The cooking-units package*

Ben Vitecek
b.vitecek@gmx.at

September 6, 2016

Abstract

This package enables user to globally format units and to switch between them. It should be used for light-hearted things like cookery books (and not e.g. scientific texts).[1]

Contents

*This document corresponds to Benedikt Vitecek v1.02, dated 2016/09/05.
[1]I did hide some grammatical and spelling errors for easter egg hunters ☺.

1 Introduction

While writing on a cookery book I used – for reasons whatsoever – three different units for weight: kilogram (kg), gram (g) and decagram (dag, or older: dkg). Later my mother told me that she doesn't like it if a cookery book uses more than two different units (for weight in this case). Happily I hardly used Decagram and therefore didn't have many problems changing the units. But, well ... I am using LaTeX and changing those units by hand seemed not very LaTeXlike, so I started writing some code to convert units. I expanded the code, rewrote it in LaTeX3 (which is much more pleasant than LaTeX 2_ε) and here it is.

2 Important note

This package uses the translator package to be able to switch between different languages (german, english and french by now). To do that translator has to know which languages are used. The (I think) easiest way is to specify used languages using the optional argument of the documentclass (you can do this for both babel and polyglossia[2]):

```
\documentclass[english,french,ngerman]{class}
\usepackage[main=english]{babel}
\usepackage{cooking-units}
...
```

or if you are using polyglossia

```
\documentclass[french,ngerman,english]{class}

\usepackage{polyglossia}
\setmainlanguage{english}
\setotherlanguages{german,french}

\usepackage{cooking-units}
...
```

At least I hope that this works, dealing with languages is a pain in the ass[3].

2.1 Supported languages

- German
- English
- French (currently suboptimal[4])

Have another language to add or a correction of an existing one? See 9 on page 18 for more details.

[2] I know that polyglossia doesn't support the babel-way of setting the language by optional argument, but it doesn't harm.

[3] If you excuse me being blunt about this

[4] You can only get limited information from the internet.

3 The Commands

This package offers the following commands for unit printing (and converting):

- `\cunum[`⟨*options*⟩`]{`⟨*amount*⟩`}[`⟨*space*⟩`]{`⟨*unit-key*⟩`}`
- `\cutext[`⟨*options*⟩`]{`⟨*amount*⟩`}{`⟨*unit-key*⟩`}`
- `\Cutext[`⟨*options*⟩`]{`⟨*amount*⟩`}{`⟨*unit-key*⟩`}`
- `\cuam[`⟨*options*⟩`] {`⟨*amount*⟩`}`

Numbers and units are printed using `\cunum`. The numerical part can interpret `_` and `/` as (mixed) fractions and `--` as a separator for ranges; to convert units use the option ⟨*old-unit*⟩=⟨*new-unit*⟩[5]. It furthermore allows the sign `?` to be used as a placeholder for not known amounts and raises a warning to remind that this amount needs a checkup[6]. `[`⟨*space*⟩`]` adds a space between the number and the unit using `\phantom`.

For a list of predefined units have a look at table 3 on page 6.

1 kg	`\cunum{1}{kg}\\`
2.3 kg	`\cunum{2.3}{kg}\\`
2.3 kg	`\cunum{2,3}{kg}\\`
2–3 kg	`\cunum{2--3}{kg}\\`
2.5–3.5 kg	`\cunum{2.5--3.5}{kg}\\`
2500–3500 g	`\cunum[kg=g]{2.5--3.5}{kg}\\`
392 °F	`\cunum[C=F]{200}{C}\\`
356–392 °F	`\cunum[C=F]{180--200}{C}\\`
½ m	`\cunum{1/2}{m}\\`
1 ½ m	`\cunum{1_1/2}{m}\\`
1 ½ m	`\cunum[m=cm]{1_1/2}{m}\\`
? ℓ	`\cunum{?}{l}\\`
50 dag	`\cunum{50}{dag}\\`
5 dag	`\cunum{5}[0]{dag}\\`
1.12 m	`\cunum{1.1234}{m}`

Decimal numbers are automatically rounded to 2 digits after the colon, temperatures (`C`, `F`, `K` and `Re`) are automatically rounded to integers.[7]

`\cutext` and `\Cutext` print the number and the written name of the unit. Furthermore, if the package option `use-numerals` is used, integers below a specific integer (by default 13; see `use-numerals-below`) are also written out with `\Cutext` capitalizing the first letter (using package fmtcount). Conversion between units is not supported. Without `use-numerals`:

[5] New keys can be added and defined, see 4 on the next page and 5 on the following page for further information.

