or decimocuarto%
3286      \or decimoquinto%
3287      \or decimosexto%
3288      \or decimos\'eptimo%
3289      \or decimoctavo%
3290      \or decimonoveno%
3291    \fi
3292 }%
3293 \global\let\@@teenthstringspanish\@@teenthstringspanish
```
   Hundreds:
```
3294 \newcommand*\@@hundredthstringspanish[1]{%
3295    \ifcase#1\relax
3296      \or cent\'esimo%
3297      \or ducent\'esimo%
3298      \or tricent\'esimo%
3299      \or cuadringent\'esimo%
3300      \or quingent\'esimo%
3301      \or sexcent\'esimo%
3302      \or septing\'esimo%
3303      \or octingent\'esimo%
3304      \or noningent\'esimo%
3305    \fi
3306 }%
```

```
3307 \global\let\@@hundredthstringspanish\@@hundredthstringspanish
```

Units (feminine):

```
3308 \newcommand*\@@unitthstringFspanish[1]{%
3309   \ifcase#1\relax
3310     cera%
3311     \or primera%
3312     \or segunda%
3313     \or tercera%
3314     \or cuarta%
3315     \or quinta%
3316     \or sexta%
3317     \or s\'eptima%
3318     \or octava%
3319     \or novena%
3320   \fi
3321 }%
3322 \global\let\@@unitthstringFspanish\@@unitthstringFspanish
```

Tens (feminine):

```
3323 \newcommand*\@@tenthstringFspanish[1]{%
3324   \ifcase#1\relax
3325     \or d\'ecima%
3326     \or vig\'esima%
3327     \or trig\'esima%
3328     \or cuadrag\'esima%
3329     \or quincuag\'esima%
3330     \or sexag\'esima%
3331     \or septuag\'esima%
3332     \or octog\'esima%
3333     \or nonag\'esima%
3334   \fi
3335 }%
3336 \global\let\@@tenthstringFspanish\@@tenthstringFspanish
```

Teens (feminine)

```
3337 \newcommand*\@@teenthstringFspanish[1]{%
3338   \ifcase#1\relax
3339     d\'ecima%
3340     \or und\'ecima%
3341     \or duod\'ecima%
3342     \or decimotercera%
3343     \or decimocuarta%
3344     \or decimoquinta%
3345     \or decimosexta%
3346     \or decimos\'eptima%
3347     \or decimoctava%
3348     \or decimonovena%
3349   \fi
3350 }%
3351 \global\let\@@teenthstringFspanish\@@teenthstringFspanish
```

Hundreds (feminine)

```
3352 \newcommand*\@@hundredthstringFspanish[1]{%
3353   \ifcase#1\relax
3354     \or cent\'esima%
3355     \or ducent\'esima%
3356     \or tricent\'esima%
3357     \or cuadringent\'esima%
3358     \or quingent\'esima%
3359     \or sexcent\'esima%
3360     \or septing\'esima%
3361     \or octingent\'esima%
3362     \or noningent\'esima%
3363   \fi
3364 }%
3365 \global\let\@@hundredthstringFspanish\@@hundredthstringFspanish
```

As above, but with initial letters in upper case

```
3366 \newcommand*\@@Unitthstringspanish[1]{%
3367   \ifcase#1\relax
3368     Cero%
3369     \or Primero%
3370     \or Segundo%
3371     \or Tercero%
3372     \or Cuarto%
3373     \or Quinto%
3374     \or Sexto%
3375     \or S\'eptimo%
3376     \or Octavo%
3377     \or Noveno%
3378   \fi
3379 }%
3380 \global\let\@@Unitthstringspanish\@@Unitthstringspanish
```

Tens:

```
3381 \newcommand*\@@Tenthstringspanish[1]{%
3382   \ifcase#1\relax
3383     \or D\'ecimo%
3384     \or Vig\'esimo%
3385     \or Trig\'esimo%
3386     \or Cuadrag\'esimo%
3387     \or Quincuag\'esimo%
3388     \or Sexag\'esimo%
3389     \or Septuag\'esimo%
3390     \or Octog\'esimo%
3391     \or Nonag\'esimo%
3392   \fi
3393 }%
3394 \global\let\@@Tenthstringspanish\@@Tenthstringspanish
```

Teens:

```
3395 \newcommand*\@@Teenthstringspanish[1]{%
3396   \ifcase#1\relax
3397     D\'ecimo%
3398     \or Und\'ecimo%
3399     \or Duod\'ecimo%
3400     \or Decimotercero%
3401     \or Decimocuarto%
3402     \or Decimoquinto%
3403     \or Decimosexto%
3404     \or Decimos\'eptimo%
3405     \or Decimoctavo%
3406     \or Decimonoveno%
3407   \fi
3408 }%
3409 \global\let\@@Teenthstringspanish\@@Teenthstringspanish
```

Hundreds

```
3410 \newcommand*\@@Hundredthstringspanish[1]{%
3411   \ifcase#1\relax
3412     \or Cent\'esimo%
3413     \or Ducent\'esimo%
3414     \or Tricent\'esimo%
3415     \or Cuadringent\'esimo%
3416     \or Quingent\'esimo%
3417     \or Sexcent\'esimo%
3418     \or Septing\'esimo%
3419     \or Octingent\'esimo%
3420     \or Noningent\'esimo%
3421   \fi
3422 }%
3423 \global\let\@@Hundredthstringspanish\@@Hundredthstringspanish
```

As above, but feminine.

```
3424 \newcommand*\@@UnitthstringFspanish[1]{%
3425   \ifcase#1\relax
3426     Cera%
3427     \or Primera%
3428     \or Segunda%
3429     \or Tercera%
3430     \or Cuarta%
3431     \or Quinta%
3432     \or Sexta%
3433     \or S\'eptima%
3434     \or Octava%
3435     \or Novena%
3436   \fi
3437 }%
3438 \global\let\@@UnitthstringFspanish\@@UnitthstringFspanish
```

Tens (feminine)

```
3439 \newcommand*\@@TenthstringFspanish[1]{%
```

```
3440    \ifcase#1\relax
3441      \or D\'ecima%
3442      \or Vig\'esima%
3443      \or Trig\'esima%
3444      \or Cuadrag\'esima%
3445      \or Quincuag\'esima%
3446      \or Sexag\'esima%
3447      \or Septuag\'esima%
3448      \or Octog\'esima%
3449      \or Nonag\'esima%
3450    \fi
3451 }%
3452 \global\let\@@TenthstringFspanish\@@TenthstringFspanish
```

Teens (feminine):

```
3453 \newcommand*\@@TeenthstringFspanish[1]{%
3454    \ifcase#1\relax
3455      D\'ecima%
3456      \or Und\'ecima%
3457      \or Duod\'ecima%
3458      \or Decimotercera%
3459      \or Decimocuarta%
3460      \or Decimoquinta%
3461      \or Decimosexta%
3462      \or Decimos\'eptima%
3463      \or Decimoctava%
3464      \or Decimonovena%
3465    \fi
3466 }%
3467 \global\let\@@TeenthstringFspanish\@@TeenthstringFspanish
```

Hundreds (feminine):

```
3468 \newcommand*\@@HundredthstringFspanish[1]{%
3469    \ifcase#1\relax
3470      \or Cent\'esima%
3471      \or Ducent\'esima%
3472      \or Tricent\'esima%
3473      \or Cuadringent\'esima%
3474      \or Quingent\'esima%
3475      \or Sexcent\'esima%
3476      \or Septing\'esima%
3477      \or Octingent\'esima%
3478      \or Noningent\'esima%
3479    \fi
3480 }%
3481 \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 documnets created with older versions. (These internal

99

macros are not meant for use in documents.)

```
3482 \newcommand*\@@numberstringspanish[2]{%
3483 \ifnum#1>99999
3484 \PackageError{fmtcount}{Out of range}%
3485 {This macro only works for values less than 100000}%
3486 \else
3487 \ifnum#1<0
3488 \PackageError{fmtcount}{Negative numbers not permitted}%
3489 {This macro does not work for negative numbers, however
3490 you can try typing "minus" first, and then pass the modulus of
3491 this number}%
3492 \fi
3493 \fi
3494 \def#2{}%
3495 \@strctr=#1\relax \divide\@strctr by 1000\relax
3496 \ifnum\@strctr>9
```

#1 is greater or equal to 10000

```
3497   \divide\@strctr by 10
3498   \ifnum\@strctr>1
3499     \let\@@fc@numstr#2\relax
3500     \edef#2{\@@fc@numstr\@tenstring{\@strctr}}%
3501     \@strctr=#1 \divide\@strctr by 1000\relax
3502     \@FCmodulo{\@strctr}{10}%
3503     \ifnum\@strctr>0\relax
3504        \let\@@fc@numstr#2\relax
3505        \edef#2{\@@fc@numstr\ \@andname\ \@unitstring{\@strctr}}%
3506     \fi
3507   \else
3508     \@strctr=#1\relax
3509     \divide\@strctr by 1000\relax
3510     \@FCmodulo{\@strctr}{10}%
3511     \let\@@fc@numstr#2\relax
3512     \edef#2{\@@fc@numstr\@teenstring{\@strctr}}%
3513   \fi
3514   \let\@@fc@numstr#2\relax
3515   \edef#2{\@@fc@numstr\ \@thousand}%
3516 \else
3517   \ifnum\@strctr>0\relax
3518     \ifnum\@strctr>1\relax
3519        \let\@@fc@numstr#2\relax
3520        \edef#2{\@@fc@numstr\@unitstring{\@strctr}\ }%
3521     \fi
3522     \let\@@fc@numstr#2\relax
3523     \edef#2{\@@fc@numstr\@thousand}%
3524   \fi
3525 \fi
3526 \@strctr=#1\relax \@FCmodulo{\@strctr}{1000}%
3527 \divide\@strctr by 100\relax
3528 \ifnum\@strctr>0\relax
```

```
3529  \ifnum#1>1000\relax
3530    \let\@@fc@numstr#2\relax
3531    \edef#2{\@@fc@numstr\ }%
3532  \fi
3533  \@tmpstrctr=#1\relax
3534  \@FCmodulo{\@tmpstrctr}{1000}%
3535  \ifnum\@tmpstrctr=100\relax
3536    \let\@@fc@numstr#2\relax
3537    \edef#2{\@@fc@numstr\@tenstring{10}}%
3538  \else
3539    \let\@@fc@numstr#2\relax
3540    \edef#2{\@@fc@numstr\@hundredstring{\@strctr}}%
3541  \fi
3542 \fi
3543 \@strctr=#1\relax \@FCmodulo{\@strctr}{100}%
3544 \ifnum#1>100\relax
3545   \ifnum\@strctr>0\relax
3546     \let\@@fc@numstr#2\relax

3547     \edef#2{\@@fc@numstr\ }%
3548   \fi
3549 \fi
3550 \ifnum\@strctr>29\relax
3551   \divide\@strctr by 10\relax
3552   \let\@@fc@numstr#2\relax
3553   \edef#2{\@@fc@numstr\@tenstring{\@strctr}}%
3554   \@strctr=#1\relax \@FCmodulo{\@strctr}{10}%
3555   \ifnum\@strctr>0\relax
3556     \let\@@fc@numstr#2\relax
3557     \edef#2{\@@fc@numstr\ \@andname\ \@unitstring{\@strctr}}%
3558   \fi
3559 \else
3560   \ifnum\@strctr<10\relax
3561     \ifnum\@strctr=0\relax
3562       \ifnum#1<100\relax
3563         \let\@@fc@numstr#2\relax
3564         \edef#2{\@@fc@numstr\@unitstring{\@strctr}}%
3565       \fi
3566     \else
3567       \let\@@fc@numstr#2\relax
3568       \edef#2{\@@fc@numstr\@unitstring{\@strctr}}%
3569     \fi
3570   \else
3571     \ifnum\@strctr>19\relax
3572       \@FCmodulo{\@strctr}{10}%
3573       \let\@@fc@numstr#2\relax
3574       \edef#2{\@@fc@numstr\@twentystring{\@strctr}}%
3575     \else
3576       \@FCmodulo{\@strctr}{10}%
3577       \let\@@fc@numstr#2\relax
```

```
3578        \edef#2{\@@fc@numstr\@teenstring{\@strctr}}%
3579     \fi
3580   \fi
3581 \fi
3582 }%
3583 \global\let\@@numberstringspanish\@@numberstringspanish
```

