The **cooking-units** package*

Ben Vitecek
b.vitecek@gmx.at

2022/12/10

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).

Contents

1 Introduction .............................................. 2
  1.1 Supported languages ........................................................................... 2
  1.2 What’s new? ......................................................................................... 3

2 The Commands .................................................... 3

3 Label & refs: Changing the amount of the recipe .................................... 5

4 Good to know stuff .................................................................................. 6
  4.1 Rounding temperatures ........................................................................ 6
  4.2 At which point is the plural used? .......................................................... 6

5 Predefined units & some notes .................................................................. 7

6 Defining units ........................................................................................... 7

7 Defining options to change units ............................................................. 10
  7.1 Obsolete Commands ............................................................................ 14

8 Language support ....................................................................................... 15
  8.1 Phrases ................................................................................................. 17

*This document corresponds to *Benedikt Vitecek* v3.00, dated 2022/12/10.
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 \LaTeX{}3 (which is much more pleasant than \LaTeX{}2e) and here it is.
1.1 Supported languages

- German
- English
- French

Want to contribute a new language or make a correction to an existing one? See section 11 for more details. Wanna just check the existing translations? See appendix A.

1.2 What’s new?

Since v3.00 fraction and ranges can now be combined! Furthermore, products are introduced! Check the next section for more info.

2 The Commands

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

- $\texttt{\textbackslash num}{⟨\text{label}⟩}{⟨\text{options}⟩}{⟨\text{amount}⟩}{⟨\text{space}⟩}{⟨\text{unit-key}⟩}$
- $\texttt{\textbackslash cunum}{⟨\text{label}⟩}{⟨\text{options}⟩}{⟨\text{amount}⟩}{⟨\text{unit-key}⟩}$
- $\texttt{\textbackslash cutext}{⟨\text{label}⟩}{⟨\text{options}⟩}{⟨\text{amount}⟩}{⟨\text{unit-key}⟩}$
- $\texttt{\textbackslash Cutext}{⟨\text{label}⟩}{⟨\text{options}⟩}{⟨\text{amount}⟩}{⟨\text{unit-key}⟩}$
- $\texttt{\textbackslash cuam}{⟨\text{label}⟩}{⟨\text{options}⟩}{⟨\text{amount}⟩}$
- \texttt{\textbackslash cusetup}{⟨\text{options}⟩}

Numbers and units are printed using $\texttt{\textbackslash cunum}$. The numerical part can interpret _ and / as (mixed) fractions, -- as a separator for ranges and x or \texttt{\textbackslash times} for products; to convert units use the option $⟨\text{old-unit}⟩=⟨\text{new-unit}⟩$. It furthermore allows the sign ? to be used as a placeholder for not known amounts and raises a warning to remind you that this amount needs a check-up. [[space]] adds a space between the number and the unit using \texttt{\textbackslash phantom}.

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

⟨label⟩ is explained in section 3.

\footnote{New keys can be added and defined, see section 5 and section 6 for further information.}
\footnote{You can customize this behavior, see section 9}
Decimal numbers are automatically rounded to 2 digits after the colon, temperatures (°C, °F, K and °R) are automatically rounded to integers.³

\texttt{cutext} and \texttt{Cutext} print the number and the written name of the unit. Since v1.10 it works similar⁴ to \texttt{cunum}: it allows the conversion between units and interprets the numerical part (again _ and / are used for (mixed) fractions, -- for ranges and \texttt{x} or \texttt{times} for products). Furthermore, \texttt{cutext} and \texttt{Cutext} allow the usages of numerals (see section 9.1 for more information).

\begin{verbatim}
1 litre \texttt{cutext\{1\}\{l\}1}
1 litre \texttt{Cutext\{1\}\{l\}1}
1 to 2 litres \texttt{Cutext\{1--2\}\{l\}1}
1 × 2 litres \texttt{Cutext\{1\times2\}\{l\}1}
12 litres \texttt{cutext\{12\}\{l\}1}
13 litres \texttt{Cutext\{13\}\{l\}1}
\end{verbatim}

and using (e.g.) package option \texttt{use-fmtcount-numerals=true}:

\begin{verbatim}
one litre \texttt{cutext\{1\}\{l\}1}
One litre \texttt{Cutext\{1\}\{l\}1}
one to two litres \texttt{cutext\{1--2\}\{l\}1}
One to two litres \texttt{Cutext\{1--2\}\{l\}1}
One × two litres \texttt{Cutext\{1\times2\}\{l\}1}
twelve litres \texttt{cutext\{12\}\{l\}1}
13 litres \texttt{Cutext\{13\}\{l\}1}
\end{verbatim}

You can customize the numeral functions used with \texttt{numeral-function} and \texttt{Numeral-function}.