[6] You can customize this behavior, see 8 on page 12

[7] You can – of course – change this behavior, see 8 on page 12.

1 litre	`\cutext{1}{l}\\`
1 litre	`\Cutext{1}{l}\\`
12 litres	`\cutext{12}{l}\\`
13 litres	`\Cutext{13}{l}`

and using package option `use-numerals=true`

one litre	`\cutext{1}{l}\\`
One litre	`\Cutext{1}{l}\\`
twelve litres	`\cutext{12}{l}\\`
13 litres	`\Cutext{13}{l}`

`\cuam` works like a more primitive version `\cunum` which doesn't need a unit, but doesn't check the input like `\cunum`. Like in `\cunum` `_` and `/` are used to imply a (mixed) fraction and `--` used to print ranges[8]:

3	`\cuam{3}\\`
2–3	`\cuam{2--3}\\`
⅔	`\cuam{2/3}\\`
1⅔	`\cuam{1_2/3}`

4 Predefined units & some notes

In table 3 on page 6 and table 2 on page 6 you can find all predefined units.

I now *did* include a separate key for "Messerspitze" (Msp.) and therefore separated "Pinch" (pn) and "Messerspitze" (Msp.). My biggest problems with the units given in table 2 on page 6 is that they may only exist in one language (or country) and therefore do not exist in another language (I think for example that knife point "Messerspitze" doesn't exist in english) so translating them would be difficult. Therefore use units known to you and if there are unsupported units or languages feel free to write (see 9 on page 18 for more details).

5 Defining units

New units can be defined using `\newcookingunit`:

`\newcookingunit` `\newcookingunit[⟨symbol⟩]{⟨new-unit-key⟩}`

This command defines the new unit ⟨*new-unit-key*⟩. If the key is not the same as the printed symbol use `[⟨symbol⟩]`.

Some examples (note: the definition of the printed degree Celsius is directly copied & pasted from [a maybe older version of] siunitx):

```
\newcookingunit{kg}
\newcookingunit{g}
\newcookingunit[Msp.] {Msp}
\newcookingunit[\ensuremath{{}^{\circ}}\kern-\scriptspace C] {C}
```

[8] Note that since v1.02 `\myfrac` is obsolete.

Table 1: List of predefined unit-keys. The "symbol" column is language dependent. Note that "electron volt" exists just for fun.

unit name	unit-key	symbol
kilogramme	kg	kg
decagramme	dag	dag
gramme	g	g
ounce	oz	oz
pound	lb	lb
degree Celsius	C	°C
degree Fahrenheit	F	°F
degree Réaumur	Re	°Ré
kelvin	K	K
day	d	d
hour	h	h
minute	min	min
second	s	s
metre	m	m
decimetre	dm	dm
centimetre	cm	cm
millimitre	mm	mm
inch	in	in
litre	l	ℓ
decilitre	dl	dl
centilitre	cl	cl
millilitre	ml	ml
calorie	cal	cal
kilocalorie	kcal	kcal
joule	J	J
kilojoule	kJ	kJ
electron volt	eV	eV

Table 2: A (not only) spoonful of (more or less) country and language dependent units. Please note that sometimes a translation is nearly impossible as a unit (e.g. "saltspoonful") may not exist in another language (like german; at least I never heard of it). So please only use units known to you.

unit name	unit-key	symbol
pinch	pn	pinch
tablespoon	EL	tsp.
teaspoon	TL	tbsp.
dessertspoonful	dsp	dsp.
coffeespoonful	csp	csp.
saltspoonful	ssp	ssp.
Messerspitze	Msp	Msp.