As above, but for ordinals

```
3584 \newcommand*\@@ordinalstringspanish[2]{%
3585 \@strctr=#1\relax
3586 \ifnum#1>99999
3587 \PackageError{fmtcount}{Out of range}%
3588 {This macro only works for values less than 100000}%
3589 \else
3590 \ifnum#1<0
3591 \PackageError{fmtcount}{Negative numbers not permitted}%
3592 {This macro does not work for negative numbers, however
3593 you can try typing "minus" first, and then pass the modulus of
3594 this number}%
3595 \else
3596 \def#2{}%
3597 \ifnum\@strctr>999\relax
3598   \divide\@strctr by 1000\relax
3599   \ifnum\@strctr>1\relax
3600     \ifnum\@strctr>9\relax
3601       \@tmpstrctr=\@strctr
3602       \ifnum\@strctr<20
3603         \@FCmodulo{\@tmpstrctr}{10}%
3604         \let\@@fc@ordstr#2\relax
3605         \edef#2{\@@fc@ordstr\@teenthstring{\@tmpstrctr}}%
3606       \else
3607         \divide\@tmpstrctr by 10\relax
3608         \let\@@fc@ordstr#2\relax
3609         \edef#2{\@@fc@ordstr\@tenthstring{\@tmpstrctr}}%
3610         \@tmpstrctr=\@strctr
3611         \@FCmodulo{\@tmpstrctr}{10}%
3612         \ifnum\@tmpstrctr>0\relax
3613           \let\@@fc@ordstr#2\relax
3614           \edef#2{\@@fc@ordstr\@unitthstring{\@tmpstrctr}}%
3615         \fi
3616       \fi
3617     \else
3618       \let\@@fc@ordstr#2\relax
3619       \edef#2{\@@fc@ordstr\@unitstring{\@strctr}}%
3620     \fi
3621   \fi
3622   \let\@@fc@ordstr#2\relax
3623   \edef#2{\@@fc@ordstr\@thousandth}%
3624 \fi
3625 \@strctr=#1\relax
```

```
3626 \@FCmodulo{\@strctr}{1000}%
3627 \ifnum\@strctr>99\relax
3628   \@tmpstrctr=\@strctr
3629   \divide\@tmpstrctr by 100\relax
3630   \ifnum#1>1000\relax
3631     \let\@@fc@ordstr#2\relax
3632     \edef#2{\@@fc@ordstr\ }%
3633   \fi
3634   \let\@@fc@ordstr#2\relax
3635   \edef#2{\@@fc@ordstr\@hundredthstring{\@tmpstrctr}}%
3636 \fi
3637 \@FCmodulo{\@strctr}{100}%
3638 \ifnum#1>99\relax
3639   \ifnum\@strctr>0\relax
3640     \let\@@fc@ordstr#2\relax
3641     \edef#2{\@@fc@ordstr\ }%
3642   \fi
3643 \fi
3644 \ifnum\@strctr>19\relax
3645   \@tmpstrctr=\@strctr
3646   \divide\@tmpstrctr by 10\relax
3647   \let\@@fc@ordstr#2\relax
3648   \edef#2{\@@fc@ordstr\@tenthstring{\@tmpstrctr}}%
3649   \@tmpstrctr=\@strctr
3650   \@FCmodulo{\@tmpstrctr}{10}%
3651   \ifnum\@tmpstrctr>0\relax
3652     \let\@@fc@ordstr#2\relax
3653     \edef#2{\@@fc@ordstr\ \@unitthstring{\@tmpstrctr}}%
3654   \fi
3655 \else
3656   \ifnum\@strctr>9\relax
3657     \@FCmodulo{\@strctr}{10}%
3658     \let\@@fc@ordstr#2\relax
3659     \edef#2{\@@fc@ordstr\@teenthstring{\@strctr}}%
3660   \else
3661     \ifnum\@strctr=0\relax
3662       \ifnum#1=0\relax
3663         \let\@@fc@ordstr#2\relax
3664         \edef#2{\@@fc@ordstr\@unitstring{0}}%
3665       \fi
3666     \else
3667       \let\@@fc@ordstr#2\relax
3668       \edef#2{\@@fc@ordstr\@unitthstring{\@strctr}}%
3669     \fi
3670   \fi
3671 \fi
3672 \fi
3673 \fi
3674 }%
```

```
3675 \global\let\@@ordinalstringspanish\@@ordinalstringspanish
```

9.1.15 fc-UKenglish.def

English definitions

```
3676 \ProvidesFCLanguage{UKenglish}[2013/08/17]%
```

Loaded fc-english.def if not already loaded

```
3677 \FCloadlang{english}%
```

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

```
3678 \global\let\@ordinalMUKenglish\@ordinalMenglish
3679 \global\let\@ordinalFUKenglish\@ordinalMenglish
3680 \global\let\@ordinalNUKenglish\@ordinalMenglish
3681 \global\let\@numberstringMUKenglish\@numberstringMenglish
3682 \global\let\@numberstringFUKenglish\@numberstringMenglish
3683 \global\let\@numberstringNUKenglish\@numberstringMenglish
3684 \global\let\@NumberstringMUKenglish\@NumberstringMenglish
3685 \global\let\@NumberstringFUKenglish\@NumberstringMenglish
3686 \global\let\@NumberstringNUKenglish\@NumberstringMenglish
3687 \global\let\@ordinalstringMUKenglish\@ordinalstringMenglish
3688 \global\let\@ordinalstringFUKenglish\@ordinalstringMenglish
3689 \global\let\@ordinalstringNUKenglish\@ordinalstringMenglish
3690 \global\let\@OrdinalstringMUKenglish\@OrdinalstringMenglish
3691 \global\let\@OrdinalstringFUKenglish\@OrdinalstringMenglish
3692 \global\let\@OrdinalstringNUKenglish\@OrdinalstringMenglish
```

9.1.16 fc-USenglish.def

US English definitions

```
3693 \ProvidesFCLanguage{USenglish}[2013/08/17]%
```

Loaded fc-english.def if not already loaded

```
3694 \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.)

```
3695 \global\let\@ordinalMUSenglish\@ordinalMenglish
3696 \global\let\@ordinalFUSenglish\@ordinalMenglish
3697 \global\let\@ordinalNUSenglish\@ordinalMenglish
3698 \global\let\@numberstringMUSenglish\@numberstringMenglish
3699 \global\let\@numberstringFUSenglish\@numberstringMenglish
3700 \global\let\@numberstringNUSenglish\@numberstringMenglish
3701 \global\let\@NumberstringMUSenglish\@NumberstringMenglish
3702 \global\let\@NumberstringFUSenglish\@NumberstringMenglish
3703 \global\let\@NumberstringNUSenglish\@NumberstringMenglish
3704 \global\let\@ordinalstringMUSenglish\@ordinalstringMenglish
3705 \global\let\@ordinalstringFUSenglish\@ordinalstringMenglish
3706 \global\let\@ordinalstringNUSenglish\@ordinalstringMenglish
3707 \global\let\@OrdinalstringMUSenglish\@OrdinalstringMenglish
```

```
3708 \global\let\@OrdinalstringFUSenglish\@OrdinalstringMenglish
3709 \global\let\@OrdinalstringNUSenglish\@OrdinalstringMenglish
```

9.2 fcnumparser.sty

```
3710 \NeedsTeXFormat{LaTeX2e}
3711 \ProvidesPackage{fcnumparser}[2017/06/15]
```

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

```
3712 \def\fc@counter@parser#1{%
3713   \expandafter\fc@number@parser\expandafter{\the#1.}%
3714 }
3715 \newcount\fc@digit@counter
3716
3717 \def\fc@end@{\fc@end}
```

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.

```
3718 \def\fc@number@analysis#1\fc@nil{%
```

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

```
3719   \def\@tempb##1.##2\fc@nil{\def\fc@integer@part{##1}\def\@tempa{##2}}%
3720   \@tempb#1.\fc@end\fc@nil
3721   \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.

```
3722       \def\@tempb##1,##2\fc@nil{\def\fc@integer@part{##1}\def\@tempa{##2}}%
3723       \@tempb#1,\fc@end\fc@nil
3724       \ifx\@tempa\fc@end@
```

No comma either, so fractional part is set empty.

```
3725           \def\fc@fractional@part{}%
3726       \else
```

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

```
3727           \def\@tempb##1,\fc@end{\def\fc@fractional@part{##1}}%
3728           \expandafter\@tempb\@tempa
3729       \fi
3730   \else
```

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

```
3731           \def\@tempb##1.\fc@end{\def\fc@fractional@part{##1}}%
3732           \expandafter\@tempb\@tempa
3733   \fi
3734 }
```

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⟩.

```
3735 \def\fc@number@parser#1{%
```

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

```
3736   \let\@tempa\@empty
3737   \def\@tempb##1##2\fc@nil{%
3738     \def\@tempc{##1}%
3739     \ifx\@tempc\space
3740     \else
3741       \expandafter\def\expandafter\@tempa\expandafter{\@tempa ##1}%
3742     \fi
3743     \def\@tempc{##2}%
3744     \ifx\@tempc\@empty
3745       \expandafter\@gobble
3746     \else
3747       \expandafter\@tempb
3748     \fi
3749     ##2\fc@nil
3750   }%
3751   \@tempb#1\fc@nil
```

Get the sign into `\fc@sign` and the unsigned number part into `\fc@number`.

```
3752   \def\@tempb##1##2\fc@nil{\def\fc@sign{##1}\def\fc@number{##2}}%
3753   \expandafter\@tempb\@tempa\fc@nil
3754   \expandafter\if\fc@sign+%
3755     \def\fc@sign@case{1}%
3756   \else
3757     \expandafter\if\fc@sign-%
3758       \def\fc@sign@case{2}%
3759     \else
3760       \def\fc@sign{}%
3761       \def\fc@sign@case{0}%
3762       \let\fc@number\@tempa
3763     \fi
3764   \fi
3765   \ifx\fc@number\@empty
3766     \PackageError{fcnumparser}{Invalid number}{Number must contain at least one non blank
3767       character after sign}%
3768   \fi
```

Now, split `\fc@number` into `\fc@integer@part` and `\fc@fractional@part`.

```
3769   \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 $2 \times$ `\fc@unit@weight` − `\fc@max@weight` upto `\fc@unit@weight` − 1.

```
3770   \expandafter\fc@digit@counter\fc@unit@weight
3771   \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:

$$\text{\texttt{\textbackslash fc@max@weight}} \leftarrow (-\text{\texttt{\textbackslash fc@digit@counter}}) + 2 \times \text{\texttt{\textbackslash fc@unit@weight}} - 1$$

```
3772    \fc@digit@counter -\fc@digit@counter
3773    \advance\fc@digit@counter by \fc@unit@weight
3774    \advance\fc@digit@counter by \fc@unit@weight
3775    \advance\fc@digit@counter by -1 %
3776    \edef\fc@max@weight{\the\fc@digit@counter}%
```

Now we loop for $i = $ `\fc@unit@weight` to `\fc@max@weight` in order to copy all the digits from `\fc@digit@`$\langle i + \text{offset} \rangle$ to `\fc@digit@`$\langle i \rangle$. First we compute offset into `\@tempi`.

```
3777    {%
3778        \count0 \fc@unit@weight\relax
3779        \count1 \fc@max@weight\relax
3780        \advance\count0 by -\count1 %
3781        \advance\count0 by -1 %
3782        \def\@tempa##1{\def\@tempb{\def\@tempi{##1}}}%
3783        \expandafter\@tempa\expandafter{\the\count0}%
3784        \expandafter
3785    }\@tempb
```

Now we loop to copy the digits. To do that we define a macro `\@templ` for terminal recursion.

```
3786    \expandafter\fc@digit@counter\fc@unit@weight
3787    \def\@templ{%
3788        \ifnum\fc@digit@counter>\fc@max@weight
3789            \let\next\relax
3790        \else
```

Here is the loop body:

```
3791            {%
3792                \count0 \@tempi
3793                \advance\count0 by \fc@digit@counter
3794                \expandafter\def\expandafter\@tempd\expandafter{\csname fc@digit@\the\count0\endcsnam
3795                \expandafter\def\expandafter\@tempe\expandafter{\csname fc@digit@\the\fc@digit@counte
3796                \def\@tempa####1####2{\def\@tempb{\let####1####2}}%
3797                \expandafter\expandafter\expandafter\@tempa\expandafter\@tempe\@tempd
3798                \expandafter
3799            }\@tempb
3800            \advance\fc@digit@counter by 1 %
3801        \fi
3802        \next
3803    }%
3804    \let\next\@templ
3805    \@templ
```

Split `\fc@fractional@part` into a sequence of `\fc@digit@`$\langle n \rangle$ with $\langle n \rangle$ ranging from `\fc@unit@weight`-1 to `\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.

```
3806    \expandafter\fc@digit@counter\fc@unit@weight
3807    \expandafter\fc@parse@integer@digits\fc@fractional@part\fc@end\fc@nil
```

```
3808    \edef\fc@min@weight{\the\fc@digit@counter}%
3809 }
```

Macro `\fc@parse@integer@digits` is used to

```
3810 \ifcsundef{fc@parse@integer@digits}{}{%
3811   \PackageError{fcnumparser}{Duplicate definition}{Redefinition of
3812     macro 'fc@parse@integer@digits'}}
3813 \def\fc@parse@integer@digits#1#2\fc@nil{%
3814   \def\@tempa{#1}%
3815   \ifx\@tempa\fc@end@
3816     \def\next##1\fc@nil{}%
3817   \else
3818   \let\next\fc@parse@integer@digits
3819   \advance\fc@digit@counter by -1
3820   \expandafter\def\csname fc@digit@\the\fc@digit@counter\endcsname{#1}%
3821   \fi
3822   \next#2\fc@nil
3823 }
3824
3825
3826 \newcommand*{\fc@unit@weight}{0}
3827
```

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.

```
3828 \ifcsundef{fc@read@unit}{}{%
3829   \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.

```
3830 \def\fc@read@unit#1#2{%
3831   \ifnum#2>\fc@max@weight
3832     #1=0\relax
3833   \else
3834     \ifnum#2<\fc@min@weight
3835       #1=0\relax
3836     \else
3837       {%
3838         \edef\@tempa{\number#2}%
3839         \count0=\@tempa
3840         \edef\@tempa{\csname fc@digit@\the\count0\endcsname}%
3841         \def\@tempb##1{\def\@tempa{#1=##1\relax}}%
3842         \expandafter\@tempb\expandafter{\@tempa}%
3843         \expandafter
3844       }\@tempa
3845     \fi
```

```
3846     \fi
3847 }
```

<code>fc@read@hundred</code> 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.