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

³You can – of course – change this behavior, see section 9.
⁴One could also say “exactly like”.
1000 millilitres
1000 millilitres
1000 to 2000 millilitres
1000 × 2000 millilitres
12000 millilitres
13000 millilitres
? litres
½ litre

\text{works like \texttt{cnum, but without a unit, so changing units doesn’t affect it. It has the same syntax as \texttt{cnum.}}}

3
2–3
2 × 3 × 4
\frac{2}{3}
1\frac{1}{2}
\frac{2}{3} – 1\frac{1}{2}
\frac{2}{3} × 1 × 1\frac{1}{2}

Furthermore it allows the concept of “phrases” (replacing a positive integer by a word; such as “12” becoming “dozen”) which can be activated by the option \texttt{use-phrases} (as I don’t know any English phrases, I switched the language to German for the following examples)

11
1 Dutzend
13
2 Dutzend
1–2 Dutzend
12–13
1 × 2 × 3 Dutzend
18
5 Dutzend

\text{\selectlanguage{ngerman}}
\text{\cusetup{use-phrases=true}}

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:

• \texttt{\culabel{⟨label⟩}{⟨number of persons⟩}}
• \texttt{\curef{⟨label⟩}}

\footnote{At least I think}
The first one is the important one: It defines a \textit{(label)} for a recipe which is initially for \textit{(number of persons)}. Afterwards \textit{(label)} can be used to tell the commands from section 2 that the given amounts are for \textit{(number of persons)}. Each \textit{(label)} must be unique and an error is raised if a \textit{(label)} is already defined.

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

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

\begin{verbatim}
\culabel{recipe}{2}
\end{verbatim}

recipe for 2 persons:
10–20\,dag flour,\hspace{1cm} \cunum<recipe>{10--20}{dag} flour,\hspace{1cm} \\
\frac{1}{2} \ell \text{ water,} \hspace{1cm} \cunum<recipe>{1/2}{l} water,\hspace{1cm} \\
10\text{ gramme nuts,} \hspace{1cm} \cutext[ref=recipe]{10}{g} nuts,\hspace{1cm} \\
2--3\text{ eggs,} \hspace{1cm} \cuam<recipe>{2--3}{eggs},\hspace{1cm} \\
180\,\text{C (356\,\text{F}) open fire} \hspace{1cm} (\cunum[C=F]{180}{C}) \text{ open fire}
\end{verbatim}

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

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

\begin{verbatim}
\culabel{recipe}{2}
\end{verbatim}

recipe for 4 persons:
20–40\,dag flour,\hspace{1cm} \cunum<recipe>{10--20}{dag} flour,\hspace{1cm} \\
1\ell \text{ water,} \hspace{1cm} \cunum<recipe>{1}{l} water,\hspace{1cm} \\
20\text{ gramme nuts,} \hspace{1cm} \cutext[ref=recipe]{10}{g} nuts,\hspace{1cm} \\
4--6\text{ eggs,} \hspace{1cm} \cuam<recipe>{2--3}{eggs},\hspace{1cm} \\
180\,\text{C (356\,\text{F}) open fire} (\cunum[C=F]{180}{C}) \text{ open fire}
\end{verbatim}

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

4 Good to know stuff

4.1 Rounding temperatures

By default temperatures are rounded to integers (using \texttt{round-precision=0}). Since v1.30 it is possible to round amounts to a negative precision. If you want to round temperatures to the tens see the following example (\texttt{\cusetoptionfor} is described in section 9.2.1).
182 °C
356 °F
144 °Ré
453 K

180 °C
360 °F
140 °Ré
450 K

4.2 At which point is the plural used?

While using \cutext and \Cutext one may wonder which rules are used in order to determine if the printed unit is singular or plural. If rules for a specific language are not found the default ruleset is used. There are currently two rule sets: One default and one for the french language.