Table 3: List of nonsense units (exist just for fun, there will be no support for those units).

unit-key	symbol
eVc-2	$^{eV}/_{c^2}$
hbareV-1	$^{\hbar}/_{eV}$
chbareV-1	$^{c\hbar}/_{eV}$
(chbareV-1)3	$^{c^3\hbar^3}/_{eV^3}$

6 Defining options

Options (to change units) can be newly defined or added to already existing keys using

- \cudefinekeys
- \cudefinesinglekey
- \cuaddkeys
- \cuaddsinglekeys
- \cuaddtokeys

\cudefinekeys
\cudefinesinglekey

```
\cudefinekeys{⟨unit-key-1⟩}
  {
    {⟨unit-key-2⟩} {⟨1 unit-key-1 are ... unit-key-2⟩}
    {⟨unit-key-3⟩} {⟨1 unit-key-1 are ... unit-key-3⟩}
    {⟨unit-key-4⟩} {⟨1 unit-key-1 are ... unit-key-4⟩}
    ...
  }
\cudefinesinglekey{⟨unit-key-1⟩}
  {
    {⟨unit-key-2⟩} {⟨1 unit-key-2 are ... unit-key-1⟩}
    {⟨unit-key-3⟩} {⟨1 unit-key-3 are ... unit-key-1⟩}
    ...
  }
```

If you define new units (see 5 on page 4) and cannot add them to already existing keys you can use \cudefinekeys bzw. \cudefinesinglekey to define new keys.

\cudefinekeys takes the {⟨*unit-key-1*⟩} as a "basis", defines a key with the name ⟨*unit-key-1*⟩ and adds the values ⟨*unit-key-1*⟩, ⟨*unit-key-2*⟩, ⟨*unit-key-3*⟩, etc. Furthermore this command also defines the keys ⟨*unit-key-2*⟩, ⟨*unit-key-3*⟩, etc. with the same values as ⟨*unit-key-1*⟩. Please note that ⟨...⟩ has to be a number.

Sometimes it is not that easy and the conversion of one unit into another needs are more complicated formula (see for example temperatures). If that is the case use \cudefinesinglekey. As the name says it defines *only* the key ⟨*unit-key-1*⟩ with the values ⟨*unit-key-1*⟩, ⟨*unit-key-2*⟩, etc. The advantage of this command is that now ⟨...⟩ can be a formula and the numerical input can be placed explicitly using `#1`.

Example: This example defines following keys with their respective value:

- the key `kg` with the values `kg`, `dag`, `g` and `oz`
- the key `dag` with the values `kg`, `dag`, `g` and `oz`
- the key `g` with the values `kg`, `dag`, `g` and `oz`
- the key `oz` with the values `kg`, `dag`, `g` and `oz`
- the key `d` with the values `d`, `h`, `min` and `s`

- ...

$1\,\mathrm{kg} = 1\,\mathrm{kg}$ $1\,\mathrm{kg} = 100\,\mathrm{dag}$ $1\,\mathrm{kg} = 1000\,\mathrm{g}$
$1\,\mathrm{kg} = 35.273\,99\,\mathrm{oz}$ $1\,\mathrm{kg} = 2.204\,622\,6\,\mathrm{lb}$

```
\cudefinekeys {kg}
  {
    {dag}{ 100 } %% 1 kg are 100 dag
    {g}  { 1000 } %% 1 kg are 1000 g
    {oz} { 35.27399 }  %% 1 kg are 35.27399 oz
    {lb} { 2.204 622 6 } %% 1 kg are  2.204 622 6 lb
  }
\cudefinekeys {d}
  {
    {h}  { 24 } %% 1 day are 24 hours
    {min}{ 1440 } %% 1 day are 1440 minutes
    {s}  { 86400 } %% 1 day are 86400 seconds
  }
```

To convert degree Fahrenheit to degree Celsius, kelvin and degree Réamur one needs the formulas

$$T_C = (T_F - 32) \cdot \frac{5}{9}$$
$$T_K = (T_F - 459.67) \cdot \frac{5}{9}$$
$$T_{Re} = (T_F - 32) \cdot \frac{4}{9}$$

with T_F being the input temperature in degree Fahrenheit and T_C being the same temperature in degree Celsius, etc. Using `\cudefinesinglekey` the key `F` and the values `C`, `K` and `Re` are defined:

```
\cudefinesinglekey {F}
  {
    {C}  { ( #1 - 32 ) *  5/9 } %% see formulas above
    {K}  { ( #1 + 459.67 ) *  5/9 }
    {Re} { ( #1 - 32 ) * 4/9 }
  }
```

This defines the key `F` with the values `F`, `C`, `K` and `Re`.