```
3848 \ifcsundef{fc@read@hundred}{}{%
3849   \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

```
3850 \def\fc@read@hundred#1#2{%
3851   {%
3852     \fc@read@unit{\count0}{#2}%
3853     \def\@tempa##1{\fc@read@unit{\count1}{##1}}%
3854     \count2=#2%
3855     \advance\count2 by 1 %
3856     \expandafter\@tempa{\the\count2}%
3857     \multiply\count1 by 10 %
3858     \advance\count1 by \count0 %
3859     \def\@tempa##1{\def\@tempb{#1=##1\relax}}
3860     \expandafter\@tempa\expandafter{\the\count1}%
3861     \expandafter
3862   }\@tempb
3863 }
```

<code>c@read@thousand</code> 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.

```
3864 \ifcsundef{fc@read@thousand}{}{%
3865   \PackageError{fcnumparser}{Duplicate definition}{Redefinition of macro
3866     '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

```
3867 \def\fc@read@thousand#1#2{%
3868   {%
3869     \fc@read@unit{\count0}{#2}%
3870     \def\@tempa##1{\fc@read@hundred{\count1}{##1}}%
3871     \count2=#2%
3872     \advance\count2 by 1 %
3873     \expandafter\@tempa{\the\count2}%
3874     \multiply\count1 by 10 %
3875     \advance\count1 by \count0 %
3876     \def\@tempa##1{\def\@tempb{#1=##1\relax}}
3877     \expandafter\@tempa\expandafter{\the\count1}%
3878     \expandafter
3879   }\@tempb
3880 }
```

<code>c@read@thousand</code> 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.

```
3881 \ifcsundef{fc@read@myriad}{}{%
3882   \PackageError{fcnumparser}{Duplicate definition}{Redefinition of macro
3883     '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

```
3884 \def\fc@read@myriad#1#2{%
3885   {%
3886     \fc@read@hundred{\count0}{#2}%
3887     \def\@tempa##1{\fc@read@hundred{\count1}{##1}}%
3888     \count2=#2
3889     \advance\count2 by 2
3890     \expandafter\@tempa{\the\count2}%
3891     \multiply\count1 by 100 %
3892     \advance\count1 by \count0 %
3893     \def\@tempa##1{\def\@tempb{#1=##1\relax}}%
3894     \expandafter\@tempa\expandafter{\the\count1}%
3895     \expandafter
3896   }\@tempb
3897 }
```

@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.

```
3898 \ifcsundef{fc@check@nonzeros}{}{%
3899   \PackageError{fcnumparser}{Duplicate definition}{Redefinition of macro
3900     'fc@check@nonzeros'}}
```

Arguments as follows:

#1   input number: minimum unit unit weight at which start to search the non-zeros

#2   input number: maximum unit weight at which end to seach 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.

```
3901 \def\fc@check@nonzeros#1#2#3{%
3902   {%
```

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

```
3903     \edef\@@tempa{\number#1}%
3904     \edef\@tempb{\number#2}%
3905     \count0=\@@tempa
3906     \count1=\@tempb\relax
```

Then we do the real job

```
3907     \fc@@check@nonzeros@inner
```

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

```
3908     \def\@tempd##1{\def\@tempa{\def#3{##1}}}%
```

110

```
3909     \expandafter\@tempd\expandafter{\@tempc}%
3910     \expandafter
3911   }\@tempa
3912 }
```

\fc@@check@nonzeros@inner Macro `\fc@@check@nonzeros@inner` Check wehther 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 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 seach the non-zeros
	output macro may have been redefined
`\@tempc`	ouput 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.

```
3913 \def\fc@@check@nonzeros@inner{%
3914   \ifnum\count0<\fc@min@weight
3915       \count0=\fc@min@weight\relax
3916   \fi
3917   \ifnum\count1>\fc@max@weight\relax
3918       \count1=\fc@max@weight
3919   \fi
3920   \count2\count0 %
3921   \advance\count2 by 1 %
3922   \ifnum\count0>\count1 %
3923     \PackageError{fcnumparser}{Unexpected arguments}{Number in argument 2 of macro
3924       'fc@check@nonzeros' must be at least equal to number in argument 1}%
3925   \else
3926     \fc@@check@nonzeros@inner@loopbody
3927     \ifnum\@tempc>0 %
3928       \ifnum\@tempc<9 %
3929         \ifnum\count0>\count1 %
3930         \else
3931           \let\@tempd\@tempc
3932           \fc@@check@nonzeros@inner@loopbody
3933           \ifnum\@tempc=0 %
3934             \let\@tempc\@tempd
3935           \else
3936             \def\@tempc{9}%
3937           \fi
3938         \fi
3939       \fi
3940     \fi
3941   \fi
3942 }
3943 \def\fc@@check@nonzeros@inner@loopbody{%
3944   % \@tempc <-  digit of weight \count0
3945   \expandafter\let\expandafter\@tempc\csname fc@digit@\the\count0\endcsname
```

```
3946    \advance\count0 by 1 %
3947    \ifnum\@tempc=0 %
3948        \ifnum\count0>\count1 %
3949            \let\next\relax
3950        \else
3951            \let\next\fc@@check@nonzeros@inner@loopbody
3952        \fi
3953    \else
3954        \ifnum\count0>\count2 %
3955            \def\@tempc{9}%
3956        \fi
3957        \let\next\relax
3958    \fi
3959    \next
3960 }
```

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

```
3961 \ifcsundef{fc@intpart@find@last}{}{%
3962    \PackageError{fcnumparser}{Duplicate definition}{Redefinition of macro
3963    '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@$\langle w \rangle$. Macro \fc@intpart@find@last takes one single argument which is a counter to set to the result.

```
3964 \def\fc@intpart@find@last#1{%
3965    {%
```

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

```
3966        \count0=\fc@unit@weight\space
3967        \ifnum\count0<\fc@min@weight\space
3968            \count0=\fc@min@weight\space
3969        \fi
```

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

```
3970        \def\@templ{%
3971            \ifnum\csname fc@digit@\the\count0\endcsname=0 %
3972                \advance\count0 by 1 %
3973                \ifnum\count0>\fc@max@weight\space
3974                    \let\next\relax
3975                \fi
3976            \else
3977                \let\next\relax
3978            \fi
3979            \next
3980        }%
3981        \let\next\@templ
3982        \@templ
```

Now propagate result after closing bracket into counter #1.

```
3983        \toks0{#1}%
3984        \edef\@tempa{\the\toks0=\the\count0}%
3985        \expandafter
3986   }\@tempa\space
3987 }
```

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

```
3988 \ifcsundef{fc@get@last@word}{}{\PackageError{fcnumparser}{Duplicate definition}{Redefinition
3989     of macro 'fc@get@last@word'}}%
3990 \def\fc@get@last@word#1#2#3{%
3991   {%
```

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.

```
3992       \def\@tempa##1\fc@wcase##2\@nil\fc@end{%
3993         \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.

```
3994         \toks1{##2\fc@wcase}%
3995       }%
3996       \@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.

```
3997       \def\@tempa##1\fc@wcase##2\fc@end{%
3998         \toks2{##1}%
3999         \def\@tempb{##2}%
4000         \toks3{##2}%
4001       }%
```

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

```
4002       \toksdef\@tempt0 %
```

Now the loop itself, this is done by terminal recursion with macro \@templ.

```
4003       \def\@templ{%
4004         \expandafter\@tempa\the\toks1 \fc@end
4005         \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.

```
4006           \let\next\relax
4007         \else
```

\@tempb is not empty, first we use

```
4008           \expandafter\expandafter\expandafter\@tempt
4009           \expandafter\expandafter\expandafter{%
```

```
4010              \expandafter\the\expandafter\@tempt
4011              \expandafter\fc@wcase\the\toks2}%
4012          \toks1\toks3 %
4013        \fi
4014        \next
4015      }%
4016      \let\next\@templ
4017      \@templ
4018      \edef\@tempa{\def\noexpand#2{\the\toks0}\def\noexpand#3{\the\toks2}}%
4019      \expandafter
4020    }\@tempa
4021 }
```

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

```
4022 \ifcsundef{fc@get@last@letter}{}{\PackageError{fcnumparser}{Duplicate definition}{Redefinition
4023      of macro 'fc@get@last@letter'}}%
4024 \def\fc@get@last@letter#1#2#3{%
4025    {%
```

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

```
4026      \toks1{#1}%
4027      \toks0{}%
4028      \toksdef\@tempt0 %
```

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

```
4029      \def\@tempa##1##2\fc@nil{%
4030        \toks2{##1}%
4031        \toks3{##2}%
4032        \def\@tempb{##2}%
4033      }%
```

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

```
4034      \def\@templ{%
4035        \expandafter\@tempa\the\toks1 \fc@nil
4036        \ifx\@tempb\@empty
```

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

```
4037          \let\next\relax
4038        \else
```

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

```
4039          \expandafter\expandafter\expandafter\@tempt
4040          \expandafter\expandafter\expandafter{%
4041            \expandafter\the\expandafter\@tempt
4042            \the\toks2}%
```

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

```
4043          \toks1\toks3 %
```

114

```
4044        \fi
4045        \next
4046      }%
```

Here run the loop.

```
4047      \let\next\@templ
4048      \next
```

Now propagate the results into macros #2 and #3 after closing brace.

```
4049      \edef\@tempa{\def\noexpand#2{\the\toks0}\def\noexpand#3{\the\toks1}}%
4050      \expandafter
4051    }\@tempa
4052 }%
```

9.3 fcprefix.sty

Pseudo-latin prefixes.

```
4053 \NeedsTeXFormat{LaTeX2e}
4054 \ProvidesPackage{fcprefix}[2012/09/28]
4055 \RequirePackage{ifthen}
4056 \RequirePackage{keyval}
4057 \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'.

```
4058 \define@key{fcprefix}{use duode and unde}[below20]{%
4059    \ifthenelse{\equal{#1}{below20}}{%
4060      \def\fc@duodeandunde{2}%
4061    }{%
4062      \ifthenelse{\equal{#1}{never}}{%
4063        \def\fc@duodeandunde{0}%
4064      }{%
4065        \PackageError{fcprefix}{Unexpected option}{%
4066          Option 'use duode and unde' expects 'below 20' or 'never' }%
4067      }%
4068    }%
4069 }
```

Default is 'below 20' like in French.

```
4070 \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.

```
4071 \define@key{fcprefix}{numeral u in duo}[false]{%
4072    \ifthenelse{\equal{#1}{false}}{%
4073      \let\fc@u@in@duo\@empty
4074    }{%
4075      \ifthenelse{\equal{#1}{true}}{%
4076        \def\fc@u@in@duo{u}%
4077      }{%
4078        \PackageError{fcprefix}{Unexpected option}{%
```

```
4079            Option 'numeral u in duo' expects 'true' or 'false' }%
4080      }%
4081    }%
4082 }
```

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.

```
4083 \define@key{fcprefix}{e accute}[false]{%
4084    \ifthenelse{\equal{#1}{false}}{%
4085      \let\fc@prefix@eaccute\@firstofone
4086    }{%
4087      \ifthenelse{\equal{#1}{true}}{%
4088        \let\fc@prefix@eaccute\'%
4089      }{%
4090        \PackageError{fcprefix}{Unexpected option}{%
4091          Option 'e accute' expects 'true' or 'false' }%
4092      }%
4093    }%
4094 }
```

Default is to set accute accent like in French.

```
4095 \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'

```
4096 \define@key{fcprefix}{power of millia}[prefix]{%
4097    \ifthenelse{\equal{#1}{prefix}}{%
4098        \let\fc@power@of@millia@init\@gobbletwo
4099        \let\fc@power@of@millia\fc@@prefix@millia
4100    }{%
4101      \ifthenelse{\equal{#1}{arabic}}{%
4102        \let\fc@power@of@millia@init\@gobbletwo
4103        \let\fc@power@of@millia\fc@@arabic@millia
4104      }{%
4105        \ifthenelse{\equal{#1}{recursive}}{%
4106          \let\fc@power@of@millia@init\fc@@recurse@millia@init
4107          \let\fc@power@of@millia\fc@@recurse@millia
4108        }{%
4109          \PackageError{fcprefix}{Unexpected option}{%
4110            Option 'power of millia' expects 'recursive', 'arabic', or 'prefix' }%
4111        }%
4112      }%
4113    }%
4114 }
```

Arguments as follows:

#1   output macro
#2   number with current weight w

```
4115 \def\fc@@recurse@millia#1#2{%
4116   \let\@tempp#1%
4117   \edef#1{millia\@tempp}%
4118 }
```

Arguments as follows — same interface as `\fc@@recurse@millia`:

#1   output macro
#2   number with current weight w

```
4119 \def\fc@@recurse@millia@init#1#2{%
4120   {%
```

Save input argument current weight w into local macro `\@tempb`.

```
4121     \edef\@tempb{\number#2}%
```

Now main loop from 0 to w. Final value of `\@tempa` will be the result.