Each set has three separate rules: one for “normal” numbers, one for ranges and one for fractions.

French

Use singular if:

number The absolute value of the number is smaller 2

fraction The absolute value of the evaluated fraction is smaller 2. (e.g. $1 \frac{1}{2} = 1.5$ uses singular, $2 \frac{3}{4} = 2.75$ uses plural).

range/product Take the last item and use one of the rules above.

For sources see [7], [8] & [9].

Default (all other languages)

Use singular if:

number The number is equal to 1.

fraction There is no mixed fraction part and the numerator is equal to 1.

range/product Take the last item and use one of the rules above.

5 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 transformations) are shown in table 2. Table 3 pretty much exists just for fun.

\footnote{Thanks a lot to Alexis Jeandeau! I never would have imagined that the french language uses the sinfular for everything smaller 2.}
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 no translation is defined for the given language. The translations used for \textit{cutext} and \textbf{Cutext} are shown in appendix A. Note that “electron volt” exists just for fun.

<table>
<thead>
<tr>
<th>description</th>
<th>key</th>
<th>default</th>
<th>description</th>
<th>key</th>
<th>default</th>
</tr>
</thead>
<tbody>
<tr>
<td>kilogramme</td>
<td>kg</td>
<td>kg</td>
<td>metre</td>
<td>m</td>
<td>m</td>
</tr>
<tr>
<td>decagramme</td>
<td>dag</td>
<td>dag</td>
<td>decimetre</td>
<td>dm</td>
<td>dm</td>
</tr>
<tr>
<td>gramme</td>
<td>g</td>
<td>g</td>
<td>centimetre</td>
<td>cm</td>
<td>cm</td>
</tr>
<tr>
<td>ounce</td>
<td>oz</td>
<td>oz</td>
<td>millimetre</td>
<td>mm</td>
<td>mm</td>
</tr>
<tr>
<td>pound</td>
<td>lb</td>
<td>lb</td>
<td>inch</td>
<td>in</td>
<td>in</td>
</tr>
<tr>
<td>stick (of butter)</td>
<td>stick</td>
<td>stick</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>day</td>
<td>d</td>
<td>d</td>
<td>litre</td>
<td>l</td>
<td>l</td>
</tr>
<tr>
<td>hour</td>
<td>h</td>
<td>h</td>
<td>decilitre</td>
<td>dl</td>
<td>dl</td>
</tr>
<tr>
<td>minute</td>
<td>min</td>
<td>min</td>
<td>centilitre</td>
<td>cl</td>
<td>cl</td>
</tr>
<tr>
<td>second</td>
<td>s</td>
<td>s</td>
<td>millilitre</td>
<td>ml</td>
<td>ml</td>
</tr>
<tr>
<td>calorie</td>
<td>cal</td>
<td>cal</td>
<td>degree Celsius</td>
<td>°C</td>
<td>°C</td>
</tr>
<tr>
<td>kilocalorie</td>
<td>kcal</td>
<td>kcal</td>
<td>degree Fahrenheit</td>
<td>°F</td>
<td>°F</td>
</tr>
<tr>
<td>joule</td>
<td>J</td>
<td>J</td>
<td>degree Réaumur</td>
<td>°Ré</td>
<td>°Ré</td>
</tr>
<tr>
<td>kilojoule</td>
<td>kJ</td>
<td>kJ</td>
<td>kelvin</td>
<td>K</td>
<td>K</td>
</tr>
<tr>
<td>electron volt</td>
<td>eV</td>
<td>eV</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

6 Defining units

New units can be defined using

- \texttt{\texttt{\textbackslash DECLARECOOKINGUNIT\{}\{symbol/key-val-list\}\}\{unit-key\}\}}
- \texttt{\texttt{\textbackslash NEWCOOKINGUNIT\{}\{symbol/key-val-list\}\}\{unit-key\}\}}
- \texttt{\texttt{\textbackslash PROVIDECOOKINGUNIT\{}\{symbol/key-val-list\}\}\{unit-key\}\}}
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. For “tablespoon” and “teaspoon” I used the german abbreviations “EL” and “TL” (because I forgot to change them initially).