\cuaddkeys
\cuaddsinglekeys

```
\cuaddkeys{⟨unit-key-1⟩}
  {
    {⟨unit-key-2⟩} {⟨1 unit-key-1 are ... unit-key-2⟩}
    {⟨unit-key-3⟩} {⟨1 unit-key-1 are ... unit-key-3⟩}
    {⟨unit-key-4⟩} {⟨1 unit-key-1 are ... unit-key-4⟩}
    ...
  }
\cuaddsinglekeys{⟨unit-key-1⟩}
  {
    {⟨unit-key-2⟩} {⟨1 unit-key-2 are ... unit-key-1⟩}
    {⟨unit-key-3⟩} {⟨1 unit-key-3 are ... unit-key-1⟩}
    ...
  }
```

These commands add ⟨*unit-key-2*⟩, etc. to the already defined key ⟨*unit-key-1*⟩.

`\cuaddkeys` takes the already defined key `{`⟨*unit-key-1*⟩`}` as a "basis", and adds ⟨*unit-key-2*⟩, ⟨*unit-key-3*⟩, etc. to its values. Furthermore it adds those new values to other keys linked to ⟨*unit-key-1*⟩ and defines the new keys ⟨*unit-key-2*⟩, etc. with the same values as ⟨*unit-key-1*⟩.

If the conversion is more complicated use `\cuaddsinglekeys`. It adds ⟨*unit-key-2*⟩, etc. as values to ⟨*unit-key-1*⟩. The numerical input can be placed using `#1` (see `\cudefinesinglekey`). This command neither defines new keys nor does it add values to other keys than ⟨*unit-key-1*⟩.

Example: Suppose you are british (I am sorry, I can't think of another reason to use those units) and you want to implement 'stone' (yes, I was surprised myself that such a unit exists, but it even appears in a Sherlock-Holmes story). You exactly know that 1 st equals 14 lb, well ... now you have two choices. `\cuaddkeys` or `\cuaddtokeys` (use the one best fitting). This example uses the first, the next the latter one.

```
\newcookingunit {st} %% defining new unit 'stone'
\cuaddkeys {lb}  %% adding st to lb (could also add to kg, dag and oz)
  {
    {st} { 1/14 }  %% 1 lb are 1/14 st as 14 lb are 1 st
  }
```

0.07 st	`\cunum[lb=st]{1}{lb}\\`
14 lb	`\cunum[st=lb]{1}{st}\\`
6350.29 g	`\cunum[st=g]{1}{st}\\`
6.35 kg	`\cunum[st=kg]{1}{st}\\`
0.16 st	`\cunum[kg=st]{1}{kg}\\`

Example: Now you want to add degree Rømer and convert Celsius to degree Rømer:

$$T_{R\emptyset} = T_C * \frac{21}{40} + 7.5$$

```
%% defining new unit 'degree R{\o}mer'
\newcookingunit [\ensuremath{ {} ^ { \circ } }\kern-\scriptspace R{\o}] {Ro}
\cuaddsinglekeys {C} %% adds value 'Ro' to 'C'.
  {
    {Ro} { #1 * 21/40 + 7.5 }
  }
\cusetup %% round to integer automatically
  {
    set-option-for-Ro = { round-to-int = true }
  }
```

10 °C	`\cunum{10}{C}\\`
13 °Rø	`\cunum[C=Ro]{10}{C}\\`

`\cuaddtokeys` `\cuaddtokeys {⟨unit-key-1⟩} {⟨unit-key-2⟩} {⟨1 unit-key-2 are ... unit-key-1⟩}`

Works similar to `\cuaddkeys` regarding the definition of keys.

Example: Continuing the example from before, this time with `\cuaddtokeys`:

```
\newcookingunit {st} %% defining (again) new unit 'stone'
\cuaddtokeys {lb} {st} { 14 }  %% 1 st are 14 lb
```

0.07 st	`\cunum[lb=st]{1}{lb}\\`
14 lb	`\cunum[st=lb]{1}{st}\\`
6350.29 g	`\cunum[st=g]{1}{st}\\`
6.35 kg	`\cunum[st=kg]{1}{st}\\`
0.16 st	`\cunum[kg=st]{1}{kg}\\`

7 Language support

The unit names and symbols depend on the language. To change the name depending on the language you can use `\cudefinename` and to only change symbols use `\cudefinesymbol`.

`decimal-mark`
`one(m)`
`one(f)`
`one(n)`

Those are special keys (as they cannot be used as units). Not only are printed units language depending, but as is the decimal mark (“.” or “,”). To set the decimal mark use `decimal-mark` (see examples below).