```
4122     \count0=0 %
4123     \let\@tempa\@empty
4124     \loop
4125       \ifnum\count0<\@tempb
4126         \advance\count0 by 1 %
4127         \expandafter\def
4128           \expandafter\@tempa\expandafter{\@tempa millia}%
4129     \repeat
```

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

```
4130     \edef\@tempb{\def\noexpand#1{\@tempa}}%
4131     \expandafter
4132   }\@tempb
4133 }
```

Arguments as follows — same interface as `\fc@@recurse@millia`:

#1   output macro
#2   number with current weight w

```
4134 \def\fc@@arabic@millia#1#2{%
4135   \ifnnum#2=0 %
4136     \let#1\@empty
4137   \else
4138     \edef#1{millia\^{}\the#2}%
4139   \fi
4140 }
```

Arguments as follows — same interface as `\fc@@recurse@millia`:

#1   output macro
#2   number with current weight w

```
4141 \def\fc@@prefix@millia#1#2{%
4142   \fc@@latin@numeral@pefix{#2}{#1}%
4143 }
```

Default value of option 'power of millia' is 'prefix':

```
4144 \let\fc@power@of@millia@init\@gobbletwo
4145 \let\fc@power@of@millia\fc@@prefix@millia
```

Compute a cardinal prefix for n-illion, like 1 ⇒ 'm', 2 ⇒ 'bi', 3 ⇒ '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`. First check that macro is not yet defined.

```
4146 \ifcsundef{fc@@latin@cardinal@pefix}{}{%
4147   \PackageError{fmtcount}{Duplicate definition}{Redefinition of macro 'fc@@latin@cardinal@pefix
```

Arguments as follows:

#1   input number to be formated

#2   outut macro name into which to place the formatted result

```
4148 \def\fc@@latin@cardinal@pefix#1#2{%
4149   {%
```

First we put input argument into local macro @cs@tempa with full expansion.

```
4150     \edef\@tempa{\number#1}%
```

Now parse number from expanded input.

```
4151     \expandafter\fc@number@parser\expandafter{\@tempa}%
4152     \count2=0 %
```

`\@tempt` will hold the optional final `t`, `\@tempu` is used to initialize `\@tempt` to 't' when the firt non-zero 3digit group is met, which is the job made by `\@tempi`.

```
4153     \let\@tempt\@empty
4154     \def\@tempu{t}%
```

`\@tempm` will hold the `millia`^$n \div 3$

```
4155     \let\@tempm\@empty
```

Loop by means of terminal recursion of herinafter defined macro `\@templ`. We loop by group of 3 digits.

```
4156     \def\@templ{%
4157       \ifnum\count2>\fc@max@weight
4158         \let\next\relax
4159       \else
```

Loop body. Here we read a group of 3 consecutive digits $d_2 d_1 d_0$ and place them respectively into `\count3`, `\count4`, and `\count5`.

```
4160         \fc@read@unit{\count3}{\count2}%
4161         \advance\count2 by 1 %
4162         \fc@read@unit{\count4}{\count2}%
4163         \advance\count2 by 1 %
4164         \fc@read@unit{\count5}{\count2}%
4165         \advance\count2 by 1 %
```

If the 3 considered digits $d_2 d_1 d_0$ are not all zero, then set `\@tempt` to 't' for the first time this event is met.

```
4166         \edef\@tempn{%
4167           \ifnum\count3=0\else 1\fi
4168           \ifnum\count4=0\else 1\fi
4169           \ifnum\count5=0\else 1\fi
4170         }%
4171         \ifx\@tempn\@empty\else
```

118

```
4172          \let\@tempt\@tempu
4173          \let\@tempu\@empty
4174        \fi
```

Now process the current group $d_2 d_1 d_0$ of 3 digits.

```
4175        \let\@tempp\@tempa
4176        \edef\@tempa{%
```

Here we process d_2 held by \count5, that is to say hundreds.

```
4177          \ifcase\count5 %
4178          \or cen%
4179          \or ducen%
4180          \or trecen%
4181          \or quadringen%
4182          \or quingen%
4183          \or sescen%
4184          \or septigen%
4185          \or octingen%
4186          \or nongen%
4187          \fi
```

Here we process $d_1 d_0$ held by \count4 & \count3, that is to say tens and units.

```
4188          \ifnum\count4=0 %
4189            % x0(0..9)
4190            \ifnum\count2=3 %
4191              % Absolute weight zero
4192              \ifcase\count3 \@tempt
4193              \or m%
4194              \or b%
4195              \or tr%
4196              \or quadr%
4197              \or quin\@tempt
4198              \or sex\@tempt
4199              \or sep\@tempt
4200              \or oc\@tempt
4201              \or non%
4202              \fi
4203            \else
```

Here the weight of \count3 is $3 \times n$, with $n > 0$, i.e. this is followed by a millia^n.

```
4204              \ifcase\count3 %
4205              \or \ifnum\count2>\fc@max@weight\else un\fi
4206              \or d\fc@u@in@duo o%
4207              \or tre%
4208              \or quattuor%
4209              \or quin%
4210              \or sex%
4211              \or septen%
4212              \or octo%
4213              \or novem%
4214              \fi
```

```
4215            \fi
4216         \else
4217            % x(10..99)
4218            \ifcase\count3 %
4219            \or un%
4220            \or d\fc@u@in@duo o%
4221            \or tre%
4222            \or quattuor%
4223            \or quin%
4224            \or sex%
4225            \or septen%
4226            \or octo%
4227            \or novem%
4228            \fi
4229            \ifcase\count4 %
4230            \or dec%
4231            \or vigin\@tempt
4232            \or trigin\@tempt
4233            \or quadragin\@tempt
4234            \or quinquagin\@tempt
4235            \or sexagin\@tempt
4236            \or septuagin\@tempt
4237            \or octogin\@tempt
4238            \or nonagin\@tempt
4239            \fi
4240         \fi
```

Insert the `millia`$^{(n \div 3)}$ only if $d_2 d_1 d_0 \neq 0$, i.e. if one of `\count3` `\count4` or `\count5` is non zero.

```
4241            \@tempm
```

And append previous version of `\@tempa`.

```
4242            \@tempp
4243         }%
```

"Concatenate" `millia` to `\@tempm`, so that `\@tempm` will expand to `millia`$^{(n \div 3)+1}$ at the next iteration. Actually whether this is a concatenation or some `millia` prefixing depends of option 'power of millia'.

```
4244            \fc@power@of@millia\@tempm{\count2}%
4245         \fi
4246         \next
4247      }%
4248      \let\@tempa\@empty
4249      \let\next\@templ
4250      \@templ
```

Propagate expansion of `\@tempa` into #2 after closing bracket.

```
4251      \def\@tempb##1{\def\@tempa{\def#2{##1}}}%
4252      \expandafter\@tempb\expandafter{\@tempa}%
4253      \expandafter
4254   }\@tempa
```

120

```
4255 }
```

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. First check that the macro is not yet defined.

```
4256 \ifcsundef{fc@@latin@numeral@pefix}{}{%
4257   \PackageError{fmtcount}{Duplicate definition}{Redefinition of macro
4258     'fc@@latin@numeral@pefix'}}
```

Arguments as follows:

#1   input number to be formatted,

#2   outut macro name into which to place the result

```
4259 \def\fc@@latin@numeral@pefix#1#2{%
4260   {%
4261     \edef\@tempa{\number#1}%
4262     \def\fc@unit@weight{0}%
4263     \expandafter\fc@number@parser\expandafter{\@tempa}%
4264     \count2=0 %
```

Macro \@tempm will hold the `millies`$^{n \div 3}$.

```
4265     \let\@tempm\@empty
```

Loop over digits. This is done by defining macro \@templ for terminal recursion.

```
4266     \def\@templ{%
4267       \ifnum\count2>\fc@max@weight
4268         \let\next\relax
4269       \else
```

Loop body. Three consecutive digits $d_2 d_1 d_0$ are read into counters \count3, \count4, and \count5.

```
4270         \fc@read@unit{\count3}{\count2}%
4271         \advance\count2 by 1 %
4272         \fc@read@unit{\count4}{\count2}%
4273         \advance\count2 by 1 %
4274         \fc@read@unit{\count5}{\count2}%
4275         \advance\count2 by 1 %
```

Check the use of `duodevicies` instead of `octodecies`.

```
4276         \let\@tempn\@secondoftwo
4277         \ifnum\count3>7 %
4278           \ifnum\count4<\fc@duodeandunde
4279             \ifnum\count4>0 %
4280               \let\@tempn\@firstoftwo
4281             \fi
4282           \fi
4283         \fi
4284         \@tempn
4285         {% use duodevicies for eighteen
4286           \advance\count4 by 1 %
4287           \let\@temps\@secondoftwo
4288         }{%  do not use duodevicies for eighteen
```

121

```
4289            \let\@temps\@firstoftwo
4290          }%
4291          \let\@tempp\@tempa
4292          \edef\@tempa{%
4293            % hundreds
4294            \ifcase\count5 %
4295            \expandafter\@gobble
4296            \or c%
4297            \or duc%
4298            \or trec%
4299            \or quadring%
4300            \or quing%
4301            \or sesc%
4302            \or septing%
4303            \or octing%
4304            \or nong%
4305            \fi
4306            {enties}%
4307            \ifnum\count4=0 %
```

Here $d_2 d_1 d_0$ is such that $d_1 = 0$.

```
4308              \ifcase\count3 %
4309              \or
4310                \ifnum\count2=3 %
4311                  s\fc@prefix@eaccute emel%
4312                \else
4313                  \ifnum\count2>\fc@max@weight\else un\fi
4314                \fi
4315              \or bis%
4316              \or ter%
4317              \or quater%
4318              \or quinquies%
4319              \or sexies%
4320              \or septies%
4321              \or octies%
4322              \or novies%
4323              \fi
4324            \else
```

Here $d_2 d_1 d_0$ is such that $d_1 \geq 1$.

```
4325              \ifcase\count3 %
4326              \or un%
4327              \or d\fc@u@in@duo o%
4328              \or ter%
4329              \or quater%
4330              \or quin%
4331              \or sex%
4332              \or septen%
4333              \or \@temps{octo}{duod\fc@prefix@eaccute e}% x8 = two before next (x+1)0
4334              \or \@temps{novem}{und\fc@prefix@eaccute e}% x9 = one before next (x+1)0
```

122

```
4335            \fi
4336            \ifcase\count4 %
4337            % can't get here
4338            \or d\fc@prefix@eaccute ec%
4339            \or vic%
4340            \or tric%
4341            \or quadrag%
4342            \or quinquag%
4343            \or sexag%
4344            \or septuag%
4345            \or octog%
4346            \or nonag%
4347            \fi
4348            ies%
4349          \fi
4350          % Insert the millies^(n/3) only if one of \count3 \count4 \count5 is non zero
4351          \@tempm
4352          % add up previous version of \@tempa
4353          \@tempp
4354        }%
```

Concatenate `millies` to `\@tempm` so that it is equal to `millies`$^{n \div 3}$ at the next iteration. Here we just have plain concatenation, contrary to cardinal for which a prefix can be used instead.

```
4355          \let\@tempp\@tempp
4356          \edef\@tempm{millies\@tempp}%
4357        \fi
4358      \next
4359    }%
4360    \let\@tempa\@empty
4361    \let\next\@templ
4362    \@templ
```

Now propagate expansion of tempa into #2 after closing bracket.

```
4363    \def\@tempb##1{\def\@tempa{\def#2{##1}}}%
4364    \expandafter\@tempb\expandafter{\@tempa}%
4365    \expandafter
4366  }\@tempa
4367 }
```

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.

```
4368 \def\fc@call@opt@arg@second#1#2{%
4369   \def\@tempb{%
4370     \ifx[\@tempa
4371       \def\@tempc[####1]{%
4372             {#1{####1}{#2}}%
4373           }%
4374     \else
4375       \def\@tempc{{#1{}{#2}}}%
4376     \fi
4377     \@tempc
4378   }%
4379   \futurelet\@tempa
4380   \@tempb
4381 }

4382 \def\fc@call@opt@arg@first#1{%
4383   \def\@tempb{%
4384     \ifx[\@tempa
4385       \def\@tempc[####1]####2{{#1{####1}{####2}}}%
4386     \else
4387       \def\@tempc####1{{#1{}{####1}}}%
4388     \fi
4389     \@tempc
4390   }%
4391   \futurelet\@tempa
4392   \@tempb
4393 }
4394
4395 \let\fc@call\fc@call@opt@arg@first
```

User API.

latinnumeralstringnum Macro \@latinnumeralstringnum. Arguments as follows:

#1	local options
#2	input number

```
4396 \newcommand*{\@latinnumeralstringnum}[2]{%
4397   \setkeys{fcprefix}{#1}%
4398   \fc@@latin@numeral@pefix{#2}\@tempa
4399   \@tempa
4400 }
```

Arguments as follows:

#1	local options
#2	input counter

```
4401 \newcommand*{\@latinnumeralstring}[2]{%
4402   \setkeys{fcprefix}{#1}%
4403   \expandafter\let\expandafter
4404       \@tempa\expandafter\csname c@#2\endcsname
4405   \expandafter\fc@@latin@numeral@pefix\expandafter{\the\@tempa}\@tempa
4406   \@tempa
4407 }

4408 \newcommand*{\latinnumeralstring}{%
4409   \fc@call\@latinnumeralstring
4410 }

4411 \newcommand*{\latinnumeralstringnum}{%
4412   \fc@call\@latinnumeralstringnum
4413 }
```

9.4 fmtcount.sty

This section deals with the code for `fmtcount.sty`

```
4414 \NeedsTeXFormat{LaTeX2e}
4415 \ProvidesPackage{fmtcount}[2017/09/16 v3.04]
4416 \RequirePackage{ifthen}

4417 \RequirePackage{xkeyval}
4418 \RequirePackage{etoolbox}
4419 \RequirePackage{fcprefix}

4420 \RequirePackage{ifxetex}
```

Need to use `\new@ifnextchar` instead of `\@ifnextchar` in commands that have a final optional argument (such as `\gls`) so require `amsgen`.

