The cooking-units package*

Ben Vitecek
b.vitecek@gmx.at

2020/01/13

Abstract

This package enables user to globally format units, to switch between them and change your recipes to a given number of persons.
For not implemented units or differences between Imperial and U.S. unit you may have a look at appendix B.
It should be used for light-hearted things like cookery books (and not e.g. scientific texts; use e.g. siunitx for those).
Please read through the section “Important Changes”

Contents

[*] This document corresponds to Benedikt Vitecek v1.46, dated 2020/01/13.

1 Introduction

While writing on a cookery book I used – for some 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 LaTeX-like, 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.

1.1 Supported languages

- German
- English
- French (currently suboptimal[1])

Want to contribute a new language or make a correction to an existing one? See section 9 for more details. Wanna just check the existing translations? See appendix A.

2 The Commands

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

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

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`*⟩[2]. 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 check-up[3]. `[`⟨*space*⟩`]` adds a space between the number and the unit using `\phantom`.

For a list of predefined units have a look at table 1.

⟨*label*⟩ is explained in section 3.

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}`

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

[2] New keys can be added and defined, see section 4 and section 5 for further information.

[3] You can customize this behavior, see section 8

Decimal numbers are automatically rounded to 2 digits after the colon, temperatures (`C`, `F`, `K` and `Re`) are automatically rounded to integers.[4]

`\cutext` and `\Cutext` print the number and the written name of the unit. Since v1.10 it works similar[5] to `\cunum`: it allows the conversion between units and interprets the numerical part (again `_` and `/` are used for (mixed) fractions and `--` for ranges). Furthermore, `\cutext` and `\Cutext` allow the usages of numerals (see section 8.1 for more information).

1 litre	`\cutext{1}{l}\\`
1 litre	`\Cutext{1}{l}\\`
1 to 2 litres	`\Cutext{1--2}{l}\\`
12 litres	`\cutext{12}{l}\\`
13 litres	`\Cutext{13}{l}`

and using (e.g.) package option `use-fmtcount-numerals=true`

one litre	`\cutext{1}{l}\\`
One litre	`\Cutext{1}{l}\\`
one to two litres	`\cutext{1--2}{l}\\`
One to two litres	`\Cutext{1--2}{l}\\`
twelve litres	`\cutext{12}{l}\\`
13 litres	`\Cutext{13}{l}`

You can customize the numeral functions used with `numeral-function` and `Numeral-function`.

Furthermore, since v1.10 `\cutext` and `\Cutext` also allow their units to be changed (this behavior can be altered using `cutext-change-unit`):

	`\cusetup{l=ml}`
1000 millilitres	`\cutext{1}{l}\\`
1000 millilitres	`\Cutext{1}{l}\\`
1000 to 2000 millilitres	`\cutext{1--2}{l}\\`
12000 millilitres	`\cutext{12}{l}\\`
13000 millilitres	`\Cutext{13}{l}\\`
? litres	`\Cutext{?}{l}\\`
½ litres	`\Cutext{1/2}{l}\\`

`\cuam` works like `\cunum`, but without a unit, so changing units doesn't affect it. Like `\cunum` `_` and `/` are used to imply a (mixed) fraction and `--` is used for ranges.

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

Furthermore it allows the concept of "phrases" (replacing a positive integer by a word; such as "12" becoming "dozen"[6]) which can be activated by the option `use-phrases` (as I don't know any english phrases, I switched the language to german for the following examples)

[4] You can – of course – change this behavior, see section 8.

[5] One could also say "exactly like".

[6] At least I think

11
1 Dutzend
13
2 Dutzend
1–2 Dutzend
12–13
18
5 Dutzend

```
\cusetup{use-phrases=true}
\cuam{11}\\
\cuam{12}\\
\cuam{13}\\
\cuam{24}\\
\cuam{12--24}\\
\cuam{12--13}\\
\cuam{18}\\
\cuam{60}
```

3 Label & refs: Changing the amount of the recipe

What if you don't want to change units, but the amounts of the recipe because you cook not for 4 persons, but for 2 and don't like to do the math? Simple, use the following commands:

- `\culabel`{⟨*label*⟩}{⟨*number of persons*⟩}
- `\curef`{⟨*label*⟩}

The first one is the important one: It defines a ⟨*label*⟩ for a recipe which is initially for ⟨*number of persons*⟩. Afterwards ⟨*label*⟩ can be used to tell the commands from section 2 that the given amounts are for ⟨*number of persons*⟩. Each ⟨*label*⟩ must be unique and an error is raised if a ⟨*label*⟩ is already defined.

If you would like to print the number of persons this recipe is for, use `\curef`, which is fully expandable.

The following example uses `\culabel` to specify that the recipe is initially intended for 2 persons:

recipe for 2 persons:
10–20 dag flour,
½ ℓ water,
10 gramme nuts,
2–3 eggs,
180 °C (356 °F) open fire

```
\culabel{recipe}{2}
recipe for \curef{recipe} persons:\\
\cunum<recipe>{10--20}{dag} flour,\\
\cunum<recipe>{1/2}{l} water,\\
\cutext[ref=recipe]{10}{g} nuts,\\
\cuam<recipe>{2--3} eggs,\\
\cunum{180}{C} (\cunum[C=F]{180}{C})
open fire
```

In combination with the option `set-number-of-persons` and `recalculate-amount` you can have this recipe changed to four persons:

```
\culabel{recipe}{2}
%% adding options:
\cusetup{set-number-of-persons=4,recalculate-amount=true}
```

recipe for 4 persons:
20–40 dag flour,
1 ℓ water,
20 gramme nuts,
4–6 eggs,
180 °C (356 °F) open fire

```
recipe for \curef{recipe} persons:\\
\cunum<recipe>{10--20}{dag} flour,\\
\cunum<recipe>{1/2}{l} water,\\
\cutext[ref=recipe]{10}{g} nuts,\\
\cuam<recipe>{2--3} eggs,\\
\cunum{180}{C}
(\cunum[C=F]{180}{C}) open fire
```

Note that fractions are automatically evaluated and that only values with a ⟨*label*⟩ are changed (`\cunum{180}{C}` for example stays the same which also makes sense as the heat should be the same).

3.1 Rounding temperatures

By default temperatures are rounded to integers (using `round-precision=0`). Since 1.30 it is possible to round amounts to a negative precision. If you want to round temperatures to the tens see the following example (`set-option-for-`⟨*`unit`*⟩ is described in section 8.2.1).

182 °C	`\cunum{182}{C}\\`
356 °F	`\cunum[C=F]{180}{C}\\`
144 °Ré	`\cunum[C=Re]{180}{C}\\`
453 K	`\cunum[C=K]{180}{C}\\`
	`\cusetoptionfor{C,F,K,Re}{round-precision=-1}`
180 °C	`\cunum{182}{C}\\`
360 °F	`\cunum[C=F]{180}{C}\\`
140 °Ré	`\cunum[C=Re]{180}{C}\\`
450 K	`\cunum[C=K]{180}{C}\\`

4 Predefined units & some notes

In table 1 and you can find all predefined units which can be transformed into each other (sorted by group). Other predefined units (which cannot be used for transformation) are shown in table 2. Table 3 pretty much exists just for fun.

Table 1: This table shows all units which can be transformed into each other, sorted by group. The columns "default" show the abbreviations used if for given language no translation is defined. The translations used for `\cutext` and `\Cutext` are shown in appendix A. Note that "electron volt" exists just for fun.

description	key	default	description	key	default
kilogramme	`kg`	kg	metre	`m`	m
decagramme	`dag`	dag	decimetre	`dm`	dm
gramme	`g`	g	centimetre	`cm`	cm
ounce	`oz`	oz	millimitre	`mm`	mm
pound	`lb`	lb	inch	`in`	in
stick (of butter)	`stick`	stick			
day	`d`	d	litre	`l`	l
hour	`h`	h	decilitre	`dl`	dl
minute	`min`	min	centilitre	`cl`	cl
second	`s`	s	millilitre	`ml`	ml
calorie	`cal`	cal	degree Celsius	`C`	°C
kilocalorie	`kcal`	kcal	degree Fahrenheit	`F`	°F
joule	`J`	J	degree Réaumur	`Re`	°Ré
kilojoule	`kJ`	kJ	kelvin	`K`	K
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.

description	key	symbol
pinch	`pn`	pinch
tablespoon	`EL`	EL
teaspoon	`TL`	TL
dessertspoonful	`dsp`	dsp.
coffeespoonful	`csp`	csp.
saltspoonful	`ssp`	ssp.
Messerspitze (point of a knife)	`Msp`	Msp.

Table 3: List of (not really) nonsense units (exist just for fun, there will be no support for those units; unless – of course – you really want it).