Furthermore if you are using the package-option `use-numerals` you may also use the keys `one(m)`, `one(f)` and `one(n)`. If you use this option, integers below a certain value (see option `use-numerals-below`) are written-out. The only problem is the written-out “1” mostly depends on the gender of the following word (e.g. “ein Baum” (m), “eine Pflanze” (f) and “ein Auto” (n)). To set the written-out 1 to be correct with the gender of the used unit, use this key (see also examples below)

\cudefinename

```
\cudefinename{⟨Language⟩}
  {
    {⟨unit-key-1⟩} [⟨symbol-1⟩] {⟨singular-1⟩} [⟨plural-1⟩] ⟨gender⟩
    {⟨unit-key-2⟩} [⟨symbol-2⟩] {⟨singular-2⟩} [⟨plural-2⟩] ⟨gender⟩
    ...
  }
```

This command defines the names (and optionally the symbol) of the commands printed in \cutext and \Cutext (and \cunum regarding the symbol) for the specific ⟨*Language*⟩. For details regarding ⟨*language*⟩ see the translator-documentation.

If the plural form of the name differs from the singular form use [⟨*plural*⟩] to specify the plural form, if no [⟨*plural*⟩] is given the plural will be set equal to its singular. The singular is only used if the number in \cutext and \Cutext is equal to 1.

⟨*gender*⟩ can be m (maskulin), f (feminin) or n (neutrum). If not given m is used as default.

```
\cudefinename {English}
  {
    {kg}  {kilogramme}
    {oz}  {ounce}
    {h}   {hour} [hours]
    {C}   {degree \space Celsius}  [degrees \space Celsius]
    {decimal-marker} {.}
    {one(m)} {one}
    {one(f)} {one}
    {one(n)} {one}
 }

\cudefinename {German}
  {
    {kg}  {Kilogramm} <n>
    {oz}  {Unze} <f>
    {d}   {Tag} [Tage]
    {h}   {Stunde} [Stunden] <f>
    {C}   {Grad\space Celsius}
    {decimal-marker} {,}
    {one(m)} {ein}
    {one(f)} {eine}
    {one(n)} {ein}
  }
```

\cudefinesymbol

```
\cudefinesymbol{⟨Language⟩}
  {
    {⟨unit-key-1⟩} {⟨symbol-1⟩}
    {⟨unit-key-2⟩} {⟨symbol-2⟩}
    ...
  }
```

This command defines the symbols of the units printed in \cunum for the specific ⟨*language*⟩. It works similar as \cudefinename, but only the symbols (and no names) can be set. For details regarding ⟨*language*⟩ see the translator-documentation.

```
\cudefinesymbol {English}
  {
    {decimal-mark} {.}
    {one(m)} {one}
    {one(f)} {one}
    {one(n)} {one}
  }
\cudefinesymbol {German}
  {
    {decimal-mark} {,}
    {one(m)} {ein}
    {one(f)} {eine}
    {one(n)} {ein}
  }
\cudefinesymbol {French}
  {
    {l} {L}
    {dl} {dL}
    {cl} {cL}
    {ml} {mL}
    {decimal-mark} {.}
    {one(m)} {un}
    {one(f)} {une}
    {one(n)} {un}
  }
```

8 Options

Options in cooking-units can mostly be set globally using \cusetup or locally using the optional argument of the respective command (but *not* as a package option). The only exception is the option given in 8.1 on the next page which needs to be used as a package option.

8.1 Load time options

use-numerals

`\usepackage[use-numerals=⟨true/false⟩]{cooking-units}`

If set to `true` loads package fmtcount and uses `\numberstringnum` for `\cutext` and `\Numberstringnum` for `\Cutext` to write-out numbers below `use-numerals-below` (13 by default), integers above are printed as numbers. Please note the keys `one(m)`, `one(f)` and `one(n)` to change the printed "one" (as "one" is in many languages dependent on the gender of the following word. E.g in German: Masculine: ein Baum, Feminin: eine Pflanze, Neutrum: ein Auto).

one kilogramme	`\cutext{1}{kg}\\`
One kilogramme	`\Cutext{1}{kg}\\`
two kilogramme	`\cutext{2}{kg}\\`
Two kilogramme	`\Cutext{2}{kg}\\`
13 kilogramme	`\cutext{13}{kg}\\`
14 kilogramme	`\Cutext{14}{kg}`

8.2 Normal options

This option can only be set as local options or using `\cusetup`, but *not* as load time options.