```
4421 \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`
```
4422 \providecommand*{\fc@orddef@ult}[1]{\fc@textsuperscript{#1}}
```

`c@ord@multiling`
```
4423 \providecommand*{\fc@ord@multiling}[1]{%
4424   \ifcsundef{fc@\languagename @alias@of}{%
```

Not a supported language, just use the default setting:

```
4425   \fc@orddef@ult{#1}}{%
4426   \expandafter\let\expandafter\@tempa\csname fc@\languagename @alias@of\endcsname
4427   \ifcsundef{fc@ord@\@tempa}{%
```

Not language specfic setting, just use the default setting:

```
4428      \fc@orddef@ult{#1}}{%
```

125

Language with specific setting, use that setting:

```
4429 \csname fc@ord@\@tempa\endcsname{#1}}}}
```

<table>
<tr><td>\padzeroes</td><td>\padzeroes[⟨n⟩]</td></tr>
</table>

Specifies how many digits should be displayed for commands such as `\decimal` and `\binary`.

```
4430 \newcount\c@padzeroesN
4431 \c@padzeroesN=1\relax
4432 \providecommand*{\padzeroes}[1][17]{\c@padzeroesN=#1}
```

<table>
<tr><td>\FCloadlang</td><td>\FCloadlang{⟨language⟩}</td></tr>
</table>

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.

```
4433 \newcount\fc@tmpcatcode
4434 \def\fc@languages{}%
4435 \def\fc@mainlang{}%
4436 \newcommand*{\FCloadlang}[1]{%
4437   \@FC@iflangloaded{#1}{}%
4438   {%
4439     \fc@tmpcatcode=\catcode'\@\relax
4440     \catcode '\@ 11\relax
4441     \InputIfFileExists{fc-#1.def}%
4442     {%
4443       \ifdefempty{\fc@languages}%
4444       {%
4445         \gdef\fc@languages{#1}%
4446       }%
4447       {%
4448         \gappto\fc@languages{,#1}%
4449       }%
4450       \gdef\fc@mainlang{#1}%
4451     }%
4452     {}%
4453     \catcode '\@ \fc@tmpcatcode\relax
4454   }%
4455 }
```

<table>
<tr><td>@FC@iflangloaded</td><td>\@FC@iflangloaded{⟨language⟩}{⟨true⟩}{⟨false⟩}</td></tr>
</table>

If fmtcount language definition file fc-⟨*language*⟩.def has been loaded, do ⟨*true*⟩ otherwise do ⟨*false*⟩

```
4456 \newcommand{\@FC@iflangloaded}[3]{%
4457   \ifcsundef{ver@fc-#1.def}{#3}{#2}%
4458 }
```

Declare fmtcount language definition file. Adapted from `\ProvidesFile`.

```
4459 \newcommand*{\ProvidesFCLanguage}[1]{%
4460   \ProvidesFile{fc-#1.def}%
4461 }
```

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

```
4462 \newif\iffmtcount@language@option
4463 \fmtcount@language@optionfalse
```

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 LaTeX 2$_\varepsilon$.

```
4464 \newcommand*\fc@supported@language@list{%
4465 english,%
4466 UKenglish,%
4467 british,%
4468 USenglish,%
4469 american,%
4470 spanish,%
4471 portuges,%
4472 french,%
4473 frenchb,%
4474 francais,%
4475 german,%
4476 germanb,%
4477 ngerman,%
4478 ngermanb,%
4479 italian}
```

```
\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 :

```
4480 \newcommand*\fc@iterate@on@languages[1]{%
4481   \ifx\fc@supported@language@list\@empty
```

That case should never happen !

```
4482     \PackageError{fmtcount}{Macro '\protect\@fc@iterate@on@languages' is empty}{You should neve
4483       Something is broken within \texttt{fmtcount}, please report the issue on
4484       \texttt{https://github.com/search?q=fmtcount\&ref=cmdform\&type=Issues}}%
4485   \else
4486     \let\fc@iterate@on@languages@body#1
4487     \expandafter\@fc@iterate@on@languages\fc@supported@language@list,\@nil,%
4488   \fi
4489 }
4490 \def\@fc@iterate@on@languages#1,{%
4491     {%
4492       \def\@tempa{#1}%
4493       \ifx\@tempa\@nnil
4494         \let\@tempa\@empty
4495       \else
4496         \def\@tempa{%
4497           \fc@iterate@on@languages@body{#1}%
4498           \@fc@iterate@on@languages
4499         }%
4500       \fi
4501       \expandafter
4502     }\@tempa
4503 }%
```

```
\@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.

```
4504 \newcommand*{\@fc@loadifbabelorpolyglossialdf}[1]{%
4505   \ifxetex
4506     \IfFileExists{gloss-#1.ldf}{\ifcsundef{#1@loaded}{}{\FCloadlang{#1}}}{}%
```

```
4507    \else
4508        \ifcsundef{ver@#1.ldf}{}{\FCloadlang{#1}}%
4509    \fi
4510 }
```

Load appropriate language definition files:

```
4511 \fc@iterate@on@languages\@fc@loadifbabelorpolyglossialdf
```

By default all languages are unique — i.e. aliases not yet defined.

```
4512 \def\fc@iterate@on@languages@body#1{%
4513    \expandafter\def\csname fc@#1@alias@of\endcsname{#1}}
4514 \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*⟩}

```
4515 \def\@tempa#1#2{%
4516    \expandafter\def\csname fc@#1@alias@of\endcsname{#2}%
4517 }%
4518 \@tempa{frenchb}{french}
4519 \@tempa{francais}{french}
4520 \@tempa{germanb}{german}
4521 \@tempa{ngermanb}{german}
4522 \@tempa{ngerman}{german}
4523 \@tempa{british}{english}
4524 \@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.

```
4525 \def\fc@iterate@on@languages@body#1{%
4526    \define@key{fmtcount}{#1}[]{%
4527        \@FC@iflangloaded{#1}%
4528        {%
4529            \setkeys{fc\csname fc@#1@alias@of\endcsname}{##1}%
4530        }{%
4531            \PackageError{fmtcount}%
4532            {Language '#1' not defined}%
4533            {You need to load \ifxetex polyglossia\else babel\fi\space before loading fmtcount}%
4534        }%
4535    }%
4536    \ifthenelse{\equal{\csname fc@#1@alias@of\endcsname}{#1}}{%
4537        \define@key{fc\csname fc@#1@alias@of\endcsname}{fmtord}{%
4538            \ifthenelse{\equal{##1}{raise}\or\equal{##1}{level}}{%
4539                \expandafter\let\expandafter\@tempa\csname fc@set@ord@as@##1\endcsname
4540                \expandafter\@tempa\csname fc@ord@#1\endcsname
4541            }{%
4542                \ifthenelse{\equal{##1}{undefine}}{%
4543                    \expandafter\let\csname fc@ord@#1\endcsname\undefined
4544                }{%
4545                    \PackageError{fmtcount}%
4546                    {Invalid value '##1' to fmtord key}%
```

```
4547          {Option 'fmtord' can only take the values 'level', 'raise'
4548            or 'undefine'}%
4549        }}
4550      }%
4551    }{%
```

When the language #1 is an alias, do the same as the language of which it is an alias:

```
4552      \expandafter\let\expandafter\@tempa\csname KV@\csname fc@#1@alias@of\endcsname @fmtord\endc
4553      \expandafter\let\csname KV@#1@fmtord\endcsname\@tempa
4554    }%
4555 }
4556 \expandafter\@fc@iterate@on@languages\fc@supported@language@list,\@nil,%
```

`fmtord`   Key to determine how to display the ordinal

```
4557 \def\fc@set@ord@as@level#1{%
4558   \def#1##1{##1}%
4559 }
4560 \def\fc@set@ord@as@raise#1{%
4561   \let#1\fc@textsuperscript
4562 }
4563 \define@key{fmtcount}{fmtord}{%
4564   \ifthenelse{\equal{#1}{level}
4565           \or\equal{#1}{raise}}%
4566   {%
4567     \csname fc@set@ord@as@#1\endcsname\fc@orddef@ult
4568     \def\fmtcount@fmtord{#1}%
4569   }%
4570   {%
4571     \PackageError{fmtcount}%
4572     {Invalid value '#1' to fmtord key}%
4573     {Option 'fmtord' can only take the values 'level' or 'raise'}%
4574   }%
4575 }
```

`\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.)

```
4576 \newif\iffmtord@abbrv
```

```
4577 \fmtord@abbrvtrue
4578 \define@key{fmtcount}{abbrv}[true]{%
4579   \ifthenelse{\equal{#1}{true}\or\equal{#1}{false}}%
4580   {%
4581     \csname fmtord@abbrv#1\endcsname
4582   }%
4583   {%
4584     \PackageError{fmtcount}%
4585     {Invalid value '#1' to fmtord key}%
4586     {Option 'abbrv' can only take the values 'true' or
4587      'false'}%
```

```
4588    }%
4589 }
```

prefix
```
4590 \define@key{fmtcount}{prefix}[scale=long]{%
4591    \RequirePackage{fmtprefix}%
4592    \fmtprefixsetoption{#1}%
4593 }
```

countsetoptions   Define command to set options.
```
4594 \def\fmtcountsetoptions{%
4595    \def\fmtcount@fmtord{}%
4596    \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.
```
4597 \InputIfFileExists{fmtcount.cfg}%
4598 {%
4599    \PackageInfo{fmtcount}{Using configuration file fmtcount.cfg}%
4600 }%
4601 {%
4602 }
```

ption@lang@list
```
4603 \newcommand*{\fmtcount@loaded@by@option@lang@list}{}
```

\metalanguage   Option ⟨*language*⟩ causes language ⟨*language*⟩ to be registered for loading.
```
4604 \newcommand*\@fc@declare@language@option[1]{%
4605    \DeclareOption{#1}{%
4606       \ifx\fmtcount@loaded@by@option@lang@list\@empty
4607          \def\fmtcount@loaded@by@option@lang@list{#1}%
4608       \else
4609          \edef\fmtcount@loaded@by@option@lang@list{\fmtcount@loaded@by@option@lang@list,#1}%
4610       \fi
4611    }}%
4612 \fc@iterate@on@languages\@fc@declare@language@option
```

level
```
4613 \DeclareOption{level}{\def\fmtcount@fmtord{level}%
4614    \def\fc@orddef@ult#1{#1}}
```

raise
```
4615 \DeclareOption{raise}{\def\fmtcount@fmtord{raise}%
4616    \def\fc@orddef@ult#1{\fc@textsuperscript{#1}}}
```

Process package options
```
4617 \ProcessOptions\relax
```

Now we do the loading of all languages that have been set by option to be loaded.

```
4618 \ifx\fmtcount@loaded@by@option@lang@list\@empty\else
4619 \def\fc@iterate@on@languages@body#1{%
4620     \@FC@iflangloaded{#1}{}{%
4621        \fmtcount@language@optiontrue
4622        \FCloadlang{#1}%
4623     }}
4624 \expandafter\@fc@iterate@on@languages\fmtcount@loaded@by@option@lang@list,\@nil,%
4625 \fi
```

\@FCmodulo

> \@FCmodulo{⟨count reg⟩}{⟨n⟩}

Sets the count register to be its value modulo ⟨n⟩. This is used for the date, time, ordinal and numberstring commands. (The fmtcount package was originally part of the datetime package.)

```
4626 \newcount\@DT@modctr
4627 \newcommand*{\@FCmodulo}[2]{%
4628   \@DT@modctr=#1\relax
4629   \divide \@DT@modctr by #2\relax
4630   \multiply \@DT@modctr by #2\relax
4631   \advance #1 by -\@DT@modctr
4632 }
```

The following registers are needed by \@ordinal etc

```
4633 \newcount\@ordinalctr
4634 \newcount\@orgargctr
4635 \newcount\@strctr
4636 \newcount\@tmpstrctr
```

Define commands that display numbers in different bases. Define counters and conditionals needed.

```
4637 \newif\if@DT@padzeroes
4638 \newcount\@DT@loopN
4639 \newcount\@DT@X
```

\binarynum   Converts a decimal number to binary, and display.

```
4640 \newrobustcmd*{\@binary}[1]{%
4641   \@DT@padzeroestrue
4642   \@DT@loopN=17\relax
4643   \@strctr=\@DT@loopN
4644   \whiledo{\@strctr<\c@padzeroesN}{0\advance\@strctr by \@ne}%
4645   \@strctr=65536\relax
4646   \@DT@X=#1\relax
4647   \loop
4648     \@DT@modctr=\@DT@X
4649     \divide\@DT@modctr by \@strctr
4650     \ifthenelse{\boolean{@DT@padzeroes}
```

132

```
4651        \and \(\@DT@modctr=0\)
4652        \and \(\@DT@loopN>\c@padzeroesN\)}%
4653     {}%
4654     {\the\@DT@modctr}%
4655     \ifnum\@DT@modctr=0\else\@DT@padzeroesfalse\fi
4656     \multiply\@DT@modctr by \@strctr
4657     \advance\@DT@X by -\@DT@modctr
4658     \divide\@strctr by \tw@
4659     \advance\@DT@loopN by \m@ne
4660   \ifnum\@strctr>\@ne
4661   \repeat
4662   \the\@DT@X
4663 }
4664
4665 \let\binarynum=\@binary
```

\octalnum   Converts a decimal number to octal, and displays.