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

5 Defining units

New units can be defined using

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

\declarecookingunit
\newcookingunit
\providecookingunit

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

These commands define the unit ⟨*unit-key*⟩. If the key is not the same as the printed symbol use `[`⟨*symbol*⟩`]`. Note that ⟨*unit-key*⟩ can neither contain `/` nor `,`.

Please note due to the current implementation catcodes may cause trouble. For example using `:` inside ⟨*unit-key*⟩ may cause the key to “not be defined” in the document. If that happens try removing or changing the ⟨*unit-key*⟩.

`\newcookingunit` raises an error if the unit is already defined, `\declarecookingunit` creates or (if given) overwrites ⟨*symbol*⟩ and `\providecookingunit` does nothing if the unit is already defined.

All units have male gender `m` by default.

Some examples:

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

Note: The definition of the printed degree Celsius is copied and pasted from (a maybe older version of) siunitx.

\declarecookingderivatives

```
\declarecookingderivatives {⟨unit-list⟩} {⟨unit-key⟩}
   {⟨mathematical-relation⟩} {⟨unit-symbol⟩}
```

This function is experimental. Defines new units which are a combination of the units given in ⟨*unit-list*⟩ *and* their linked-units. ⟨*unit-key*⟩, ⟨*mathematical-relation*⟩ and ⟨*unit-symbol*⟩ accept `#1` to `#n` as arguments with n being the number of units given in ⟨*unit-list*⟩. n *cannot* be greater than 8 (and it will probably compile for quite a while). Also note that this command doesn't work/isn't tested for single keys.

Also note that it is quite possible that an “overflow-error” will occur if there are too many units.

Note: Due to catcodes (and my inability to deal with them properly) there can be problems when using `:` (and probably other signs) inside ⟨*unit-key*⟩.

Example: Your homework is to change the unit of energy $\mathrm{kg\,m^2\,s^{-2}}$ into $\mathrm{oz\,in^2\,min^{-2}}$. To check if you are correct you use `\declarecookingderivatives`:

```
\declarecookingderivatives{kg,m,s}{#1*#2-#3}
{ (#1)*(#2)^2/(#3)^2 } {\sfrac{#1\,#2${}^2$}{#3${}^2$}}
```

Using `\cunum[kg*m-s=oz*in-min]{1}{kg*m-s}` shows that $1\,\mathrm{kg\,m^2/s^2}$ is equal to $196829101.34\,\mathrm{oz\,in^2/min^2}$. (Note: This is *really* experimental and it can easily happen that an “overflow-error” occurs) (like for me which is why I hard-coded the results after testing).

6 Defining options to change units

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

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

I apologize for the (name) inconsistency between `\cudefinekeys` and `\cudefinesinglekey` (although they are named similarly, they work differently).

`\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 section 5) and cannot add them to already existing keys you can use `\cudefinekeys` or `\cudefinesinglekey` respectively to define new keys.

`\cudefinekeys` takes `{`⟨*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`

- …

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

```
\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[7]

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

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` with values `C`, `K` and `Re` is 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`.

[7] See Wikipedia.

\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}\\`
101.6 kg	`\cunum[st=kg]{16}{st}`

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

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

```
%% 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-precision = 0 }
  }
```

```
   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}\\
   101.6 kg                           \cunum[st=kg]{16}{st}
```

7 Language support

Unit names and symbols depend on the language. To change the name and symbol for given language you can use \cudefinename; to only change symbols use \cudefinesymbol.

decimal-mark
cutext-range-sign
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 ,) and the text which substitutes the range-sign. To set the decimal mark use decimal-mark (see examples below), to set the range-sign for \cutext and \Cutext use cutext-range-sign.

Note that cutext-range-sign is "overwritten" by the *option* cutext-range-sign. If the *option* is set, then the language symbol will be ignored.

Furthermore if you are using numerals you may also use the keys one(m), one(f) and one(n). 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 word following (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 these keys (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 units printed in `\cutext` and `\Cutext` (and `\cunum` regarding the symbol) for the specific ⟨*Language*⟩. For details regarding ⟨*language*⟩ see the translations documentation.

If the plural form of the name differs from the singular form use `[`⟨*plural*⟩`]` to specify the plural form, else it will be equal to its singular form. The singular form 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} {.}
    {cutext-range-sign} {~to~}
    {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} {,}
    {cutext-range-sign} {~bis~}
    {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 translations documentation.

```
\cudefinesymbol {English}
  {
    {decimal-mark} {.}
    {cutext-range-sign} {~to~}
    {one(m)} {one}
    {one(f)} {one}
    {one(n)} {one}
  }
\cudefinesymbol {German}
  {
    {decimal-mark} {,}
    {cutext-range-sign} {~bis~}
    {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}
  }
```

Example: Imagine that instead of the abbreviation “dag” for “decagramme” you want to use “ducks” (because . . . I don’t know). You can easily do this via

```
\cudefinesymbol {English}
  {
    {dag} {ducks}
  }
```

As you can see it may be a bit suboptimal as there is no plural version allowed. You do it anyway and end up with:

12 ducks weed	`\cunum{12}{dag} weed\\`
3 ducks nuts	`\cunum{3}[0]{dag} nuts\\`
10 ducks duckmeat	`\cunum{10}{dag} duckmeat`

7.1 Phrases

Each language has synonyms for certain (integer) numbers. This package supports those phrases and they can be implemented with the following command and used by \cuam:

`\cudefinephrase`

```
\cudefinephrase{⟨Language⟩}
  {
    {⟨integer-1⟩}   {⟨phrase-1⟩} [⟨phrase-1-plural⟩] <⟨gender-1⟩>
    {⟨integer-2⟩} * {⟨phrase-2⟩} [⟨phrase-2-plural⟩] <⟨gender-2⟩>
    ...
  }
```

This command pairs for a given `{`⟨*Language*⟩`}` (see package translations) the number `{`⟨*integer-1*⟩`}` with `{`⟨*phrase-1*⟩`}` (& plural and gender). The package then checks if the amount given in `\cuam` is either this number or a *multiple* of it.

If the behavior of checking for a multiple is not wanted, you can use the optional star `*` for a given `{`⟨*integer*⟩`}`

⟨*gender*⟩ can be `m`, `f` or `n`. It is `m` by default.

Afterwards the numbers are ordered from highest to lowest so that the phrase with the highest number is used (if used at all).

Furthermore, it chooses star (`*`) phrases over non-star phrases.

Note: Numbers with the optional star `*` are stored as negative numbers.

Example: The following example creates some phrases for the language “German”:

```
\cudefinephrase {German}
  {
    { 12 }  {Dutzend} <n> %% implemented by default
    { 60 }  {Schock} <n>
    { 6  }* {halbes\ Dutzend} <n>
  }
```

Let’s just use them (german language activated!):

1 Dutzend
2 Dutzend
1 Schock
2 Schock
1 halbes Dutzend
18

```
\cusetup{use-phrases=true}
\cuam{12}\\
\cuam{24}\\
\cuam{60}\\
\cuam{120}\\
\cuam{6}\\
\cuam{18}
```

As you can see, “Schock” (60) is preferred over “Dutzend” (12) as it linked to the higher number. Furthermore, for 6 the phrase “halbes Dutzend” (half a dozen) is used, but because it is a star version it is *not* used for 18.

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 section 8.1 which needs to be used as a package option.

`\cusetup`

```
\cusetup{⟨options⟩}
```

Options can be set using `\cusetup{`⟨*options*⟩`}`.

\cusetoptionfor
\cuaddoptionfor
\cuclearoptionfor

```
\cusetoptionfor{⟨unit-list⟩}{⟨options⟩}
\cuaddoptionfor{⟨unit-list⟩}{⟨options⟩}
\cuclearoptionfor{⟨unit-list⟩}
```

cooking-units allows you to attach options to units. Those options are activated if (and only if) the specific unit is used *or* if another unit is converted into it. Those options allow you to e.g. round temperatures to integers automatically. Furthermore, those added options are overwritten by local options.

`\cusetoptionfor` *sets* ⟨*options*⟩ to each unit in ⟨*unit-list*⟩ overwriting the old ones.

`\cuaddoptionfor` *adds* ⟨*options*⟩ to each unit in ⟨*unit-list*⟩.

`\cuclearoptionfor` clears all options given to each unit in ⟨*unit-list*⟩.

Example: Temperatures `C`, `F`, `K` and `Re` are by default rounded to integers.

75 °C	`\cunum{75.23}{C}\\`
75 °F	`\cunum{75.23}{F}\\`
75 K	`\cunum{75.23}{K}\\`
75 °Ré	`\cunum{75.23}{Re}\\`
	`\cusetoptionfor{C,F,K,Re}{round-precision=-1}`
80 °C	`\cunum{75.23}{C}\\`
80 °F	`\cunum{75.23}{F}\\`
80 K	`\cunum{75.23}{K}\\`
80 °Ré	`\cunum{75.23}{Re}\\`
	`\cuclearoptionfor{C,F,K,Re}`
75.23 °C	`\cunum{75.23}{C}\\`
75.23 °F	`\cunum{75.23}{F}\\`
75.23 K	`\cunum{75.23}{K}\\`
75.23 °Ré	`\cunum{75.23}{Re}`

8.1 Load time options

use-fmtcount-numerals

```
\usepackage[use-fmtcount-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. You can decide to not print any numerals by setting `print-numerals` to `false`.