\cusetup

Options can be set globally using `\cusetup`.

unit

`⟨unit-key-1⟩ = ⟨unit-key-2⟩`

Convert units from ⟨*unit-key-1*⟩ to ⟨*unit-key-2*⟩ (see 6 on page 7 to define new options).

set-option-for-<unit-key>
add-option-for-<unit-key>
erase-all-options

`set-option-for-⟨unit-key⟩ = ⟨key1=value1,...⟩`
`add-option-for-⟨unit-key⟩ = ⟨key1=value1,...⟩`
`erase-all-options`

Sets and adds ⟨*key1=value1,...*⟩, for a specific ⟨*unit-key*⟩ `erase-all-options` is used to erase all options for all ⟨*unit-key*⟩s.

You may want to attach some options to a special ⟨*unit-key*⟩. Those options are automatically activated if (and only if) the specific ⟨*unit-key*⟩ is used (or changed into this unit). Setting options overwrites old options. Adding options, well ... adds the options to the old ones.

The following rounds the values to integers for `F`, `C`, `K` and `Re`.

```
\cusetup
  {
    set-option-for-F  = { round-to-int = true } ,
    set-option-for-C  = { round-to-int = true } ,
    set-option-for-K  = { round-to-int = true } ,
    set-option-for-Re = { round-to-int = true }
  }
```

You can “delete” the options by setting an empty value for a specific ⟨*unit-key*⟩ (or use `erase-all-options` to erase all options for all ⟨*unit-key*⟩s)

eval-fraction

`eval-fraction` = ⟨*true/false*⟩

This option takes `true` or `false` as values. If set to `true` fractions are evaluated. Please note that divisions through zero are not allowed.

```
                                \cusetup{eval-fraction=true}
0.33 kg                         \cunum{1/3}{kg}\\
0.5 kg                          \cunum{1/2}{kg}\\
500 g                           \cunum[kg=g]{1/2}{kg}\\
1.5 kg                          \cunum{1_1/2}{kg}\\
1500 g                          \cunum[kg=g]{1_1/2}{kg}
```

round-precision

`round-precision` = ⟨*integer*⟩

Rounds the amount automatically to ⟨*integer*⟩ digits after the colon. Note that units like `C`, `F`, `K` and `Re` are still rounded to integers due to `set-option-for-`⟨*unit-key*⟩.

```
                                \cusetup{round-precision=5}
1.23457 kg                      \cunum{1.23456789}{kg}\\
0.01259 kg                      \cunum[g=kg]{12.587}{g}\\
194 kg                          \cunum{194}{kg}\\
392–410 °F                      \cunum[C=F]{200--210}{C}\\
−273 °C                         \cunum[K=C]{0.0012}{K}\\
                                \cusetup{round-precision=1}
1.2 kg                          \cunum{1.23456789}{kg}\\
12.6 kg                         \cunum{12.58}{kg}\\
0.2 kg                          \cunum[g=kg]{194}{g}\\
392–410 °F                      \cunum[C=F]{200--210}{C}\\
−273 °C                         \cunum[K=C]{0.0012}{K}
```

round-to-int

`round-to-int` = ⟨*true/false*⟩

Rounds the amount to an integer if set `true`.

```
                                \cusetup{round-to-int=true}
1 kg                            \cunum{1.23456789}{kg}\\
13 kg                           \cunum{12.58}{kg}\\
0–0 kg                          \cunum[g=kg]{194--294}{g}\\
1235 g                          \cunum[kg=g]{1.23456789}{kg}\\
```

`range-sign` `range-sign = ⟨string⟩`
`cunum-range-sign = ⟨string⟩`
`cutext-range-sign = ⟨string⟩`

The second sets the *printed* range-sign used in \cunum (and \cuam) to ⟨*string*⟩, the third sets the printed range-sign used in \cutext/\Cutext to ⟨*string*⟩.

Use the `range-sign` to set the printed range-signs for both \cunum (and \cuam) and \cutext/\Cutext to ⟨*string*⟩.