```
4666 \newrobustcmd*{\@octal}[1]{%
4667   \@DT@X=#1\relax
4668   \ifnum\@DT@X>32768
4669     \PackageError{fmtcount}%
4670     {Value of counter too large for \protect\@octal}
4671     {Maximum value 32768}
4672   \else
4673   \@DT@padzeroestrue
4674   \@DT@loopN=6\relax
4675   \@strctr=\@DT@loopN
4676   \whiledo{\@strctr<\c@padzeroesN}{0\advance\@strctr by \@ne}%
4677   \@strctr=32768\relax
4678   \loop
4679     \@DT@modctr=\@DT@X
4680     \divide\@DT@modctr by \@strctr
4681     \ifthenelse{\boolean{@DT@padzeroes}
4682        \and \(\@DT@modctr=0\)
4683        \and \(\@DT@loopN>\c@padzeroesN\)}%
4684     {}{\the\@DT@modctr}%
4685     \ifnum\@DT@modctr=0\else\@DT@padzeroesfalse\fi
4686     \multiply\@DT@modctr by \@strctr
4687     \advance\@DT@X by -\@DT@modctr
4688     \divide\@strctr by \@viiipt
4689     \advance\@DT@loopN by \m@ne
4690   \ifnum\@strctr>\@ne
4691   \repeat
4692   \the\@DT@X
4693   \fi
4694 }
4695 \let\octalnum=\@octal
```

@hexadecimalnum   Converts number from 0 to 15 into lowercase hexadecimal notation.

```
4696 \newcommand*{\@@hexadecimal}[1]{%
4697   \ifcase#10\or1\or2\or3\or4\or5\or
4698   6\or7\or8\or9\or a\or b\or c\or d\or e\or f\fi
4699 }
```

\hexadecimalnum   Converts a decimal number to a lowercase hexadecimal number, and displays it.

```
4700 \newrobustcmd*{\@hexadecimal}[1]{%
4701   \@DT@padzeroestrue
4702   \@DT@loopN=\@vpt
4703   \@strctr=\@DT@loopN
4704   \whiledo{\@strctr<\c@padzeroesN}{0\advance\@strctr by \@ne}%
4705   \@strctr=65536\relax
4706   \@DT@X=#1\relax
4707   \loop
4708     \@DT@modctr=\@DT@X
4709     \divide\@DT@modctr by \@strctr
4710     \ifthenelse{\boolean{@DT@padzeroes}
4711       \and \(\@DT@modctr=0\)
4712       \and \(\@DT@loopN>\c@padzeroesN\)}
4713     {}{\@@hexadecimal\@DT@modctr}%
4714     \ifnum\@DT@modctr=0\else\@DT@padzeroesfalse\fi
4715     \multiply\@DT@modctr by \@strctr
4716     \advance\@DT@X by -\@DT@modctr
4717     \divide\@strctr by 16\relax
4718     \advance\@DT@loopN by \m@ne
4719   \ifnum\@strctr>\@ne
4720   \repeat
4721   \@@hexadecimal\@DT@X
4722 }
4723 \let\hexadecimalnum=\@hexadecimal
```

@Hexadecimalnum   Converts number from 0 to 15 into uppercase hexadecimal notation.

```
4724 \newcommand*{\@@Hexadecimal}[1]{%
4725   \ifcase#10\or1\or2\or3\or4\or5\or6\or
4726   7\or8\or9\or A\or B\or C\or D\or E\or F\fi
4727 }
```

\Hexadecimalnum   Uppercase hexadecimal

```
4728 \newrobustcmd*{\@Hexadecimal}[1]{%
4729   \@DT@padzeroestrue
4730   \@DT@loopN=5\relax
4731   \@strctr=\@DT@loopN
4732   \whiledo{\@strctr<\c@padzeroesN}{0\advance\@strctr by \@ne}%
4733   \@strctr=65536\relax
4734   \@DT@X=#1\relax
4735   \loop
4736     \@DT@modctr=\@DT@X
4737     \divide\@DT@modctr by \@strctr
4738     \ifthenelse{\boolean{@DT@padzeroes}
```

```
4739        \and \(\@DT@modctr=0\)
4740        \and \(\@DT@loopN>\c@padzeroesN\)}%
4741    {}{\@@Hexadecimal\@DT@modctr}%
4742    \ifnum\@DT@modctr=0\else\@DT@padzeroesfalse\fi
4743    \multiply\@DT@modctr by \@strctr
4744    \advance\@DT@X by -\@DT@modctr
4745    \divide\@strctr by 16\relax
4746    \advance\@DT@loopN by \m@ne
4747  \ifnum\@strctr>\@ne
4748  \repeat
4749  \@@Hexadecimal\@DT@X
4750 }
4751
4752 \let\Hexadecimalnum=\@Hexadecimal
```

\aaalphnum   Lowercase alphabetical representation (a ... z aa ... zz)

```
4753 \newrobustcmd*{\@aaalph}{\fc@aaalph\@alph}
4754 \newcommand*\fc@aaalph[2]{%
4755    \@DT@loopN=#2\relax
4756    \@DT@X\@DT@loopN
4757    \advance\@DT@loopN by \m@ne
4758    \divide\@DT@loopN by 26\relax
4759    \@DT@modctr=\@DT@loopN
4760    \multiply\@DT@modctr by 26\relax
4761    \advance\@DT@X by \m@ne
4762    \advance\@DT@X by -\@DT@modctr
4763    \advance\@DT@loopN by \@ne
4764    \advance\@DT@X by \@ne
4765    \edef\@tempa{#1\@DT@X}%
4766    \loop
4767      \@tempa
4768      \advance\@DT@loopN by \m@ne
4769    \ifnum\@DT@loopN>0
4770    \repeat
4771 }
4772
4773 \let\aaalphnum=\@aaalph
```

\AAAlphnum   Uppercase alphabetical representation (a ... z aa ... zz)

```
4774 \newrobustcmd*{\@AAAlph}{\fc@aaalph\@Alph}%
4775
4776 \let\AAAlphnum=\@AAAlph
```

\abalphnum   Lowercase alphabetical representation

```
4777 \newrobustcmd*{\@abalph}{\fc@abalph\@alph}%
4778 \newcommand*\fc@abalph[2]{%
4779    \@DT@X=#2\relax
4780    \ifnum\@DT@X>17576\relax
4781      \ifx#1\@alph\def\@tempa{\@abalph}%
```

```
4782      \else\def\@tempa{\@ABAlph}\fi
4783      \PackageError{fmtcount}%
4784      {Value of counter too large for \expandafter\protect\@tempa}%
4785      {Maximum value 17576}%
4786    \else
4787      \@DT@padzeroestrue
4788      \@strctr=17576\relax
4789      \advance\@DT@X by \m@ne
4790      \loop
4791        \@DT@modctr=\@DT@X
4792        \divide\@DT@modctr by \@strctr
4793        \ifthenelse{\boolean{@DT@padzeroes}
4794          \and \(\@DT@modctr=1\)}%
4795        {}{#1\@DT@modctr}%
4796        \ifnum\@DT@modctr=\@ne\else\@DT@padzeroesfalse\fi
4797        \multiply\@DT@modctr by \@strctr
4798        \advance\@DT@X by -\@DT@modctr
4799        \divide\@strctr by 26\relax
4800      \ifnum\@strctr>\@ne
4801      \repeat
4802      \advance\@DT@X by \@ne
4803      #1\@DT@X
4804    \fi
4805 }
4806
4807 \let\abalphnum=\@abalph
```

\ABAlphnum   Uppercase alphabetical representation

```
4808 \newrobustcmd*{\@ABAlph}{\fc@abalph\@Alph}%
4809 \let\ABAlphnum=\@ABAlph
```

\@fmtc@count   Recursive command to count number of characters in argument. \@strctr should be set to zero before calling it.

```
4810 \def\@fmtc@count#1#2\relax{%
4811   \if\relax#1%
4812   \else
4813     \advance\@strctr by 1\relax
4814     \@fmtc@count#2\relax
4815   \fi
4816 }
```

\@decimal   Format number as a decimal, possibly padded with zeroes in front.

```
4817 \newrobustcmd*{\@decimal}[1]{%
4818   \@strctr=0\relax
4819   \expandafter\@fmtc@count\number#1\relax
4820   \@DT@loopN=\c@padzeroesN
4821   \advance\@DT@loopN by -\@strctr
4822   \ifnum\@DT@loopN>0\relax
4823     \@strctr=0\relax
```

```
4824        \whiledo{\@strctr < \@DT@loopN}{0\advance\@strctr by 1\relax}%
4825    \fi
4826    \number#1\relax
4827 }
4828
4829 \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.

```
4830 \newcommand{\FCordinal}[1]{%
4831    \ordinalnum{%
4832        \the\value{#1}}%
4833 }
```

\ordinal   If \ordinal isn't defined make \ordinal a synonym for \FCordinal to maintain compatibility with previous versions.

```
4834 \ifcsundef{ordinal}
4835 {\let\ordinal\FCordinal}%
4836 {%
4837    \PackageWarning{fmtcount}%
4838    {\protect\ordinal \space already defined use
4839     \protect\FCordinal \space instead.}
4840 }
```

\ordinalnum   Display ordinal where value is given as a number or count register instead of a counter:

```
4841 \newrobustcmd*{\ordinalnum}[1]{%
4842    \new@ifnextchar[%
4843    {\@ordinalnum{#1}}%
4844    {\@ordinalnum{#1}[m]}%
4845 }
```

\@ordinalnum   Display ordinal according to gender (neuter added in v1.1, \xspace added in v1.2, and removed in v1.3[7]):

```
4846 \def\@ordinalnum#1[#2]{%
4847    {%
```

---

[7]I couldn't get it to work consistently both with and without the optional argument

```
4848      \ifthenelse{\equal{#2}{f}}%
4849      {%
4850        \protect\@ordinalF{#1}{\@fc@ordstr}%
4851      }%
4852      {%
4853        \ifthenelse{\equal{#2}{n}}%
4854        {%
4855          \protect\@ordinalN{#1}{\@fc@ordstr}%
4856        }%
4857        {%
4858          \ifthenelse{\equal{#2}{m}}%
4859          {}%
4860          {%
4861            \PackageError{fmtcount}%
4862            {Invalid gender option '#2'}%
4863            {Available options are m, f or n}%
4864          }%
4865          \protect\@ordinalM{#1}{\@fc@ordstr}%
4866        }%
4867      }%
4868      \@fc@ordstr
4869    }%
4870 }
```

\storeordinal   Store the ordinal (first argument is identifying name, second argument is a counter.)

```
4871 \newcommand*\storeordinal}[2]{%
4872    {%
4873      \toks0{\storeordinalnum{#1}}%
4874      \expandafter
4875    }\the\toks0\expandafter{%
4876      \the\value{#2}}%
4877 }
```

storeordinalnum   Store ordinal (first argument is identifying name, second argument is a number or count register.)

```
4878 \newrobustcmd*\storeordinalnum}[2]{%
4879    \@ifnextchar[%
4880    {\@storeordinalnum{#1}{#2}}%
4881    {\@storeordinalnum{#1}{#2}[m]}%
4882 }
```

storeordinalnum   Store ordinal according to gender:

```
4883 \def\@storeordinalnum#1#2[#3]{%
4884    \ifthenelse{\equal{#3}{f}}%
4885    {%
4886      \protect\@ordinalF{#2}{\@fc@ord}
4887    }%
4888    {%
4889      \ifthenelse{\equal{#3}{n}}%
```

```
4890       {%
4891         \protect\@ordinalN{#2}{\@fc@ord}%
4892       }%
4893       {%
4894         \ifthenelse{\equal{#3}{m}}%
4895         {}%
4896         {%
4897           \PackageError{fmtcount}%
4898           {Invalid gender option '#3'}%
4899           {Available options are m or f}%
4900         }%
4901         \protect\@ordinalM{#2}{\@fc@ord}%
4902       }%
4903     }%
4904     \expandafter\let\csname @fcs@#1\endcsname\@fc@ord
4905 }
```

\FMCuse   Get stored information:

```
4906 \newcommand*{\FMCuse}[1]{\csname @fcs@#1\endcsname}
```

\ordinalstring   Display ordinal as a string (argument is a counter)

```
4907 \newcommand*{\ordinalstring}[1]{%
4908   \ordinalstringnum{\expandafter\expandafter\expandafter
4909     \the\value{#1}}%
4910 }
```

rdinalstringnum   Display ordinal as a string (argument is a count register or number.)

```
4911 \newrobustcmd*{\ordinalstringnum}[1]{%
4912   \new@ifnextchar[%
4913   {\@ordinal@string{#1}}%
4914   {\@ordinal@string{#1}[m]}%
4915 }
```