Note: You don't need to use this function to use numerals. Using `print-numerals` and setting `numeral-function` and `Numeral-function` also works.

one kilogramme	`\cutext{1}{kg}\\`
One kilogramme	`\Cutext{1}{kg}\\`
two kilogramme	`\cutext{2}{kg}\\`
Two kilogramme	`\Cutext{2}{kg}\\`
twelve kilogramme	`\cutext{12}{kg}\\`
Twelve kilogramme	`\Cutext{12}{kg}\\`
13 kilogramme	`\cutext{13}{kg}\\`
13 kilogramme	`\cutext{13}{kg}\\`
14 kilogramme	`\Cutext{14}{kg}`

Note: `use-fmtcount-numerals` is a package option as it needs to load fmtcount which is not loaded by default.

Note: 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).

Note: You can always change the functions used to print numerals with `numeral-function` and `Numeral-function`.

8.2 Normal options

Options in this subsection can only be set as local options or using `\cusetup`, but *not* as load time options.

8.2.1 Unit Specific options

`<unit>`

`⟨unit-key-1⟩ = ⟨unit-key-2⟩`

Change ⟨*unit-key-1*⟩ to ⟨*unit-key-2*⟩ (see section 6 to define new options).

`<group>`

`⟨group⟩ = ⟨unit-key⟩`

Changes each unit contained in ⟨*group*⟩ to ⟨*unit-key*⟩ (⟨*unit-key*⟩ must be part of ⟨*group*⟩).

⟨*group*⟩	default ⟨*unit-key*⟩s
`weight`	`kg`, `dag`, `g`, `oz`, `lb`, `stick`
`length`	`m`, `dm`, `cm`, `mm`, `in`
`volume`	`l`, `dl`, `cl`, `ml`
`temperature`	`C`, `F`, `K`, `Re`
`energy`	`cal`, `kcal`, `J`, `kJ`, `eV`
`time`	`d`, `h`, `min`, `s`

1000 g
10 g
1 g
28.35 g
453.59 g
113.4 g

```
\cusetup{weight=g}
\cunum{1}{kg}\\
\cunum{1}{dag}\\
\cunum{1}{g}\\
\cunum{1}{oz}\\
\cunum{1}{lb}\\
\cunum{1}{stick}\\
```

You can define new groups using `\cudeclareunitgroup`:

`\cudeclareunitgroup`

`\cudeclareunitgroup {⟨group-name⟩} {⟨unit-list⟩}`

Defines the group ⟨*group-name*⟩ containing the list ⟨*unit-list*⟩. This allows the usage of `\meta{group-name}=\meta{unit-key}` to change all units in the group ⟨*group-name*⟩ to ⟨*unit-key*⟩ (which has to be part of ⟨*unit-list*⟩).

Example: Define the group “weight”:

```
\cudeclareunitgroup {weight} { kg , dag, g, oz, lb, stick }
```

Now `\cusetup{weight=dag}` can be used to change all units contained in `weight` to `dag`.

`\cuaddtounitgroup`

`\cuaddtounitgroup{⟨group⟩}{⟨unit-list⟩}`

Adds ⟨*unit-list*⟩ to an already existing ⟨*group*⟩ (both need to exist).

Example: Adding `st` to the group `weight`

1000 g
10 g
1 g
28.35 g
453.59 g
113.4 g
6350.29 g

```
\cuaddtounitgroup{weight}{st}
\cusetup{weight=g}
\cunum{1}{kg}\\
\cunum{1}{dag}\\
\cunum{1}{g}\\
\cunum{1}{oz}\\
\cunum{1}{lb}\\
\cunum{1}{stick}\\
\cunum{1}{st}
```

`add-unit-to-group`

```
add-unit-to-group =
  {
    ⟨group1⟩ = {⟨unit-key-list⟩},
    ⟨group2⟩ = {⟨unit-key-list⟩},
    ...
  }
```

Adds each ⟨*unit-key*⟩ in ⟨*unit-keys-list*⟩ to ⟨*group*⟩. The key-val equivalent of `\cuaddtounitgroup`.

Example: The same as above: This example adds the unit `st` to the group `weight` and `Ro` to `temperature`.

```
\cusetup
  {
    add-unit-to-group = { weight = {st} , temperature = {Ro} }
  }
```

1000 g
10 g
1 g
28.35 g
453.59 g
113.4 g
6350.29 g

```
\cusetup{weight=g}
\cunum{1}{kg}\\
\cunum{1}{dag}\\
\cunum{1}{g}\\
\cunum{1}{oz}\\
\cunum{1}{lb}\\
\cunum{1}{stick}\\
\cunum{1}{st}
```

`set-option-for-<unit-key>`
`add-option-for-<unit-key>`

```
set-option-for-⟨unit-key⟩ = {⟨ key1 = value1, ... ⟩}
add-option-for-⟨unit-key⟩ = {⟨ key1 = value1, ... ⟩}
```

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

The less flexible key-value version of `\cusetoptionfor` and `\cuaddoptionfor`.

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

```
\cusetup
  {
    set-option-for-F  = { round-precision = 0 } ,
    set-option-for-C  = { round-precision = 0 } ,
    set-option-for-K  = { round-precision = 0 } ,
    set-option-for-Re = { round-precision = 0 }
  }
```

although note that it would be easier to simply write

```
\cusetoptionfor {F,C,K,Re} { round-precision = 0 }
```

`set-option-for`
`add-option-for`

```
set-option-for =
  {
    ⟨unit-key1⟩ = {⟨keys=vals⟩},
    ⟨unit-key2⟩ = {⟨keys=vals⟩},
    ...
  }
add-option-for =
  {
    ⟨unit-key1⟩ = {⟨keys=vals⟩},
    ⟨unit-key2⟩ = {⟨keys=vals⟩},
    ...
  }
```

Sets/adds each ⟨*keys=vals*⟩ to the specific ⟨*unit-key*⟩. Works pretty much the same way their `set-option-for-`⟨*unit-key*⟩ and `add-option-for-`⟨*unit-key*⟩ counterparts.

The less flexible versions of the commands `\cusetoptionfor` and `\cuaddoptionfor`.

Example: The following example does the same as the example above:

```
\cusetup
  {
    set-option-for =
      {
        F  = { round-precision = 0 } ,
        C  = { round-precision = 0 } ,
        K  = { round-precision = 0 } ,
        Re = { round-precision = 0 }
      }
  }
```

`erase-all-options`
`erase-all-options-for`

```
erase-all-options
erase-all-options-for = {⟨unit-key1, unit-key2, ...⟩}
```

Erase options added to units. `erase-all-options` erases all options for *all* ⟨*unit-key*⟩s.

`erase-all-options-for` is used to remove added options from the specified ⟨*unit-key*⟩s (key-value version of `\cuclearoptionfor`).

Example: The following code erases all attached options from `C`, `F`, `K` and `Re`:

```
\cusetup{ erase-all-options-for = {C, F, K, Re} }
```

It's the same as

```
\cuclearoptionfor {C, F, K, Re}
```

8.2.2 Command behavior

`cutext-to-cunum`

`cutext-to-cunum` = ⟨*true/false*⟩

Want to get rid of all `\cutext` and `\Cutext`? Set this option to `true` and all `\cutext` and `\Cutext` are changed into `\cunum`.

1 kilogramme	`\cutext{1}{kg}\\`
2 kilogramme	`\Cutext{2}{kg}\\`
½ kilogramme	`\cutext{1/2}{kg}\\`
? kilogramme	`\cutext{?}{kg}\\`
1000 to 2000 gramme	`\cutext[kg=g]{1--2}{kg}\\`
	`\cusetup{cutext-to-cunum=true}`
1 kg	`\cutext{1}{kg}\\`
2 kg	`\Cutext{2}{kg}\\`
½ kg	`\cutext{1/2}{kg}\\`
? kg	`\cutext{?}{kg}\\`
1000–2000 g	`\cutext[kg=g]{1--2}{kg}`

`cutext-change-unit`

`cutext-change-unit` = ⟨*true/false*⟩

Set this option to `false` if you do *not* want the units of `\cutext` and `\Cutext` to be changed. Set to `true` by default

1000 gramme	`\cutext[kg=g]{1}{kg}\\`
½ kilogramme	`\cutext[kg=g]{1/2}{kg}\\`
1000 to 2000 gramme	`\cutext[kg=g]{1--2}{kg}\\`
	`\cusetup{cutext-change-unit=false}`
1 kilogramme	`\cutext[kg=g]{1}{kg}\\`
½ kilogramme	`\cutext[kg=g]{1/2}{kg}\\`
1 to 2 kilogramme	`\cutext[kg=g]{1--2}{kg}`

`cuam-version`
`cutext-version`

`cuam-version` = ⟨*old/new*⟩
`cutext-version` = ⟨*old/new*⟩

Since v1.10 this package also parses and checks the input of `\cutext` and `\Cutext` and `\cuam`. If you want to restore the old behavior, set this option to `old`, but note that then you can neither change the amounts for a given number of persons nor change the unit of `\cutext` and `\Cutext`. Both of them are set to `new` by default.