The default for ⟨*string*⟩ is `--` (for both).

```
                                \cusetup{cunum-range-sign={~to~}}
1 to 2 kg                       \cunum{1--2}{kg}\\
1 to 2kg                        \cuam{1--2}{kg}\\
1–2 kilogramme                  \cutext{1--2}{kg}\\
1–2 kilogramme                  \Cutext{1--2}{kg}\\

                                \cusetup{cutext-range-sign={~to~}}
1–2 kg                          \cunum{1--2}{kg}\\
1–2kg                           \cuam{1--2}{kg}\\
1 to 2 kilogramme               \cutext{1--2}{kg}\\
1 to 2 kilogramme               \Cutext{1--2}{kg}\\

                                \cusetup{range-sign={~to~}}
1 to 2 kg                       \cunum{1--2}{kg}\\
1 to 2kg                        \cuam{1--2}{kg}\\
1 to 2 kilogramme               \cutext{1--2}{kg}\\
1 to 2 kilogramme               \Cutext{1--2}{kg}\\
```

`fraction-command` `fraction-command = ⟨\command⟩`

Sets the command used for printing fractions equal to ⟨\command⟩. ⟨\command⟩ has to take two arguments. By default it is equal to \sfrac from xfrac.

Please note that the amount is *not* printed inside a math environment by default.

```
                                \newcommand\myfrac[2]{#1/#2}
                                \cusetup{fraction-command=\myfrac}
1/8                             \cuam{1/8}\\
1/2 kg                          \cunum{1/2}{kg}\\
4/5 °C                          \cunum{4/5}{C}\\
1 2/3 kg                        \cunum{1_2/3}{kg}\\
                                \cusetup{fraction-command=\nicefrac}
1/8                             \cuam{1/8}\\
1/2 kg                          \cunum{1/2}{kg}\\
4/5 °C                          \cunum{4/5}{C}\\
1 2/3 kg                        \cunum{1_2/3}{kg}
```

`fraction-inline` `fraction-inline = ⟨input containing #1 and #2⟩`

Similar to `fraction-command` only that you don't have to define a command to alter the output of the fraction.

	`\cusetup{fraction-inline={#1/#2}}`
1/8	`\cuam{1/8}\\`
1/2 kg	`\cunum{1/2}{kg}\\`
4/5 °C	`\cunum{4/5}{C}\\`
1 2/3 kg	`\cunum{1_2/3}{kg}\\`
	`\cusetup{fraction-inline={\nicefrac{#2}{#1}}}`
8/1	`\cuam{1/8}\\`
2/1 kg	`\cunum{1/2}{kg}\\`
5/4 °C	`\cunum{4/5}{C}\\`
1 3/2 kg	`\cunum{1_2/3}{kg}`

`mixed-fraction-space` `mixed-fraction-space = ⟨length⟩`

Sets the length between the fraction and the number in a mixed-fraction, default is `0.1em`.

1⅔kg	`\cuam{1_2/3}{kg}\\`
1⅔ kg	`\cunum{1_2/3}{kg}\\`
10⅔ kg	`\cunum{10_2/3}{kg}\\`
	`\cusetup{mixed-fraction-space=1em}`
1 ⅔kg	`\cuam{1_2/3}{kg}\\`
1 ⅔ kg	`\cunum{1_2/3}{kg}\\`
10 ⅔ kg	`\cunum{10_2/3}{kg}\\`
	`\cusetup{mixed-fraction-space=0em}`
1⅔kg	`\cuam{1_2/3}{kg}\\`
1⅔ kg	`\cunum{1_2/3}{kg}\\`
10⅔ kg	`\cunum{10_2/3}{kg}\\`

`set-special-sign` `set-special-sign = ⟨character(s)⟩`
`add-special-sign` `add-special-sign = ⟨character(s)⟩`

Allows ⟨*character(s)*⟩ to be used in the first mandatory argument of `\cunum` without raising an error (you can customize this behavior, see `set-unknown-message`). By default it is set to `?`.