<table>
<thead>
<tr>
<th>description</th>
<th>key</th>
<th>symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>pinch</td>
<td>pn</td>
<td>pinch</td>
</tr>
<tr>
<td>tablespoon</td>
<td>EL</td>
<td>EL</td>
</tr>
<tr>
<td>teaspoon</td>
<td>TL</td>
<td>TL</td>
</tr>
<tr>
<td>dessertspoonful</td>
<td>dsp</td>
<td>dsp.</td>
</tr>
<tr>
<td>coffeespoonful</td>
<td>csp</td>
<td>csp.</td>
</tr>
<tr>
<td>saltspoonful</td>
<td>ssp</td>
<td>ssp.</td>
</tr>
<tr>
<td>Messerspitze (point of a knife)</td>
<td>Msp</td>
<td>Msp.</td>
</tr>
</tbody>
</table>

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).

<table>
<thead>
<tr>
<th>unit-key</th>
<th>symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>eVc^-2</td>
<td>$eV/c^2$</td>
</tr>
<tr>
<td>hbareV-1</td>
<td>$\hbar/eV$</td>
</tr>
<tr>
<td>chbareV-1</td>
<td>$\hbar c/eV$</td>
</tr>
<tr>
<td>(chbareV-1)^3</td>
<td>$\hbar^3 c^3/eV^3$</td>
</tr>
</tbody>
</table>

The commands define the unit <unit-key>. Note that <unit-key> can neither contain / nor ;, but it is allowed to be a command since v2.00 (see examples below).

If the key is not the same as the printed symbol use the optional argument. It can either contain the symbol you want printed or a key-value list (see below) for more advanced adjustments.

\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 (unless you change it using a key below).

Some examples:

\declarecookingunit{kg}
\declarecookingunit{g}
\declarecookingunit[Msp.] {Msp}
\declarecookingunit[\ensuremath{{}^\circ}\kern-\scriptspace C] {C}
\declarecookingunit{\%} % can use commands now

Note: The definition of the printed degree Celsius is copied and pasted from (a maybe
Those keys can only be used in the optional argument of \declarcokinderingunit, \newcookingunit or \providecookingunit. They can be used to define some properties of the unit during its initialization.

symbol allows you to set the printed symbol of the unit. A similar effect can be achieved by just using the optional argument. Use this option if you want to use other keys during the definition. This symbol is used as a fallback for all languages, if no explicit symbol is found for said language.

gender sets the gender of the unit (default ist m). Allowed is m, f or n. Note that this sets the default gender for all languages.

set-option allows to add some key-vals to the specific unit which are activated once the unit is used. See page 18.

add-to-group adds the unit defined to \{group\}. See section 9.2.1 for more information.

natural-unit is a simple true/false switch. If true the unit will be specified to be a “natural-unit”. This is more or less a joke option.

\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 key-chain. \{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.

**Example:** Your homework is to change the unit of energy kg m$^2$s$^{-2}$ into 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 kg m$^2$/s$^2$ is equal to 196829101.34 oz in$^2$/min$^2$.

**Note:** As this is a bit more experimental and can easily lead to overflow-errors, no actual \LaTeX keys are created with \declarecookingderivatives. Internally the keys and possible values are stored in a huge property list. If an unknown key is encountered, it checks if said key can be found in the property list.

### 7 Defining options to change units

Options (to change units) can be newly defined or added to already existing ones using
\cudefinekeychain
\cudefinesinglekey
\cuaddtokeychain
\cuaddsinglekeys

\cudefinekeychain
\cudefinesinglekey

\cudefinekeychain
\cudefinesinglekey

\cudefinekeychain\cudefinesinglekey
\cudefinekeychain
\cudefinesinglekey

If you define new units (see section 6) and cannot add them to already existing keys you may use \cudefinekeychain or \cudefinesinglekey respectively to define new key-chains or single keys.