8.2.3 Hooks

`commands-add-hook`
`cunum-add-hook`
`cutext-add-hook`
`Cutext-add-hook`
`cuam-add-hook`

```
commands-add-hook = {⟨code⟩}
cunum-add-hook  = {⟨code⟩}
cutext-add-hook = {⟨code⟩}
Cutext-add-hook = {⟨code⟩}
cuam-add-hook   = {⟨code⟩}
```

Adds ⟨*code*⟩ to the respective command (or in case of the first key: to *all* commands). The hook is executed *after* setting the keys, but *before* parsing and processing the input.

Please be carful with spaces, they will be printed.

Example: You would like to count how often all commands of this package are used. Simply add:

```
\newcounter{CookingUnitsCounter} %% or however you like it
\cusetup{commands-add-hook={\stepcounter{CookingUnitsCounter}}}
   %% beware of spaces inside the add-hook keys.
```

to your preamble. The following table lists how often each command is used in this documentation (with help of totalcount):

command	times
`\cunum`	217
`\cutext`	66
`\Cutext`	27
`\cuam`	59
total	369

8.2.4 Input and Outputs

`expand-both`
`expand-amount`
`expand-unit`

```
expand-both = ⟨n/o/f/x⟩
expand-amount = ⟨n/o/f/x⟩
expand-unit = ⟨n/o/f/x⟩
```

By default the commands `\cunum`, `\cutext` and `\Cutext` and `\cunum` do *not* expand their input. You can change the expansion behavior of ⟨*amount*⟩ and/or ⟨*unit-key*⟩ using the options specified above. The meaning of the available values are the same as specified in the LaTeX3 document "interface3".

It is set to `n` (no expansion) by default.

`set-special-sign`
`add-special-sign`

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

Allows ⟨*character(s)*⟩ to be used in the first mandatory argument of `\cunum`, `\cuam`, `\cutext` and `\Cutext` without raising an error (you can customize this behavior, see `set-unknown-message`). By default it is set to `?`. Please note that the sign `<` is not allowed as a special sign.

? 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}`

`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.

`set-cutext-translation-message`

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

If a translation for `\cutext` and `\Cutext` is not available the commands are replaced by `\cunum`. Currently – if this is happening – a warning is shown, you may change the behavior of the message (error, warning or not showing at all) using this option.

`print-numerals`

`print-numerals` = ⟨*true/false*⟩

Prints numerals for integers smaller then `use-numerals-below` if set to `true`. If set to `false` no numerals are printed.

If you use the package option `use-fmtcount-numerals` this option is automatically set to `true`.

If you want to use another package, just set this option to `true` and use `numeral-function` and `Numeral-function`.

Example: (Using the package option `use-fmtcount-numerals`:

one kilogramme	`\cutext{1}{kg}\\`
two kilogramme	`\cutext{2}{kg}\\`
twelve kilogramme	`\cutext{12}{kg}\\`
13 kilogramme	`\cutext{13}{kg}\\`
	`\cusetup{print-numerals=false}`
1 kilogramme	`\cutext{1}{kg}\\`
2 kilogramme	`\cutext{2}{kg}\\`
12 kilogramme	`\cutext{12}{kg}\\`
13 kilogramme	`\cutext{13}{kg}\\`

`use-numerals-below`

`use-numerals-below` = ⟨*integer*⟩

If `print-numerals` is `true`, prints the numerals in `\cutext` and `\Cutext` for integers smaller than ⟨*integer*⟩. ⟨*integer*⟩ is by default 13. You can deactivate the printing of numerals by `print-numerals=false`.

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}\\`
	`\cusetup{use-numerals-below=12001}`
one thousand gramme	`\cutext[kg=g]{1}{kg}\\`
two thousand gramme	`\cutext[kg=g]{2}{kg}\\`
twelve thousand gramme	`\cutext[kg=g]{12}{kg}\\`
13000 gramme	`\cutext[kg=g]{13}{kg}\\`

`numeral-function`
`Numeral-function`

`numeral-function = ⟨`*`function`*`⟩`
`Numeral-function = ⟨`*`function`*`⟩`

Sets the functions used for printing numerals. `numeral-function` is used for lowercase, `Numeral-function` for capitalized cases.

Example: Using the commands from fmtcount you can set the numeral function equal to

```
\cusetup{
  numeral-function = \numberstringnum ,
  Numeral-function = \Numberstringnum
}
```

(this happens if you use the package option `use-fmtcount-numerals`)

`parse-number`

`parse-number = ⟨`*`true/false`*`⟩`

If set to `false` prints the number of `\cunum`, `\cutext`, `\Cutext` and `\cuam` as they are (after some . . . well . . . parsing due to "`_`"). Is set to `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}\\`
someweirdstuff kg	`\cunum{someweirdstuff}{kg}\\`
1 kilogramme	`\cutext{1}{kg}\\`
100 kilogramme	`\cutext{100}{kg}\\`
gjfak kilogramme	`\cutext{gjfak}{kg}\\`
12 kilogramme	`\cutext[kg=g]{12}{kg}\\`
1———-2	`\cuam{1----------2}\\`
1,2	`\cuam{1,2}\\`
1_1/2	`\cuam{1_1/2}\\`
kwflk	`\cuam{kwflk}\\`

`range-sign`

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

`cunum-range-sign` sets the *printed* range sign used in `\cunum` (and `\cuam`) to ⟨*string*⟩, `cutext-range-sign` sets the printed range sign used in `\cutext` and `\Cutext` to ⟨*string*⟩. Using `range-sign` sets the range signs for both `\cunum` (and `\cuam`) and `\cutext`/`\Cutext` to ⟨*string*⟩.

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

Since version 1.45 there also exits the language symbol `cutext-range-sign` (see section 7). If the *option* `cutext-range-sign` is set the language symbol will be ignored.

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

`use-phrases` `use-phrases = ⟨true/false⟩`

Setting this option to `true` replaces certain integers (see section 7.1 for more information) with their phrase counterpart. This option is set to `false` by default.

Example: For the German language:

12	`\cuam{12}\\`
12–24	`\cuam{12--24}\\`
36	`\cuam{36}\\`
	`\cusetup{use-phrases=true}`
1 Dutzend	`\cuam{12}\\`
1–2 Dutzend	`\cuam{12--24}\\`
3 Dutzend	`\cuam{36}\\`
	`\cusetup{use-phrases=true,print-numerals=true}`
ein Dutzend	`\cuam{12}\\`
ein–zwei Dutzend	`\cuam{12--24}\\`
drei Dutzend	`\cuam{36}\\`

8.2.5 Rounding options

`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}`

Note: Also negative numbers are allowed.

	`\cusetoptionfor{C,F}{round-precision=-1}`
−270 °C	`\cunum{-271,2}{C}\\`
−270 °C	`\cunum[K=C]{0.0012}{K}\\`
180 °C	`\cunum{185}{C}\\`
360–390 °F	`\cunum[C=F]{180--200}{C}\\`

`round-to-int` `round-to-int = ⟨true/false⟩`

This option is deprecated. Rounds the amount to an integer if set `true`. Use `round-precision=0` instead.

round-half

`round-half = ⟨default/commercial⟩`

This option is only important for half-way numbers (e.g. 0.005). By setting it to `default` the value will be rounded to the nearest even number. Setting it to `commercial` rounds the value away from zero.

It is set to `default` by … default.

Note: `default` actually refers to the fact that it is the default rounding algorithm used by `\fp_eval:n { round( ) }` without a third argument.

0 kg
−0 kg
1.24 kg

0.01 kg
−0.01 kg
1.25 kg

```
\cusetup{round-half=default}
\cunum{0.005}{kg}\\
\cunum{-0.005}{kg}\\
\cunum{1.245}{kg}\\
\cusetup{round-half=commercial}
\cunum{0.005}{kg}\\
\cunum{-0.005}{kg}\\
\cunum{1.245}{kg}
```

8.2.6 Fractions

eval-fraction

`eval-fraction = ⟨true/false⟩`

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

0.33 kg
0.5 kg
500 g
1.5 kg
1500 g
−500 g
1 ?⁄2 kg

```
\cusetup{eval-fraction=true}
\cunum{1/3}{kg}\\
\cunum{1/2}{kg}\\
\cunum[kg=g]{1/2}{kg}\\
\cunum{1_1/2}{kg}\\
\cunum[kg=g]{1_1/2}{kg}\\
\cunum[kg=g]{-1_1/2}{kg}\\
\cunum[kg=g]{1_?/2}{kg}\\
```

convert-fraction

`convert-fraction = ⟨true/false⟩`

By default units of fractions are not converted into another unit. Setting this option to `true` allows fractions to be evaluated when a change of units is requested (and *only* if a change of unit is requested).