@ordinal@string   Display ordinal as a string according to gender.

```
4916 \def\@ordinal@string#1[#2]{%
4917   {%
4918     \ifthenelse{\equal{#2}{f}}%
4919     {%
4920       \protect\@ordinalstringF{#1}{\@fc@ordstr}%
4921     }%
4922     {%
4923       \ifthenelse{\equal{#2}{n}}%
4924       {%
4925         \protect\@ordinalstringN{#1}{\@fc@ordstr}%
4926       }%
4927       {%
4928         \ifthenelse{\equal{#2}{m}}%
4929         {}%
4930         {%
```

```
4931            \PackageError{fmtcount}%
4932            {Invalid gender option '#2' to \protect\ordinalstring}%
4933            {Available options are m, f or n}%
4934          }%
4935          \protect\@ordinalstringM{#1}{\@fc@ordstr}%
4936        }%
4937      }%
4938      \@fc@ordstr
4939    }%
4940 }
```

reordinalstring   Store textual representation of number. First argument is identifying name, second argument is the counter set to the required number.

```
4941 \newcommand*{\storeordinalstring}[2]{%
4942    {%
4943      \toks0{\storeordinalstringnum{#1}}%
4944      \expandafter
4945    }\the\toks0\expandafter{\the\value{#2}}%
4946 }
```

rdinalstringnum   Store textual representation of number. First argument is identifying name, second argument is a count register or number.

```
4947 \newrobustcmd*{\storeordinalstringnum}[2]{%
4948    \@ifnextchar[%
4949    {\@store@ordinal@string{#1}{#2}}%
4950    {\@store@ordinal@string{#1}{#2}[m]}%
4951 }
```

@ordinal@string   Store textual representation of number according to gender.

```
4952 \def\@store@ordinal@string#1#2[#3]{%
4953    \ifthenelse{\equal{#3}{f}}%
4954    {%
4955      \protect\@ordinalstringF{#2}{\@fc@ordstr}%
4956    }%
4957    {%
4958      \ifthenelse{\equal{#3}{n}}%
4959      {%
4960        \protect\@ordinalstringN{#2}{\@fc@ordstr}%
4961      }%
4962      {%
4963        \ifthenelse{\equal{#3}{m}}%
4964        {}%
4965        {%
4966          \PackageError{fmtcount}%
4967          {Invalid gender option '#3' to \protect\ordinalstring}%
4968          {Available options are m, f or n}%
4969        }%
4970        \protect\@ordinalstringM{#2}{\@fc@ordstr}%
4971      }%
```

140

```
4972   }%
4973   \expandafter\let\csname @fcs@#1\endcsname\@fc@ordstr
4974 }
```

`\Ordinalstring`   Display ordinal as a string with initial letters in upper case (argument is a counter)

```
4975 \newcommand*{\Ordinalstring}[1]{%
4976   \Ordinalstringnum{\expandafter\expandafter\expandafter\the\value{#1}}%
4977 }
```

`rdinalstringnum`   Display ordinal as a string with initial letters in upper case (argument is a number or count register)

```
4978 \newrobustcmd*{\Ordinalstringnum}[1]{%
4979   \new@ifnextchar[%
4980   {\@Ordinal@string{#1}}%
4981   {\@Ordinal@string{#1}[m]}%
4982 }
```

`@Ordinal@string`   Display ordinal as a string with initial letters in upper case according to gender

```
4983 \def\@Ordinal@string#1[#2]{%
4984   {%
4985     \ifthenelse{\equal{#2}{f}}%
4986     {%
4987       \protect\@OrdinalstringF{#1}{\@fc@ordstr}%
4988     }%
4989     {%
4990       \ifthenelse{\equal{#2}{n}}%
4991       {%
4992         \protect\@OrdinalstringN{#1}{\@fc@ordstr}%
4993       }%
4994       {%
4995         \ifthenelse{\equal{#2}{m}}%
4996         {}%
4997         {%
4998           \PackageError{fmtcount}%
4999           {Invalid gender option '#2'}%
5000           {Available options are m, f or n}%
5001         }%
5002         \protect\@OrdinalstringM{#1}{\@fc@ordstr}%
5003       }%
5004     }%
5005     \@fc@ordstr
5006   }%
5007 }
```

`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.

```
5008 \newcommand*{\storeOrdinalstring}[2]{%
5009   {%
```

```
5010      \toks0{\storeOrdinalstringnum{#1}}%
5011      \expandafter
5012   }\the\toks0\expandafter{\the\value{#2}}%
5013 }
```

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.

```
5014 \newrobustcmd*{\storeOrdinalstringnum}[2]{%
5015   \@ifnextchar[%
5016   {\@store@Ordinal@string{#1}{#2}}%
5017   {\@store@Ordinal@string{#1}{#2}[m]}%
5018 }
```

@Ordinal@string   Store textual representation of number according to gender, with initial letters in upper case.

```
5019 \def\@store@Ordinal@string#1#2[#3]{%
5020   \ifthenelse{\equal{#3}{f}}%
5021   {%
5022     \protect\@OrdinalstringF{#2}{\@fc@ordstr}%
5023   }%
5024   {%
5025     \ifthenelse{\equal{#3}{n}}%
5026     {%
5027       \protect\@OrdinalstringN{#2}{\@fc@ordstr}%
5028     }%
5029     {%
5030       \ifthenelse{\equal{#3}{m}}%
5031       {}%
5032       {%
5033         \PackageError{fmtcount}%
5034         {Invalid gender option '#3'}%
5035         {Available options are m or f}%
5036       }%
5037       \protect\@OrdinalstringM{#2}{\@fc@ordstr}%
5038     }%
5039   }%
5040   \expandafter\let\csname @fcs@#1\endcsname\@fc@ordstr
5041 }
```

reORDINALstring   Store upper case textual representation of ordinal. The first argument is identifying name, the second argument is a counter.

```
5042 \newcommand*{\storeORDINALstring}[2]{%
5043   {%
5044     \toks0{\storeORDINALstringnum{#1}}%
5045     \expandafter
5046   }\the\toks0\expandafter{\the\value{#2}}%
5047 }
```

RDINALstringnum   As above, but the second argument is a count register or a number.

```
5048 \newrobustcmd*{\storeORDINALstringnum}[2]{%
5049   \@ifnextchar[%
5050   {\@store@ORDINAL@string{#1}{#2}}%
5051   {\@store@ORDINAL@string{#1}{#2}[m]}%
5052 }
```

@ORDINAL@string Gender is specified as an optional argument at the end.

```
5053 \def\@store@ORDINAL@string#1#2[#3]{%
5054   \ifthenelse{\equal{#3}{f}}%
5055   {%
5056     \protect\@ordinalstringF{#2}{\@fc@ordstr}%
5057   }%
5058   {%
5059     \ifthenelse{\equal{#3}{n}}%
5060     {%
5061       \protect\@ordinalstringN{#2}{\@fc@ordstr}%
5062     }%
5063     {%
5064       \ifthenelse{\equal{#3}{m}}%
5065       {}%
5066       {%
5067         \PackageError{fmtcount}%
5068         {Invalid gender option '#3'}%
5069         {Available options are m or f}%
5070       }%
5071       \protect\@ordinalstringM{#2}{\@fc@ordstr}%
5072     }%
5073   }%
5074
5075   \expandafter\protected@edef\csname @fcs@#1\endcsname{%
5076     \noexpand\MakeUppercase{\@fc@ordstr}%
5076   }%
5077 }
```

\ORDINALstring Display upper case textual representation of an ordinal. The argument must be a counter.

```
5078 \newcommand*{\ORDINALstring}[1]{%
5079   \ORDINALstringnum{\expandafter\expandafter\expandafter
5080     \the\value{#1}%
5081   }%
5082 }
```

RDINALstringnum As above, but the argument is a count register or a number.

```
5083 \newrobustcmd*{\ORDINALstringnum}[1]{%
5084   \new@ifnextchar[%
5085   {\@ORDINAL@string{#1}}%
5086   {\@ORDINAL@string{#1}[m]}%
5087 }
```

@ORDINAL@string Gender is specified as an optional argument at the end.

```
5088 \def\@ORDINAL@string#1[#2]{%
5089   {%
5090     \ifthenelse{\equal{#2}{f}}%
5091     {%
5092       \protect\@ordinalstringF{#1}{\@fc@ordstr}%
5093     }%
5094     {%
5095       \ifthenelse{\equal{#2}{n}}%
5096       {%
5097         \protect\@ordinalstringN{#1}{\@fc@ordstr}%
5098       }%
5099       {%
5100         \ifthenelse{\equal{#2}{m}}%
5101         {}%
5102         {%
5103           \PackageError{fmtcount}%
5104           {Invalid gender option '#2'}%
5105           {Available options are m, f or n}%
5106         }%
5107         \protect\@ordinalstringM{#1}{\@fc@ordstr}%
5108       }%
5109     }%
5110     \MakeUppercase{\@fc@ordstr}%
5111   }%
5112 }
```

Convert number to textual respresentation, and store. First argument is the identifying name, second argument is a counter containing the number.

```
5113 \newcommand*{\storenumberstring}[2]{%
5114   \expandafter\protect\expandafter\storenumberstringnum{#1}{%
5115     \expandafter\the\value{#2}}%
5116 }
```

As above, but second argument is a number or count register.

```
5117 \newcommand{\storenumberstringnum}[2]{%
5118   \@ifnextchar[%
5119   {\@store@number@string{#1}{#2}}%
5120   {\@store@number@string{#1}{#2}[m]}%
5121 }
```

Gender is given as optional argument, *at the end*.

```
5122 \def\@store@number@string#1#2[#3]{%
5123   \ifthenelse{\equal{#3}{f}}%
5124   {%
5125     \protect\@numberstringF{#2}{\@fc@numstr}%
5126   }%
5127   {%
5128     \ifthenelse{\equal{#3}{n}}%
5129     {%
```

```
5130        \protect\@numberstringN{#2}{\@fc@numstr}%
5131      }%
5132      {%
5133        \ifthenelse{\equal{#3}{m}}%
5134        {}%
5135        {%
5136          \PackageError{fmtcount}
5137          {Invalid gender option '#3'}%
5138          {Available options are m, f or n}%
5139        }%
5140        \protect\@numberstringM{#2}{\@fc@numstr}%
5141      }%
5142    }%
5143    \expandafter\let\csname @fcs@#1\endcsname\@fc@numstr
5144 }
```

\numberstring   Display textual representation of a number. The argument must be a counter.

```
5145 \newcommand*{\numberstring}[1]{%
5146   \numberstringnum{\expandafter\expandafter\expandafter
5147     \the\value{#1}}%
5148 }
```

numberstringnum   As above, but the argument is a count register or a number.

```
5149 \newrobustcmd*{\numberstringnum}[1]{%
5150   \new@ifnextchar[%
5151   {\@number@string{#1}}%
5152   {\@number@string{#1}[m]}%
5153 }
```

\@number@string   Gender is specified as an optional argument *at the end*.

```
5154 \def\@number@string#1[#2]{%
5155   {%
5156     \ifthenelse{\equal{#2}{f}}%
5157     {%
5158       \protect\@numberstringF{#1}{\@fc@numstr}%
5159     }%
5160     {%
5161       \ifthenelse{\equal{#2}{n}}%
5162       {%
5163         \protect\@numberstringN{#1}{\@fc@numstr}%
5164       }%
5165       {%
5166         \ifthenelse{\equal{#2}{m}}%
5167         {}%
5168         {%
5169          \PackageError{fmtcount}%
5170          {Invalid gender option '#2'}%
5171          {Available options are m, f or n}%
5172         }%
```

```
5173          \protect\@numberstringM{#1}{\@fc@numstr}%
5174        }%
5175      }%
5176      \@fc@numstr
5177   }%
5178 }
```

oreNumberstring   Store textual representation of number. First argument is identifying name, second argument is a counter.

```
5179 \newcommand*{\storeNumberstring}[2]{%
5180   {%
5181     \toks0{\storeNumberstringnum{#1}}%
5182     \expandafter
5183   }\the\toks0\expandafter{\the\value{#2}}%
5184 }
```

Numberstringnum   As above, but second argument is a count register or number.

```
5185 \newcommand{\storeNumberstringnum}[2]{%
5186   \@ifnextchar[%
5187   {\@store@Number@string{#1}{#2}}%
5188   {\@store@Number@string{#1}{#2}[m]}%
5189 }
```

e@Number@string   Gender is specified as an optional argument *at the end*:

```
5190 \def\@store@Number@string#1#2[#3]{%
5191   \ifthenelse{\equal{#3}{f}}%
5192   {%
5193     \protect\@NumberstringF{#2}{\@fc@numstr}%
5194   }%
5195   {%
5196     \ifthenelse{\equal{#3}{n}}%
5197     {%
5198       \protect\@NumberstringN{#2}{\@fc@numstr}%
5199     }%
5200     {%
5201       \ifthenelse{\equal{#3}{m}}%
5202       {}%
5203       {%
5204         \PackageError{fmtcount}%
5205         {Invalid gender option '#3'}%
5206         {Available options are m, f or n}%
5207       }%
5208       \protect\@NumberstringM{#2}{\@fc@numstr}%
5209     }%
5210   }%
5211   \expandafter\let\csname @fcs@#1\endcsname\@fc@numstr
5212 }
```

\Numberstring   Display textual representation of number. The argument must be a counter.