\cudefinekeychain collects the unit-key’s given and defines a key-chain. This allows you to change every unit into every other unit given in the list. So \langle unit-key-1 \rangle can take \langle unit-key-1 \rangle, \langle unit-key-2 \rangle, \langle unit-key-3 \rangle, ... as values; \langle unit-key-2 \rangle can take \langle unit-key-1 \rangle, \langle unit-key-2 \rangle, \langle unit-key-3 \rangle, ... as values, etc. Please note that \langle ... \rangle 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 a single key \langle unit-key-1 \rangle with the values \langle unit-key-1 \rangle, \langle unit-key-2 \rangle, etc. The advantage of this command is that now \langle ... \rangle can be a formula and the numerical input of \cunum, etc. 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
- ...

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

\cudefinekeychain
{
  {d} { 1 }
  {h} { 24 } \%\% 1 day are 24 hours
  {min} { 1440 } \%\% 1 day are 1440 minutes
  {s} { 86400 } \%\% 1 day are 86400 seconds
}

**Note:** The value of the first item can be something different from 1. So something like this is also possible:

\cudefinekeychain
{
  {kg} { 0.4535924 }
  {dag} { 45.35924 }
  {g} { 453.5924 }
  {oz} { 16 }
  {lb} { 1 }
}

**Example:** To convert degree Fahrenheit to degree Celsius, kelvin and degree Réamur one needs the formulas\(^7\)

\[
    T_C = (T_F - 32) \cdot \frac{5}{9}
\]
\[
    T_K = (T_F - 459.67) \cdot \frac{5}{9}
\]
\[
    T_Re = (T_F - 32) \cdot \frac{4}{9}
\]

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 \(F\) with values \(C\), \(K\) and \(Re\) is defined by:

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

\(^7\)See Wikipedia.
This defines the key \texttt{F} with the values \texttt{F}, \texttt{C}, \texttt{K} and \texttt{Re}.

\begin{verbatim}
cuaddtokeychain   \cuaddtokeychain
\cuaddsinglekeys  {
  \{\texttt{unit-key-1}\} \{\texttt{value}\}
  \{\texttt{unit-key-2}\} \{\ldots \texttt{unit-key-2 are } \langle \texttt{value} \rangle \texttt{unit-key-1}\}
  \{\texttt{unit-key-3}\} \{\ldots \texttt{unit-key-3 are } \langle \texttt{value} \rangle \texttt{unit-key-1}\}
  \ldots
}\cuaddsinglekeys\{\texttt{unit-key-1}\}
{
  \{\texttt{unit-key-2}\} \{1 \texttt{unit-key-2 are } \ldots \texttt{unit-key-1}\}
  \{\texttt{unit-key-3}\} \{1 \texttt{unit-key-3 are } \ldots \texttt{unit-key-1}\}
  \ldots
}
\cuaddtokeychain\texttt{first parses through its unit-list and searches for a base unit key which is already in a key-chain (aka. was defined by } \texttt{cudefinekeychain}). \texttt{The other units, not yet part of a key-chain, are added to the same key-chain as the base unit. So the newly added units are available as a key and a value for the other units in the same key-chain. Note that } \langle \ldots \rangle \texttt{must be a number.}

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

\textbf{Example:}\hspace{1em} 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. \texttt{cuaddkeys} or \texttt{cuadddtokeys} (use the one best fitting). This example uses the first, the next the latter one.

\begin{verbatim}
newcookingunit{st} \%% defining new unit ‘stone’
cuaddtokeychain
{
  \{\texttt{lb}\} \{14\} \%% unit already in a key-chain.
  \{\texttt{st}\} \{1\} \%% new unit. 1st = 14lb
}
\cunum[lb=st]{1}{lb}
\cunum[st=lb]{1}{st}
\cunum[st=g]{1}{st}
\cunum[st=kg]{16}{st}
0.07 st
14 lb
6350.29 g
6.35 kg
0.16 st
101.6 kg
\end{verbatim}

\textbf{Note:} \hspace{1em} Of course using

\begin{verbatim}
cuaddtokeychain
{
  \{\texttt{st}\} \{1/14\} \%% 1lb = 1/14st
  \{\texttt{lb}\} \{1\} \%% unit already in a key-chain.
}
\end{verbatim}
is also possible

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

\[ T_{\text{Ro}} = T_C \times \frac{21}{40} + 7.5 \]