1⁄3 kg
333.33 g
1 1⁄2 kg
1500 g
1 ?⁄2 kg

```
\cusetup{convert-fraction=true}
\cunum{1/3}{kg}\\
\cunum[kg=g]{1/3}{kg}\\
\cunum{1_1/2}{kg}\\
\cunum[kg=g]{1_1/2}{kg}\\
\cunum[kg=g]{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.

1/8
1/2 kg
4/5 °C
1 2/3 kg

$^{1}/_{8}$
$^{1}/_{2}$ kg
$^{4}/_{5}$ °C
1 $^{2}/_{3}$ kg

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

1/8
1/2 kg
4/5 °C
1 2/3 kg

$^{8}/_{1}$
$^{2}/_{1}$ kg
$^{5}/_{4}$ °C
1 $^{3}/_{2}$ kg

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

8.2.7 Spaces

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

Sets the length between the fraction and the number in a mixed-fraction, default is `0.1em` (because I said so; if someone has some literature or sources to look up the space, please let me know).

1 $^{2}/_{3}$
1 $^{2}/_{3}$ kg
10 $^{2}/_{3}$ kg

1 $\quad ^{2}/_{3}$
1 $\quad ^{2}/_{3}$ kg
10 $\quad ^{2}/_{3}$ kg

1$^{2}/_{3}$
1$^{2}/_{3}$ kg
10$^{2}/_{3}$ kg

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

`cutext-space` `cutext-space = {⟨string⟩}`

⟨*string*⟩ is inserted between the numeral part and the unit part when using `\cutext` and `\Cutext`. By default it is set to `\space`. Use this option if you want to e.g. insert an unbreakable space.

1 kilogramme	`\cutext{1}{kg}\\`
10 kilogramme	`\Cutext{10}{kg}\\`
	`\cusetup{cutext-space=~}`
1 kilogramme	`\cutext{1}{kg}\\`
10 kilogramme	`\Cutext{10}{kg}\\`
	`\cusetup{cutext-space={}}`
1kilogramme	`\cutext{1}{kg}\\`
10kilogramme	`\Cutext{10}{kg}\\`
	`\cusetup{cutext-space={qwe}}`
1qwekilogramme	`\cutext{1}{kg}\\`
10qwekilogramme	`\Cutext{10}{kg}\\`

`phrase-space` `phrase-space = {⟨string⟩}`

⟨*string*⟩ is inserted between the numeral part and the phrase part while using `\cuam`. By default it is set to `\space`. Use this option if you want to e.g. insert an unbreakable space.

(Switching to german)

1 Dutzend	`\cuam{12}\\`
12 Dutzend	`\cuam{144}\\`
	`\cusetup{phrase-space=~}`
1 Dutzend	`\cuam{12}\\`
12 Dutzend	`\cuam{144}\\`
	`\cusetup{phrase-space={}}`
1Dutzend	`\cuam{12}\\`
12Dutzend	`\cuam{144}\\`
	`\cusetup{phrase-space={qwe}}`
1qweDutzend	`\cuam{12}\\`
12qweDutzend	`\cuam{144}\\`

`amount-unit-space` `amount-unit-space = {⟨string⟩}`

Change the spacing for `\cunum` between the printed amount(s) and the unit. The default value is `\thinspace`.

1 kg	`\cunum{1}{kg}\\`
½ kg	`\cunum{1/2}{kg}\\`
1–2 kg	`\cunum{1--2}{kg}\\`
	`\cusetup{amount-unit-space={\hspace{1em}}}`
1 kg	`\cunum{1}{kg}\\`
½ kg	`\cunum{1/2}{kg}\\`
1–2 kg	`\cunum{1--2}{kg}\\`
	`\cusetup{amount-unit-space={}}`
1kg	`\cunum{1}{kg}\\`
½kg	`\cunum{1/2}{kg}\\`
1–2kg	`\cunum{1--2}{kg}\\`
	`\cusetup{amount-unit-space={qwe}}`
1qwekg	`\cunum{1}{kg}\\`
½qwekg	`\cunum{1/2}{kg}\\`
1–2qwekg	`\cunum{1--2}{kg}\\`

8.2.8 label & refs

`recalculate-amount`

`recalculate-amount` = ⟨*true/false*⟩

Set this option to `true` if you want to change your recipes to the given number of people set by `set-number-of-persons`. Note that only those values who have a label are changed.

`set-number-of-persons`

`set-number-of-persons` = ⟨*integer*⟩

With this option you can determine the number of people your recipes are for. Note that this option only has an effect on those who have a ⟨*label*⟩ given. It is set to `4` by default. Please also note the use of `recalculate-amount`.

2 persons
1 kg
1
1 kilogramme
2 persons

```
\culabel{anotherrecipe}{2}
\curef{anotherrecipe}~persons\\
\cunum<anotherrecipe>{1}{kg}\\
\cuam<anotherrecipe>{1}\\
\cutext<anotherrecipe>{1}{kg}\\
\curef{anotherrecipe}~persons\\
```

4 persons
2 kg
2
2 kilogramme
20 kilogramme

```
\cusetup{recalculate-amount=true}
\curef{anotherrecipe}~persons\\
\cunum<anotherrecipe>{1}{kg}\\
\cuam<anotherrecipe>{1}\\
\cutext<anotherrecipe>{1}{kg}\\
\Cutext[ref=anotherrecipe]{10}{kg}\\
```

3 persons
1.5 kg
1.5
1.5 kilogramme
15 kilogramme

```
\cusetup{set-number-of-persons=3}
\curef{anotherrecipe}~persons\\
\cunum<anotherrecipe>{1}{kg}\\
\cuam<anotherrecipe>{1}\\
\cutext<anotherrecipe>{1}{kg}\\
\Cutext[ref=anotherrecipe]{10}{kg}\\
```

2 persons
1 kg
1
1 kilogramme
10 kilogramme

```
\cusetup{set-number-of-persons=2}
\curef{anotherrecipe}~persons\\
\cunum<anotherrecipe>{1}{kg}\\
\cuam<anotherrecipe>{1}\\
\cutext<anotherrecipe>{1}{kg}\\
\Cutext[ref=anotherrecipe]{10}{kg}\\
```

1 person
0.5 kg
0.5
0.5 kilogramme
5 kilogramme

```
\cusetup{set-number-of-persons=1}
\curef{anotherrecipe}~person\\
\cunum<anotherrecipe>{1}{kg}\\
\cuam<anotherrecipe>{1}\\
\cutext<anotherrecipe>{1}{kg}\\
\Cutext[ref=anotherrecipe]{10}{kg}\\
```

`label` `label = {⟨string⟩*⟨integer⟩}`

The key-value version of `\culabel`. It defines the label ⟨*string*⟩ which is originally for ⟨*integer*⟩ people. Please note that the `*` is mandatory as it separates the string from the integer. Each label is defined globally and must be unique.

1 person
2
2 dag

4 persons
8
8 dag

```
\cusetup{label=Toast*1}
\curef{Toast}~person\\
\cuam<Toast>{2}\\
\cunum<Toast>{2}{dag}\\
\cusetup{recalculate-amount=true}
\curef{Toast}~persons\\
\cuam<Toast>{2}\\
\cunum<Toast>{2}{dag}
```

`get-label` `get-label = {⟨label⟩}`

The key-value version of \curef. Note that this key doesn't save the value inside a macro but rather prints it directly into the document.

```
                              \culabel{Schinken}{3}
3                             \cusetup{get-label=Schinken}\\
3                             \curef{Schinken}\\
                              \cusetup{recalculate-amount=true}
4                             \cusetup{get-label=Schinken}\\
4                             \curef{Schinken}\\
```

Note: \curef *is* expendable.

`ref` `ref = {⟨label⟩}`

Instead of using the first optional arguments of the commands in section 2 you may use this option. It requires a valid value and throws an error if ⟨*label*⟩ is not defined.