146

```
5213 \newcommand*{\Numberstring}[1]{%
5214   \Numberstringnum{\expandafter\expandafter\expandafter
5215     \the\value{#1}}%
5216 }
```

Numberstringnum   As above, but the argument is a count register or number.

```
5217 \newrobustcmd*{\Numberstringnum}[1]{%
5218   \new@ifnextchar[%
5219   {\@Number@string{#1}}%
5220   {\@Number@string{#1}[m]}%
5221 }
```

\@Number@string   Gender is specified as an optional argument at the end.

```
5222 \def\@Number@string#1[#2]{%
5223   {%
5224     \ifthenelse{\equal{#2}{f}}%
5225     {%
5226       \protect\@NumberstringF{#1}{\@fc@numstr}%
5227     }%
5228     {%
5229       \ifthenelse{\equal{#2}{n}}%
5230       {%
5231         \protect\@NumberstringN{#1}{\@fc@numstr}%
5232       }%
5233       {%
5234         \ifthenelse{\equal{#2}{m}}%
5235         {}%
5236         {%
5237           \PackageError{fmtcount}%
5238           {Invalid gender option '#2'}%
5239           {Available options are m, f or n}%
5240         }%
5241         \protect\@NumberstringM{#1}{\@fc@numstr}%
5242       }%
5243     }%
5244     \@fc@numstr
5245   }%
5246 }
```

oreNUMBERstring   Store upper case textual representation of number. The first argument is identifying name, the second argument is a counter.

```
5247 \newcommand{\storeNUMBERstring}[2]{%
5248   {%
5249     \toks0{\storeNUMBERstringnum{#1}}%
5250     \expandafter
5251   }\the\toks0\expandafter{\the\value{#2}}%
5252 }
```

NUMBERstringnum   As above, but the second argument is a count register or a number.

```
5253 \newcommand{\storeNUMBERstringnum}[2]{%
5254   \@ifnextchar[%
5255   {\@store@NUMBER@string{#1}{#2}}%
5256   {\@store@NUMBER@string{#1}{#2}[m]}%
5257 }
```

e@NUMBER@string   Gender is specified as an optional argument at the end.

```
5258 \def\@store@NUMBER@string#1#2[#3]{%
5259   \ifthenelse{\equal{#3}{f}}%
5260   {%
5261     \protect\@numberstringF{#2}{\@fc@numstr}%
5262   }%
5263   {%
5264     \ifthenelse{\equal{#3}{n}}%
5265     {%
5266       \protect\@numberstringN{#2}{\@fc@numstr}%
5267     }%
5268     {%
5269       \ifthenelse{\equal{#3}{m}}%
5270       {}%
5271       {%
5272         \PackageError{fmtcount}%
5273         {Invalid gender option '#3'}%
5274         {Available options are m or f}%
5275       }%
5276       \protect\@numberstringM{#2}{\@fc@numstr}%
5277     }%
5278   }%
5279   \expandafter\edef\csname @fcs@#1\endcsname{%
5280     \noexpand\MakeUppercase{\@fc@numstr}%
5281   }%
5282 }
```

\NUMBERstring   Display upper case textual representation of a number. The argument must be a counter.

```
5283 \newcommand*{\NUMBERstring}[1]{%
5284   \NUMBERstringnum{\expandafter\expandafter\expandafter
5285     \the\value{#1}}%
5286 }
```

NUMBERstringnum   As above, but the argument is a count register or a number.

```
5287 \newrobustcmd*{\NUMBERstringnum}[1]{%
5288   \new@ifnextchar[%
5289   {\@NUMBER@string{#1}}%
5290   {\@NUMBER@string{#1}[m]}%
5291 }
```

\@NUMBER@string   Gender is specified as an optional argument at the end.

```
5292 \def\@NUMBER@string#1[#2]{%
5293   {%
```

```
5294      \ifthenelse{\equal{#2}{f}}%
5295      {%
5296        \protect\@numberstringF{#1}{\@fc@numstr}%
5297      }%
5298      {%
5299        \ifthenelse{\equal{#2}{n}}%
5300        {%
5301          \protect\@numberstringN{#1}{\@fc@numstr}%
5302        }%
5303        {%
5304          \ifthenelse{\equal{#2}{m}}%
5305          {}%
5306          {%
5307            \PackageError{fmtcount}%
5308            {Invalid gender option '#2'}%
5309            {Available options are m, f or n}%
5310          }%
5311          \protect\@numberstringM{#1}{\@fc@numstr}%
5312        }%
5313      }%
5314      \MakeUppercase{\@fc@numstr}%
5315    }%
5316 }
```

\binary   Number representations in other bases. Binary:

```
5317 \providecommand*{\binary}[1]{%
5318   \@binary{\expandafter\expandafter\expandafter
5319     \the\value{#1}}%
5320 }
```

\aaalph   Like \alph, but goes beyond 26. (a ... z aa ...zz ...)

```
5321 \providecommand*{\aaalph}[1]{%
5322   \@aaalph{\expandafter\expandafter\expandafter
5323     \the\value{#1}}%
5324 }
```

\AAAlph   As before, but upper case.

```
5325 \providecommand*{\AAAlph}[1]{%
5326   \@AAAlph{\expandafter\expandafter\expandafter
5327     \the\value{#1}}%
5328 }
```

\abalph   Like \alph, but goes beyond 26. (a ... z ab ...az ...)

```
5329 \providecommand*{\abalph}[1]{%
5330   \@abalph{\expandafter\expandafter\expandafter
5331     \the\value{#1}}%
5332 }
```

\ABAlph   As above, but upper case.

```
5333 \providecommand*{\ABAlph}[1]{%
5334   \@ABAlph{\expandafter\expandafter\expandafter
5335     \the\value{#1}}%
5336 }
```

\hexadecimal   Hexadecimal:

```
5337 \providecommand*{\hexadecimal}[1]{%
5338   \@hexadecimal{\expandafter\expandafter\expandafter
5339     \the\value{#1}}%
5340 }
```

\Hexadecimal   As above, but in upper case.

```
5341 \providecommand*{\Hexadecimal}[1]{%
5342   \@Hexadecimal{\expandafter\expandafter\expandafter
5343     \the\value{#1}}%
5344 }
```

\octal   Octal:

```
5345 \providecommand*{\octal}[1]{%
5346   \@octal{\expandafter\expandafter\expandafter
5347     \the\value{#1}}%
5348 }
```

\decimal   Decimal:

```
5349 \providecommand*{\decimal}[1]{%
5350   \@decimal{\expandafter\expandafter\expandafter
5351     \the\value{#1}}%
5352 }
```

9.4.1 Multilinguage Definitions

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.

```
5353 \def\@setdef@ultfmtcount{%
5354   \ifcsundef{@ordinalMenglish}{\FCloadlang{english}}{}%
5355   \def\@ordinalstringM{\@ordinalstringMenglish}%
5356   \let\@ordinalstringF=\@ordinalstringMenglish
5357   \let\@ordinalstringN=\@ordinalstringMenglish
5358   \def\@OrdinalstringM{\@OrdinalstringMenglish}%
5359   \let\@OrdinalstringF=\@OrdinalstringMenglish
5360   \let\@OrdinalstringN=\@OrdinalstringMenglish
5361   \def\@numberstringM{\@numberstringMenglish}%
5362   \let\@numberstringF=\@numberstringMenglish
5363   \let\@numberstringN=\@numberstringMenglish
5364   \def\@NumberstringM{\@NumberstringMenglish}%
5365   \let\@NumberstringF=\@NumberstringMenglish
5366   \let\@NumberstringN=\@NumberstringMenglish
```

```
5367    \def\@ordinalM{\@ordinalMenglish}%
5368    \let\@ordinalF=\@ordinalM
5369    \let\@ordinalN=\@ordinalM
5370    \let\fmtord\fc@orddef@ult
5371 }
```

\fc@multiling   \fc@multiling{⟨*name*⟩}{⟨*gender*⟩}

```
5372 \newcommand*{\fc@multiling}[2]{%
5373    \ifcsundef{@#1#2\languagename}%
5374    {% try loading it
5375       \FCloadlang{\languagename}%
5376    }%
5377    {%
5378    }%
5379    \ifcsundef{@#1#2\languagename}%
5380    {%
5381       \PackageWarning{fmtcount}%
5382       {No support for \expandafter\protect\csname #1\endcsname\space for
5383        language '\languagename'}%
5384       \ifthenelse{\equal{\languagename}{\fc@mainlang}}%
5385       {%
5386          \FCloadlang{english}%
5387       }%
5388       {%
5389       }%
5390       \ifcsdef{@#1#2\fc@mainlang}%
5391       {%
5392          \csuse{@#1#2\fc@mainlang}%
5393       }%
5394       {%
5395          \PackageWarningNoLine{fmtcount}%
5396          {No languages loaded at all! Loading english definitions}%
5397          \FCloadlang{english}%
5398          \def\fc@mainlang{english}%
5399          \csuse{@#1#2english}%
5400       }%
5401    }%
5402    {%
5403       \csuse{@#1#2\languagename}%
5404    }%
5405 }
```

itling@fmtcount   This defines the number and ordinal string macros to use \languagename:

```
5406 \def\@set@mulitling@fmtcount{%
```

The masculine version of \numberstring:

```
5407    \def\@numberstringM{%
5408       \fc@multiling{numberstring}{M}%
5409    }%
```

151

The feminine version of \numberstring:

```
5410   \def\@numberstringF{%
5411      \fc@multiling{numberstring}{F}%
5412   }%
```

The neuter version of \numberstring:

```
5413   \def\@numberstringN{%
5414      \fc@multiling{numberstring}{N}%
5415   }%
```

The masculine version of \Numberstring:

```
5416   \def\@NumberstringM{%
5417      \fc@multiling{Numberstring}{M}%
5418   }%
```

The feminine version of \Numberstring:

```
5419   \def\@NumberstringF{%
5420      \fc@multiling{Numberstring}{F}%
5421   }%
```

The neuter version of \Numberstring:

```
5422   \def\@NumberstringN{%
5423      \fc@multiling{Numberstring}{N}%
5424   }%
```

The masculine version of \ordinal:

```
5425   \def\@ordinalM{%
5426      \fc@multiling{ordinal}{M}%
5427   }%
```

The feminine version of \ordinal:

```
5428   \def\@ordinalF{%
5429      \fc@multiling{ordinal}{F}%
5430   }%
```

The neuter version of \ordinal:

```
5431   \def\@ordinalN{%
5432      \fc@multiling{ordinal}{N}%
5433   }%
```

The masculine version of \ordinalstring:

```
5434   \def\@ordinalstringM{%
5435      \fc@multiling{ordinalstring}{M}%
5436   }%
```

The feminine version of \ordinalstring:

```
5437   \def\@ordinalstringF{%
5438      \fc@multiling{ordinalstring}{F}%
5439   }%
```

The neuter version of \ordinalstring:

```
5440   \def\@ordinalstringN{%
5441      \fc@multiling{ordinalstring}{N}%
5442   }%
```

The masculine version of `\Ordinalstring`:

```
5443    \def\@OrdinalstringM{%
5444       \fc@multiling{Ordinalstring}{M}%
5445    }%
```

The feminine version of `\Ordinalstring`:

```
5446    \def\@OrdinalstringF{%
5447       \fc@multiling{Ordinalstring}{F}%
5448    }%
```

The neuter version of `\Ordinalstring`:

```
5449    \def\@OrdinalstringN{%
5450       \fc@multiling{Ordinalstring}{N}%
5451    }%
```

Make `\fmtord` language dependent:

```
5452    \let\fmtord\fc@ord@multiling
5453 }
```

Check to see if `babel`, polyglossia or `ngerman` packages have been loaded, and if yes set fmt-count in multiling.

```
5454 \expandafter\@ifpackageloaded
5455 \expandafter{\ifxetex polyglossia\else babel\fi}%
5456 {%
5457   \@set@mulitling@fmtcount
5458 }%
5459 {%
5460   \@ifpackageloaded{ngerman}%
5461   {%
5462     \FCloadlang{ngerman}%
5463     \@set@mulitling@fmtcount
5464   }%
5465   {%
```

In the case that neither babel/polyglossia, nor ngerman has been loaded, then we go to multiling if a language has been loaded by package option, and to delfault language otherwise.

```
5466       \iffmtcount@language@option
5467           \@set@mulitling@fmtcount
```

Some sanity check at the beginning of document may help the end user understand what is wrong:

```
5468        \AtBeginDocument{%
5469          \ifcsundef{languagename}%
5470          {%
5471            \PackageWarning{fmtcount}{%
5472              '\protect\languagename' is undefined, you should use package babel/polyglossia wh
5473              language via package option. Reverting to default language.
5474            }%
5475            \@setdef@ultfmtcount
5476          }{%
5477            \@FC@iflangloaded{\languagename}{}{%
```

The current \languagename is not a 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.

```
5478                 \PackageWarning{fmtcount}{%
5479                   Setting '\protect\languagename' to '\fc@mainlang'.\MessageBreak
5480                   Reason is that '\protect\languagename' was '\languagename',\MessageBreak
5481                   but '\languagename' was not loaded by fmtcount,\MessageBreak
5482                   whereas '\fc@mainlang' was the last language loaded by fmtcount ;
5483                 }%
5484                 \let\languagename\fc@mainlang
5485               }
5486             }%
5487           }
5488     \else
5489        \@setdef@ultfmtcount
5490     \fi
5491   }%
5492 }

5493 \AtBeginDocument{%
5494    \ifcsundef{FBsupR}{\let\fc@textsuperscript\textsuperscript}{\let\fc@textsuperscript\fup}%
5495 }
```

Backwards compatibility:

```
5496 \let\@ordinal=\@ordinalM
5497 \let\@ordinalstring=\@ordinalstringM
5498 \let\@Ordinalstring=\@OrdinalstringM
5499 \let\@numberstring=\@numberstringM
5500 \let\@Numberstring=\@NumberstringM
```