%%% defining new unit 'degree R\(\text{o}\)mer'
\newcookingunit \[\text{\ensuremath{ {} ^{\circ} }\text{R\(\text{o}\)}}\] \(\text{Ro}\)
\cuaddsinglekeys \{C\} %% adds value ‘Ro’ to key ‘C’.
\{\text{Ro}\} \{ \#1 \times 21/40 + 7.5 \}
\cusetoptionfor\{\text{Ro}\}\{ \text{round-precision = 0} \} %% round to integer automatically

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

7.1 Obsolete Commands

\cudefinekeys \cudefinekeys\{\text{\langle unit-key-1\rangle}\}
\{\langle unit-key-1\rangle \text{ are } \ldots \text{ unit-key-2}\}
\{\langle unit-key-1\rangle \text{ are } \ldots \text{ unit-key-3}\}
\{\langle unit-key-1\rangle \text{ are } \ldots \text{ unit-key-4}\}
\ldots
\}

This command is going to be obsolete at one point. It is advised to use \cudefinekeychain instead.

\cudefinekeys takes \{\text{\langle unit-key-1\rangle}\} as a “basis”, defines a key with the name \text{\langle unit-key-1\rangle} and adds the values \text{\langle unit-key-1\rangle}, \text{\langle unit-key-2\rangle}, \text{\langle unit-key-3\rangle}, etc. Furthermore this command also defines the keys \text{\langle unit-key-2\rangle}, \text{\langle unit-key-3\rangle}, etc. with the same values as \text{\langle unit-key-1\rangle}. Please note that \{\ldots\} has to be a number.

\cuaddkeys \cuaddkeys\{\text{\langle unit-key-1\rangle}\}
\{\langle unit-key-1\rangle \text{ are } \ldots \text{ unit-key-2}\}
\{\langle unit-key-1\rangle \text{ are } \ldots \text{ unit-key-3}\}
\{\langle unit-key-1\rangle \text{ are } \ldots \text{ unit-key-4}\}
\ldots
\}
\cuaddtokeys \cuaddtokeys\{\text{\langle unit-key-1\rangle}\} \{\text{\langle unit-key-2\rangle}\} \{\text{\langle unit-key-2\rangle\ are } \ldots \text{ unit-key-1}\}

Those commands are going to be obsolete at one point. It is advised to use \cuaddtokeychain instead.

\cuaddkeys takes the already defined key \text{\langle unit-key-1\rangle} as a “basis”, and adds \text{\langle unit-key-2\rangle}, \text{\langle unit-key-3\rangle}, etc. to its values. Furthermore it adds those new values to other keys linked to \text{\langle unit-key-1\rangle} and defines the new keys \text{\langle unit-key-2\rangle}, etc. with the same values as \text{\langle unit-key-1\rangle}.

Works similar to \cuaddkeys regarding the definition of keys.
8 Language support

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

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-phrase.

Note that cutext-range-phrase 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 problem is that the written-out “1” depends on the gender of the word following (e.g. “ein Baum” (m), “eine Pflanze” (f) and “ein Auto” (n)). Use those keys to set the specific gender of “1” (see also examples below).

\cundefinename\cundefinename{\langle Language\rangle}
{
  {\langle unit-key-1\rangle} {\langle symbol-1\rangle} {\langle singular-1\rangle} \{\langle plural-1\rangle\} <\langle gender\rangle>
  {\langle unit-key-2\rangle} {\langle symbol-2\rangle} {\langle singular-2\rangle} \{\langle plural-2\rangle\} <\langle gender\rangle>
  ...
}

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 \langle Language\rangle. For details regarding \langle language\rangle see the translations documentation.

If the plural form of the name differs from the singular form use \{\langle plural\rangle\} 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.

\langle gender\rangle can be m (maskulin), f (feminin) or n (neutrum). If not given, m is used as default.