```
                              \culabel{Kaese}{3}
10 dm                         \cunum<Kaese>[m=dm]{1}{m}\\
10 dm                         \cunum[ref=Kaese,m=dm]{1}{m}\\
                              \cusetup{recalculate-amount=true}
13.33 dm                      \cunum<Kaese>[m=dm]{1}{m}\\
13.33 dm                      \cunum[ref=Kaese,m=dm]{1}{m}
```

```
curef-add-forbidden-unit       curef-add-forbidden-unit    = {⟨unit list⟩}
curef-remove-forbidden-unit    curef-remove-forbidden-unit = {⟨unit list⟩}
curef-clear-forbidden-units    curef-clear-forbidden-units = ⟨true/false⟩
```

There are units which do not depend on the number of folks you are cooking for, units measuring the temperature are an example. Changing those units with the label & ref system would be accidental and in the best case throw an error. With the following options you can add units to the "forbidden unit list", remove them and clear the whole list entirely.

By default the list contains `C`, `F`, `K` and `Re`.

2 m
2 kg
1 °C

1 m
1 kg
1 °C

1 m
1 kg
2 °C

2 m
2 kg
2 °C

```
\culabel{check}{2}
\cusetup{recalculate-amount=true}
\cunum<check>{1}{m}\\
\cunum<check>{1}{kg}\\
\cunum[ref=check]{1}{C}\\
\cusetup{curef-add-forbidden-unit={m,kg}}
\cunum<check>[m]{1}{m}\\
\cunum<check>[m]{1}{kg}\\
\cunum[ref=check]{1}{C}\\
\cusetup{curef-remove-forbidden-unit={C}}
\cunum<check>[m]{1}{m}\\
\cunum<check>[m]{1}{kg}\\
\cunum[ref=check]{1}{C}\\
\cusetup{curef-clear-forbidden-units=true}
\cunum<check>[m]{1}{m}\\
\cunum<check>[m]{1}{kg}\\
\cunum[ref=check]{1}{C}
```

8.3 Weird options

`check-temperature`

```
check-temperature = ⟨true/false⟩
```

Checks if the used temperature is below absolute zero. 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. To add new units see `add-temperature-to-check`.

`add-temperature-to-check`

```
add-temperature-to-check =
  {
    ⟨unit-key-1⟩ = ⟨minimum-value-1⟩ ,
    ⟨unit-key-2⟩ = ⟨minimum-value-2⟩ ,
    ...
  }
```

This option adds ⟨*unit-key-1*⟩ and so on to the list of units to be checked if `check-temperature` is active. The argument can be a comma-separated list of ⟨*unit-key*⟩ = ⟨*minimum-value*⟩. This sets the allowed minimum value of ⟨*unit-key*⟩ to ⟨*minimum-value*⟩.

Example: This package implements the allowed minimum values for the temperatures `C`, `F`, `K` and `Re` to be checked if `check-temperature` is active using:

```
\cusetup
  {
    add-temperature-to-check =
      {
        K = 0,
        C = -273.15 ,
        F = -459.67 ,
        Re = -218.52
      }
  }
```

If you want to add a new value, for example degree Rømer (which has be defined in another example) you can write:

```
\cusetup
  {
    add-temperature-to-check = { Ro = -135.90375 }
  }
```

convert-to-eV

`convert-to-eV = ⟨true/false⟩`

Converts (nearly) every unit in table 1 to electron volt or the respective derivative (if possible). 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). Also you may want to check the values if you really want to use them, just to be sure (I've checked them several times and hope they are finally correct, but mistakes happen).

	`\cusetup{convert-to-eV=true}`
560958865000000000000000000000000000 $^{eV}/_{c^2}$	`\cunum{1}{kg}\\`
130148929500000000 $^{c^3\hbar^3}/_{eV^3}$	`\cunum{1}{l}\\`
6241509126000000000 eV	`\cunum{1}{J}\\`
5067730.76 $^{c\hbar}/_{eV}$	`\cunum{1}{m}\\`
0.02 eV	`\cunum{1}{C}\\`
1519267461000000 $^{\hbar}/_{eV}$	`\cunum{1}{s}`

add-natural-unit

`add-natural-unit = ⟨unit-key⟩`

This option adds ⟨*unit-key*⟩ to the list of units `convert-to-eV` uses to determine how a unit is transformed if set to `true`.

42

`42 = ⟨true/false⟩`

Take a good guess.

	`\cusetup{42=true}`
42 kg	`\cunum{1}{kg}\\`
42 g	`\cunum[kg=g]{1}{kg}\\`
42 J	`\cunum{1.5}{J}\\`
42 °C	`\cunum{180}{C}\\`
42 s	`\cunum{15}{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. If you use mail 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. It can also take longer of GitHub, but I hope I figured out how to get a mail if a new issue is created (by not me).

Feedback and requests (commands, units, etc.) are also welcome. Please also add (if possible) an example of the desired output into the minimal example (and – if by mail – 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 the translations known to you. A list of all units (and their current translations) is given in appendix A.

10 Bens Einheitensammelsurium (Bens unit Almanac)

Units are a fascinating mess. There are so many different ones which are different and the few ones which are the same (in name at least) are *also* different, depending on geographical position, time period and probably pure spite. We can be glad that SI-units exist.

So for those units which didn't make it into table 1 and table 2, this section exists. Please note that this list is intended to be a just-for-fun list and not a compilation of every unit in existence with its exact value ordered by geographical and chronological position. I am sadly neither a historian nor very good in regards to languages. It would sound like fun, but ultimately, I wouldn't have the time. Therefore I am only taking units into account which I either found in literature (stone, canna, etc.), are well known (foot) or have some other experience with them (ell) (exception: Batman). The reason I am not including units which I found in the internet is that I would like to see those units in their "natural environment".

unit (translation) [abbreviation] Description, containing a quote or not. *Please note that most of the units are country dependent! So the translation may not have the same amount as the word it is translated to.*

Batman So . . . You wanna be Batman? Be like Bruce Wayne? Having a secret identity? Then congratulations! You *are* Batman! How much Batman depends on the location, but Wikipedia is your friend in this matter.

Rotolo$^{\text{sicilian}}$ (Rottel$^{\text{de}}$) Around 0.850 kg

Auf den Fußboden lagen vier ungereifte Käse zu je zwölf Rottel*, jeder ungefähr zehn Kilo schwer.* (see [1] page 51)

Canna$^{\text{sicilian}}$ (Rute$^{\text{de}}$, rod$^{\text{en}}$) About 2 m bzw. about 6 foot.

"Unsinn, Stella, Unsinn; was soll mir zustoßen? Sie kennen mich alle: Männer, die eine Rute *lange sind, gibt es wenige in Palermo."* (see [1] page 25)

Stone [st] 6.35 kg. According to a fellow student this unit is still used in Great Britain. I've also recently found it in a video game; in the german translation of said video game to be precise. Why is the german translation using stone and not kilogram (at least in braces)?

As we had expected, the telegramm was soon followed by its sender, and the card of Mr. Cyril Overton, Trinity College, Cambridge, announced the arrival of an enormous young man, sixteen stone [101.6 kg] *of solid bone and muscle, who spanned the doorway with his broad shoulders* [. . .] (see [2] page 988)

(Story "The missing Three Quarters")

Foot [ft] Equals exactly 0.3048 m or 12 in.

A bit of a strange unit (for me at least). Where I am from, people tend to have different feet sizes. Also present in the german translation of the video game that uses "Stone".

degree Rèamur [°Ré] Like degree Celsius, but instead of having the water boiling at 100° (Celsius), water boils at 80°. Water thankfully still freezes at 0°. Don't think that this unit is used anymore. I think I learned about in physics.

Ell Just read the Wikipedia article.

Fun Fact: At the Stephansdom in Vienna left of the main entrance are two metal bars. One is the "Tuchelle" (drapery ell, circa 78 cm), the other the "Leinenelle" (linen ell, around 89.6 cm).

cup I think the idea of having a "cup" and it not being equal to 250 ml is a bit strange, for me at least. What other sizes can a cup have? I can imagine 500 ml, but are there other sizes?

stick A unit I've made fun of because it is quite regional and doesn't make any sense for foreigners. Then I realized that I am using the unit "Packerl" in my cookery book which is also quite locally[8] and – even worse – the weight changes depending the content (See *Packerl*).

Packerl$^{\mathbf{de}}$ (small bag) I'm a bit split on this unit as I don't actually know if it exists. The reason I have the unit *Packerl* for my cookery book is that in Austria you can buy baking powder, (dry) Germ, Natrium, etc. in small bags (similar to *stick*). The problem: Depending on the content, the weight of *Packerl* differs. Not only that, but it can also differ between different producers (but not more than 2 g bzw. 0.07 oz). Here is a table:

1 Packerl	Backpulver	(baking powder)	16 g	(0.56 oz)
	Natrium		14 g	(0.49 oz)
	Vanillin(-zucker)	(vanillin(-sugar))	8 g	(0.28 oz)
	Germ*		7 g	(0.25 oz)

*Tockengerm (dry Germ) to be precise

For what kind of thing do I need *Natrium* for?

A Translations

This section contains the list of available translations. Each table shows the available translations regarding the unit symbol, the unit name (printed if `\cutext` or `\Cutext` is used) and the plural form (if different from the singular form). A second table shows the translations used for phrases (if given).

If a translation is not available a "—" is shown.

[8] And maybe doesn't even exist outside my family

A.1 English

⟨*unit-key*⟩	printed unit	unitname	(plural)	gender
`kg`	kg	kilogramme		m
`dag`	dag	decagramme		m
`g`	g	gramme		m
`oz`	oz	ounce		m
`lb`	lb	pound	(pounds)	m
`C`	°C	degree Celsius	(degrees Celsius)	m
`F`	°F	degree Fahrenheit	(degrees Fahrenheit)	m
`Re`	°Ré	degree Réaumur	(degrees Réaumur)	m
`K`	K	kelvin		m
`d`	d	day	(days)	m
`h`	h	hour	(hours)	m
`min`	min	minute	(minutes)	m
`s`	s	second	(seconds)	m
`m`	m	metre	(metres)	m
`dm`	dm	decimetre	(decimetres)	m
`cm`	cm	centimetre	(centimetres)	m
`mm`	mm	millimitre	(millimitres)	m
`in`	in	inch	(inches)	m
`l`	ℓ	litre	(litres)	m
`dl`	dl	decilitre	(decilitres)	m
`cl`	cl	centilitre	(centilitres)	m
`ml`	ml	millilitre	(millilitres)	m
`cal`	cal	calorie	(calories)	m
`kcal`	kcal	kilocalorie	(kilocalories)	m
`J`	J	joule	(joules)	m
`kJ`	kJ	kilojoule	(kilojoules)	m
`eV`	eV	electron volt		m
`pn`	pinch	pinch	(pinches)	m
`EL`	tbsp.	tablespoon	(tablespoons)	m
`TL`	tsp.	teaspoon	(teaspoons)	m
`csp`	csp.	coffeespoonful		m
`dsp`	dsp.	dessertspoonful		m
`ssp`	ssp.	saltspoonful		m
`Msp`	Msp.	—		m
`decimal-mark`	—	.	—	m
`one(m)`	—	one	—	m
`one(f)`	—	one	—	m
`one(n)`	—	one	—	m

A.2 american

⟨*unit-key*⟩	printed unit	unitname	(plural)	gender
`kg`	kg	kilogram		m
`dag`	dag	decagram		m
`g`	g	gram		m
`oz`	oz	ounce		m
`lb`	lb	pound	(pounds)	m
`C`	°C	degree Celsius	(degrees Celsius)	m
`F`	°F	degree Fahrenheit	(degrees Fahrenheit)	m
`Re`	°Ré	degree Réaumur	(degrees Réaumur)	m
`K`	K	kelvin		m
`d`	d	day	(days)	m
`h`	h	hour	(hours)	m
`min`	min	minute	(minutes)	m
`s`	s	second	(seconds)	m
`m`	m	meter	(meters)	m
`dm`	dm	decimeter	(decimeters)	m
`cm`	cm	centimeter	(centimeters)	m
`mm`	mm	millimiter	(millimiters)	m
`in`	in	inch	(inches)	m
`l`	ℓ	liter	(liters)	m
`dl`	dl	deciliter	(deciliters)	m
`cl`	cl	centiliter	(centiliters)	m
`ml`	ml	milliliter	(milliliters)	m
`cal`	cal	calorie	(calories)	m
`kcal`	kcal	kilocalorie	(kilocalories)	m
`J`	J	joule	(joules)	m
`kJ`	kJ	kilojoule	(kilojoules)	m
`eV`	eV	electron volt		m
`pn`	pn.	pinch	(pinches)	m
`EL`	tbsp.	tablespoon	(tablespoons)	m
`TL`	tsp.	teaspoon	(teaspoons)	m
`csp`	csp.	coffeespoonful		m
`dsp`	dsp.	dessertspoonful		m
`ssp`	ssp.	saltspoonful		m
`Msp`	Msp.	—		m
`decimal-mark`	—	.	—	m
`one(m)`	—	one	—	m
`one(f)`	—	one	—	m
`one(n)`	—	one	—	m

A.3 German

⟨*unit-key*⟩	printed unit	unitname	(plural)	gender
`kg`	kg	Kilogramm		n
`dag`	dag	Dekagramm		n
`g`	g	Gramm		n
`oz`	oz	Unze		f
`lb`	lb	Pfund		n
`C`	°C	Grad Celsius		m
`F`	°F	Grad Fahrenheit		m
`Re`	°Ré	Grad Réamur		m
`K`	K	Kelvin		n
`d`	d	Tag	(Tage)	m
`h`	h	Stunde	(Stunden)	f
`min`	min	Minute	(Minuten)	f
`s`	s	Sekunde	(Sekunden)	f
`m`	m	Meter		n
`dm`	dm	Dezimeter		n
`cm`	cm	Centimeter		n
`mm`	mm	Millimeter		n
`in`	in	Zoll		m
`l`	l	Liter		m
`dl`	dl	Deziliter		m
`cl`	cl	Centiliter		m
`ml`	ml	Milliliter		m
`cal`	cal	Kalorie	(Kalorien)	f
`kcal`	kcal	Kilokalorie	(Kilokalorien)	f
`J`	J	Joule		m
`kJ`	kJ	Kilojoule		m
`eV`	eV	Elektronenvolt		n
`pn`	Prise	Prise	(Prisen)	f
`EL`	EL	Esslöffel		m
`TL`	TL	Teelöffel		m
`csp`	KL	Mokkalöffel		m
`dsp`	dsp.	—		m
`ssp`	ssp.	—		m
`Msp`	Msp.	Messerspitze	(Messerspitzen)	f
`decimal-mark`	—	,	—	m
`one(m)`	—	ein	—	m
`one(f)`	—	eine	—	m
`one(n)`	—	ein	—	m

⟨*Phrase-key*⟩	phrase	(plural)	gender
12	Dutzend		n