? kg	`\cunum{?}{kg}\\`
10?–20? kg	`\cunum[g=kg]{10?--20?}{kg}\\`
	`\cusetup{add-special-sign={xX}}`
x kg	`\cunum{x}{kg}\\`
X–? kg	`\cunum{X--?}{kg}\\`
	`\cusetup{set-special-sign={}}`
1 kg	`\cunum{1}{kg}\\`
1–2 kg	`\cunum{1--2}{kg}\\`

parse-number `parse-number =` ⟨*true/false*⟩

If set to `false` prints the number of `\cunum`, `\cutext` and `\Cutext` as they are (after some ... well ... parsing due to "_"). It is `true` by default.

```
                                \cusetup{parse-number=false}
1 kg                            \cunum[kg=g]{1}{kg}\\
1–2 kg                          \cunum{1--2}{kg}\\
1————-2 kg                      \cunum{1----------2}{kg}\\
1.2 kg                          \cunum{1.2}{kg}\\
1,2 kg                          \cunum[kg=g]{1,2}{kg}\\
1/2 kg                          \cunum{1/2}{kg}\\
1_2/3 kg                        \cunum{1_2/3}{kg}\\
1/2_3 kg                        \cunum{1/2_3}{kg}\\
qwertzuiop kg                   \cunum{qwertzuiop}{kg}\\
1 kilogramme                    \cutext{1}{kg}\\
100 kilogramme                  \cutext{100}{kg}\\
gjfak kilogramme                \cutext{gjfak}{kg}\\
```

use-numerals-below `use-numerals-below =` ⟨*integer*⟩

Only usable if the package option `use-numerals` is active. Prints the name of the numbers for integers used in `\cutext` and `\Cutext` smaller than ⟨*integer*⟩. ⟨*integer*⟩ it is by default 13.

```
one kilogramme                  \cutext{1}{kg}\\
two kilogramme                  \cutext{2}{kg}\\
twelve kilogramme               \cutext{12}{kg}\\
13 kilogramme                   \cutext{13}{kg}\\
                                \cusetup{use-numerals-below=10}
one kilogramme                  \cutext{1}{kg}\\
two kilogramme                  \cutext{2}{kg}\\
12 kilogramme                   \cutext{12}{kg}\\
13 kilogramme                   \cutext{13}{kg}\\
                                \cusetup{use-numerals-below=0}
1 kilogramme                    \cutext{1}{kg}\\
2 kilogramme                    \cutext{2}{kg}\\
12 kilogramme                   \cutext{12}{kg}\\
13 kilogramme                   \cutext{13}{kg}\\
```

set-unknown-message `set-unknown-message =` ⟨*error/warning/none*⟩

Using a special sign (`?` by default) causes a warning to be raised. Set this option to `error` if you want an error (as an extra emphasis), `warning` if you want a warning (default) and `none` if you don't want to know anything about it.

8.3 Weird options

check-temperature

`check-temperature = ⟨true/false⟩`

Checks if the used temperature is below the absolute zero point. Currently `C`, `F`, `K` and `Re` are supported. While `\cunum{0}{K}` is ok, `\cunum{-1}{K}` raises an error, same for the others. Is set to `false` by default.

convert-to-eV

`convert-to-eV = ⟨true/false⟩`

Converts (nearly) every unit in table 3 on page 6 to electron volt or the respective derivative. Note that this option is: a) experimental and probably will forever be and b) just a joke, you are not supposed to use this units in a cookery book (and as you see this package doesn't support the arrangement of such huge numbers).

```
                                \cusetup{convert-to-eV=true}
5609589206809303 eV/c^2         \cunum{1}{kg}\\
130148943000000000 c^3ħ^3/eV^3  \cunum{1}{l}\\
6241509125883258000 eV          \cunum{1}{J}\\
5067730.93 cħ/eV                \cunum{1}{m}\\
0.02 eV                         \cunum{1}{C}\\
1519267460691082 ħ/eV           \cunum{1}{s}\\
```

9 Bugs & Feedback

Bug reports are always welcome. If you are sending a bug report please include a minimal working example showing the bug and a short description. Furthermore please add "cooking-units" to the e-mail header. GMX has the habit of putting e-mails into the spam account and adding "cooking-units" to the header makes it easier to recognize those e-mails.

Feedback and requests (commands, units) are most welcome. Please also add (if possible) an example of the desired output into the minimal example (and also add "cooking-units" to the header).

Furthermore, as you can see I am not able to speak too many languages (german and english to be precise; I managed to add french with the help of the internet, which is not optimal) so if you are able to speak a language not yet implemented and would like to help you can send me a list of the translations of the units given in 4 on page 4. I would need

- their singular (and plural) form,
- the gender,
- the printed symbol (if different),
- `decimal-mark` and `one(m)`, `one(f)`, `one(n)`

Oh yeah, if someone has a better idea of how to deal with languages I am happy to know.