\cundefinename C{English}
{
  \{kg\} \{kilogrammes\}
  \{oz\} \{ounce\}
  \{h\} \{hour\} \{hours\}
  \{C\} \{degree space Celsius\} \{degrees space Celsius\}
  \{decimal-marker\} {.} \{cutext-range-phrase\} {-to-}
  \{one\langle m\rangle\} \{one\}
  \{one\langle f\rangle\} \{one\}
  \{one\langle n\rangle\} \{one\}
}

\cundefinename C{German}
{
  \{kg\} \{Kilogramm\} <n>
\cundefinesymbol \cundefinesymbol{(Language)}
{
  \{\text{unit-key-1}\} \{\text{symbol-1}\}
  \{\text{unit-key-2}\} \{\text{symbol-2}\}
...
}

This command defines the symbols of the units printed in \cunum for the specific \langle Language \rangle. It works similar as \cundefinename, but only the symbols (and no names) can be set. For details regarding \langle Language \rangle see the translations documentation.

\cundefinesymbol \cundefinesymbol{English}
{
  \{\text{decimal-mark}\} \{.\}
  \{\text{cutext-range-phrase}\} \{-to-\}
  \{\text{one(m)}\} \{\text{one}\}
  \{\text{one(f)}\} \{\text{one}\}
  \{\text{one(n)}\} \{\text{one}\}
}

\cundefinesymbol \cundefinesymbol{German}
{
  \{\text{decimal-mark}\} \{,\}
  \{\text{cutext-range-phrase}\} \{-bis-\}
  \{\text{one(m)}\} \{\text{ein}\}
  \{\text{one(f)}\} \{\text{eine}\}
  \{\text{one(n)}\} \{\text{ein}\}
}

\cundefinesymbol \cundefinesymbol{French}
{
  \{1\} \{L\}
  \{dl\} \{dL\}
  \{cl\} \{cL\}
  \{ml\} \{mL\}
  \{\text{cutext-range-phrase}\} \{-'\{a\}-\}
  \{\text{decimal-mark}\} \{,\}
  \{\text{one(m)}\} \{\text{un}\}
  \{\text{one(f)}\} \{\text{une}\}
  \{\text{one(n)}\} \{\text{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:

\texttt{12ducks weed}
\texttt{\cunum{12}{dag} weed}
\texttt{3 ducks nuts}
\texttt{\cunum{3}{0}{dag} nuts}
\texttt{10ducks duckmeat}
\texttt{\cunum{10}{dag} duckmeat}

8.1 Phrases

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

\cudefinephrase{\langle Language\rangle}
{
    {\langle integer-1\rangle} \{\langle phrase-1\rangle\} \{\langle phrase-1-plural\rangle\} \langle gender-1\rangle
    {\langle integer-2\rangle} * \{\langle phrase-2\rangle\} \{\langle phrase-2-plural\rangle\} \langle gender-2\rangle
    ...
}

This command pairs for a given {\langle Language\rangle} (see package translations) the number {\langle integer-1\rangle} with {\langle phrase-1\rangle} (\& {\langle phrase-1-plural\rangle} and {\langle gender-1\rangle}). Afterwards the package can check if an 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 *.

\langle gender\rangle 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.

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!):
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.

9 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 9.1 which needs to be used as a package option.

\cusetup{⟨options⟩}

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

\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 Ré are by default rounded to integers.

75 °C \cunum{75}{C}
75 °F \cunum{75}{F}
75 K \cunum{75}{K}
75 ℃ \cunum{75}{Re}
80 °C \cunum{80}{C}
80 °F \cunum{80}{F}
80 K \cunum{80}{K}
80 ℃ \cunum{80}{Re}
75.23 °C \cunum{75.23}{C}
75.23 °F \cunum{75.23}{F}
75.23 K \cunum{75.23}{K}
75.23 ℃ \cunum{75.23}{Re}
9.1 Load time options

\usepackage[use-fmtcount-numerals=(true/false)]{cooking-units}

If set to true loads package fmtcount and uses \numberstringnum for \text and \Numberstringnum for \text{cut} to write-out numbers below \text{use-numerals-below} (13 by default), integers above are printed as numbers. You can decide to not print any numerals by setting \text{print-numerals} to false.

Note: You don’t need to use this function to use numerals. Using \text{print-numerals} and setting \text{numeral-function} and \text{Numeral-function} also works.

\begin{verbatim}
\begin{tabular}{l}
one kilogramme & \text{1}{kg} \\
One kilogramme & \text{1}{kg} \\
two kilogramme & \text{2}{kg} \\
Two kilogramme & \text{2}{kg} \\
twelve kilogramme & \text{12}{kg} \\
Twelve kilogramme & \text{12}{kg} \\
13 kilogramme & \text{13}{kg} \\
13 kilogramme & \text{13}{kg} \\
14 kilogramme & \text{14}{kg} \\
\end{tabular}
\end{verbatim}

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 \text{one(m)}, \text{one(f)} and \text{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 \text{numeral-function} and \text{Numeral-function}.