Some further phrases, just to write them down (they are not implemented, as they are barely used).

⟨*number*⟩	name	Note		(plural)	gender
60	Schock	(5 Dutzend,	$12 * 5$)		n
144	Gros	(12 Dutzend,	$12 * 12$)		n
1728	Großgros	(12 Groß,	$12 * 144$)		n

Note that Großgros has other (probably more common) synonyms.

A.4 French

⟨*unit-key*⟩	printed unit	unitname	(plural)	gender
`kg`	kg	kilogramme	(kilogrammes)	m
`dag`	dag	décagramme	(décagrammes)	m
`g`	g	gramme		m
`oz`	oz	once		f
`lb`	lb	livre	(livres)	f
`C`	°C	degré Celsius	(degrés Celsius)	m
`F`	°F	kelvin	(kelvins)	m
`Re`	°Ré	échelle Réaumur	(degrés Réaumur)	m
`K`	K	degré Fahrenheit	(degrés Fahrenheit)	m
`d`	d	jour	(jours)	m
`h`	h	heure	(heures)	f
`min`	min	minute	(minutes)	f
`s`	s	seconde	(secondes)	f
`m`	m	mètre	(mètres)	m
`dm`	dm	décimètre	(décimètres)	m
`cm`	cm	centimètre	(centimètres)	m
`mm`	mm	millimètre	(millimètres)	m
`in`	po	pouce	(pouces)	m
`l`	L	litre	(litres)	m
`dl`	dL	décilitre	(décilitres)	m
`cl`	cL	centilitre	(centilitres)	m
`ml`	mL	millilitre	(millilitres)	m
`cal`	cal	calorie		m
`kcal`	kcal	kilocalorie	(kilocalories)	m
`J`	J	joule	(joules)	m
`kJ`	kJ	kilojoule	(kilojoules)	m
`eV`	eV	électron-volt	(électron-volts)	m
`pn`	pinch	pincée		f
`EL`	c.à.s.	cuillère à soupe		f
`TL`	c.à.c.	cuillère à café		f
`csp`	csp.	—		m
`dsp`	dsp.	—		m
`ssp`	ssp.	—		m
`Msp`	Msp.	—		m
`decimal-mark`	—	.	—	m
`one(m)`	—	un	—	m
`one(f)`	—	une	—	m
`one(n)`	—	un	—	m

If the spoons should be extra full:

- cuillère à soupe rase
- cuillère à café rase

B US, Imperial and Other units

As source [5] has been used for imperial units, while [4] and [3] were used for U.S. units. I hope someone will find this bringing together useful.

1 yard $= 0.9144\,\text{m}$ (exact)
1 yard = 3 foot
1 yard = 36 Inch
1 Inch $= 0.0254\,\text{m}$ (also exact)

1 liter $= 1\,\text{dm}^3$	
1 gallon $= 4.546\,09$ liter (exact)	1 U.S. gallon $= 231\,\text{Inch}^3 = 231 \times 0.016\,387\,064$ liter
1 gallon = 4 Quart	1 U.S. gallon = 4 $\text{Quart}^{\text{U.S.}}$
1 gallon = 8 Pint	1 U.S. gallon = 8 $\text{Pint}^{\text{U.S.}}$
1 gallon = 32 Gill	1 U.S. gallon = 32 $\text{Gill}^{\text{U.S.}}$
1 gallon = 160 fl. oz	1 U.S. gallon = 128 $\text{fl. oz}^{\text{U.S.}}$
1 fl. oz $= 0.028\,413\,062\,5$ liter	1 $\text{fl. oz}^{\text{U.S.}} = 0.029\,573\,529\,562\,5$ liter

Note 1: I think the American $\text{fl. oz}^{\text{U.S.}}$ is more common. Maybe. Most bottles have something like 10 fl. oz, which they say is equal to 30 ml. This would work really well with $\text{fl. oz}^{\text{U.S.}}$.

Note 2: Sometimes "fl. oz" is written without the dot. I am also not sure what kind of spacing has to be between "fl." and "oz" (currently using `\thinspace`).

Note 3: This maybe sounds stupid, but could we introduce something like "flouz", "floiz" and "floez"? "flouz" would be "$\text{fl. oz}^{\text{U.S.}}$", "floiz" would be "Imperial fl. oz" and "floez" would simply be equal to 30 ml?

For "stick" see [6].

1 lb $= 0.453\,592\,37\,\text{kg}$ (exact)
1 lb = 16 oz
1 lb $= \frac{1}{14}$ st
1 lb $= \frac{175}{12}$ ounce troy
1 lb = 4 stick

1 cup	≈ 0.25 litre	$= 250\,\text{ml}$	1 $\text{cup}^{\text{U.S.}}$	$= 8\ \text{fl. oz}^{\text{U.S.}}$
1 tablespoon	≈ 0.015 litre	$= 15\,\text{ml}$	1 $\text{tablespoon}^{\text{U.S.}}$	$= \frac{1}{2}\ \text{fl. oz}^{\text{U.S.}}$
1 teaspoon	≈ 0.005 litre	$= 5\,\text{ml}$	1 $\text{teaspoon}^{\text{U.S.}}$	$= \frac{1}{6}\ \text{fl. oz}^{\text{U.S.}}$

Note 1: I tested the approximation for tablespoon with water ($1\,\text{mg} \approx 1\,\text{mg}$) and the approximation looks good enough. It of course depends on how full you fill your spoon.

References

[1] Guiseppe Tomasi di Lampedusa, *Der Gattopardo*, Piper, Volume 8 (2018), ISBN 978-3-492-24586-9

[2] Sir Arthur Conan Doyle, *Sherlock Holmes The Complete Novels and Stories Volume II*, Bantam Books

[3] *Guide for the Use of the International System of Units (SI)*, NIST Special Publication 811, 2008 Edition, Ambler Thompson and Barry N. Taylor

[4] *The International System of Units (SI) – Conversion Factors for General Use*, NIST Special Publication 1038, May 2006, Kenneth Butcher, Linda Crown and Elizabeth J. Gentry

[5] *Weights and Measures Act 1985*, `https://www.legislation.gov.uk/ukpga/1985/72`

[6] `https://cooking.stackexchange.com/questions/784/translating-cooking-terms-between-us-uk-au-ca-nz`

Change History

2016/06/11
- General: Added the package option to load 'fmtcount'. 1

2016/08/31
- General: Fixed calculation: degree Reamur to eV 1
  - Initial version 1

2016/09/03
- General: Added units 'ssp', 'csp', 'dsp' 1
  - British English: 'pinch' is written in full 1
  - English unit: litre (and only litre) uses the curly l ℓ now 1
  - Separated Messerspitze and pinch 1

2016/09/05
- General: New message: 'obsolete-command' 1
  - Replaced `\cufrac` by `\cuam` 1

2016/09/09
- General: `\@@_calculate_input_and_store_in:nN` optimiert durch neue property-key: single. 1
  - Add 'single' to property list of singlekeys. 1
  - Changed name from `\@@_cunum_parse_range` (and derivatives) to `\@@_cutext_parse_range`. 1
  - Changed name from `\@@_parse_fraction_in_input:www` to `\@@_parse_mixed_fraction_in_input:www` 1
  - Corrected mistake: 'ELektronenvolt' (note uppercase L) to 'Elektronenvolt' in german. 1
  - Delete 'single' from property lists of singlekeys cause it is not as safe as I thought. 1
  - In `\@@_cutext_default:nnn` it is only checked once if a range is inside. 1

2016/09/16
- General: Only use `\phantom` if the argument (for `\phantom`) is not empty. 1

2016/09/26
- General: `\cuaddsinglekeys` now tests if the unit exists (it didn't before). 1
  - New option (and needed macros): add-temperature-to-check. 1
  - New option: 'round-half'. 1
  - Recalculated all electron volt values for conversion (as 'kg' was wrong before). Let's hope they are correct this time. 1
  - Replaced `\prop_clear_new:c` by `\prop_clear:c`. 1

2016/10/19
- General: 'convert-to-eV' now also as optional argument available. 1
  - Option 'load-time-option' now spells 'available' correct. 1
  - Update of documentation. 1
  - Use `\keys_set:nn` only if second argument is not empty. 1

2016/10/28
- General: `\cutext` (and `\Cutext`) and `\cuam` now parse their input like `\cunum`. This is needed as they also need to be changed. 1
  - Start implementation of "Change recipe from *n* to *m* persons.". 1

2016/10/29
- General: Tiding code: Now every command is separated into a "calc" function, a "print numeric value"

and a “print unit” (if there) function. At least, that’s the plan. 1

2016/10/30

- General: Fractions should now deal correctly with minus signs. 1

2016/11/07

- General: Finished writing v1.10. 1

2016/11/13

- General: `\cutext`, `\Cutext` and `\cuam` check their input, allows conversion of units. 1
- Change amounts for specific number of persons. 1
- New commands: `\culabel` and `\curef`. 1
- New commands: `\declarecookingunit` and `\providecookingunit`. 1
- New options: `cuam-version` and `cutext-version` 1
- New options: `cutext-to-cunum`, `cutext-change-unit` and `cutext-space`. 1
- New options: `recalculate-amount` and `set-number-of-persons`, `label`, `get-label`, `ref`. 1

2017/03/10

- General: `\curef` is now defined by `\NewExpandableDocumentCommand` instead of the Declare variant. 1
- Removed `\translate` and others from code and replaced them with wrapper-macros. 1
- Removed things like ’cu-unit’ from translate input and placed them into separate tl’s. 1

2017/10/23

- General: Added “phrases”. 1
- Added unit “stick” (of butter). 1
- New option: `amount-unit-space`. 1
- New option: `phrase-space`. 1
- New option: `print-numerals`. 1
- New option: `set-cutext-translation-message`. 1
- New option: `use-phrases`. 1
- Now checks for ranges if both values can be printed as numerals (if activated) (bug fix). 1
- Replaced translator by translations. 1
- Reworked quite a lot of code. 1

2018/04/20

- General: Add “Division-by-zero” error. 1
- Allow round precision to be negative. 1
- Change large portions of code. 1
- Cooking Units-keys are not allowed to contain either “,” or “/”. 1
- Fix argument specifiers. 1
- Introduce key-groups (weight, volume, etc.). 1
- New feature: Hooks 1
- New Option: `42`. 1
- New option: `add-unit-to-group` 1
- New option: `erase-all-options-for` 1
- New options: `expand-both`, `expand-amount`, `expand-unit`. 1
- New options: `set-option-for` & `add-option-for`. 1
- New parsing algorithm. Hopefully better error recovery (if signs for fractions are in wrong order e.g.) 1
- Option: `add-natural-unit`. 1

2018/06/05

- General: `set-unknown-message`: Fix default value. 1
- Add “range-sign” for translations (not usable yet). 1
- Bugfix (`phrases`): Use the phrase from the first amount to check the second (and don’t parse through the second amount). 1
- Bugfix (`unit-change`): `convert-to-eV` can be again used as a local argument. 1
- true) will print the second word small. 1
- Change (`amount-not-known`): Change message a bit. 1
- Convert `clist` to `seq` if possible. 1
- Fix some more argument specifiers. 1
- Improve error-recovery by a lot|hdpindex 1
- Remove unnecessary variants. 1
- Renaming of some internal commands. 1
- Rework parsing code (again). As this is basically an improved version of the old parsing algorithm, there is no huge version change. 1
- This version introduces mayor internal changes. For users not many things change. 1

2018/09/24

- General: Changes prefix from `cooking_units` to `cookingunits`. 1
- Improved french (not in general, only for this package) 1

New language symbols:
cutext-range-sign 1
New section in documentation. 1

Remove exhaustive expansion from internals (shouldn't change anything for users). 1

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.