9.2 Normal options

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

9.2.1 Unit Specific options

\begin{verbatim}<unit> (unit-key-1) = (unit-key-2)
\end{verbatim}

Change \text{unit-key-1} to \text{unit-key-2} (see section 7 to define new options).
\textbf{\texttt{\textbackslash group}} \ (\texttt{\textbackslash group}) = \ (\texttt{\textbackslash unit-key})

Changes each unit contained in \texttt{\textbackslash group} to \texttt{\textbackslash unit-key} (\texttt{\textbackslash unit-key} must be part of \texttt{\textbackslash group}).

<table>
<thead>
<tr>
<th>\texttt{\textbackslash group}</th>
<th>default \texttt{\textbackslash unit-key}s</th>
</tr>
</thead>
<tbody>
<tr>
<td>weight</td>
<td>kg, dag, g, oz, lb, stick</td>
</tr>
<tr>
<td>length</td>
<td>m, dm, cm, mm, in</td>
</tr>
<tr>
<td>volume</td>
<td>l, dl, cl, ml</td>
</tr>
<tr>
<td>temperature</td>
<td>C, F, K, Re</td>
</tr>
<tr>
<td>energy</td>
<td>cal, kcal, J, kJ, eV</td>
</tr>
<tr>
<td>time</td>
<td>d, h, min, s</td>
</tr>
</tbody>
</table>

You can define new groups using \texttt{\textbackslash cuDeclareUnitGroup}:

\texttt{\textbackslash cuDeclareUnitGroup} \{\texttt{\textbackslash group-name}\} \{\texttt{\textbackslash unit-list}\}

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

\textbf{Example:} Define the group “weight”:

\texttt{\textbackslash cuDeclareUnitGroup} \{weight\} \{ kg , dag, g, oz, lb, stick \}

Now \texttt{\textbackslash cuSetup\{weight=dag\}} can be used to change all units contained in weight to dag.

\texttt{\textbackslash cuAddToUnitGroup} \{\texttt{\textbackslash group}\}\{\texttt{\textbackslash unit-list}\}

Adds \texttt{\textbackslash unit-list} to an already existing \texttt{\textbackslash group} (both need to exist).

\textbf{Example:} Adding \texttt{st} to the group \texttt{weight}

\texttt{\textbackslash cuAddToUnitGroup\{weight\}\{st\}}

\texttt{\textbackslash cuSetup\{weight=g\}}

\begin{verbatim}
1000 g  \cunum{1}{kg}\\n10 g   \cunum{1}{dag}\\n1 g    \cunum{1}{g}\\n28.35 g \cunum{1}{oz}\\n453.59 g\cunum{1}{lb}\\n113.4 g\cunum{1}{stick}\\n6350.29 g\cunum{1}{st}
\end{verbatim}
add-unit-to-group =
{
  ⟨group1⟩ = {⟨unit-key-list⟩},
  ⟨group2⟩ = {⟨unit-key-list⟩},
  ...
}

This option is going to be obsolete at one point. Adds each ⟨unit-key⟩ in ⟨unit-keys-list⟩ to ⟨group⟩. The key-val equivalent of \cuaddtounitgroup.

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

This option is going to be obsolete at one point. 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.

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

This option is going to be obsolete at one point. 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.

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

This option is going to be obsolete at one point. 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).

9.2.2 Command behavior

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 \texttt{\textbackslash cutext\{1\}-\{kg\}\}\\ 
2 kilogramme \texttt{\textbackslash Cutext\{2\}-\{kg\}\}\\ 
⅓ kilogramme \texttt{\textbackslash cutext\{1/2\}-\{kg\}\}\\ 
½ kilogramme \texttt{\textbackslash cutext\{?\}-\{kg\}\}\\ 
1000 to 2000 gramme \texttt{\textbackslash cutext\{kg=g\}\{1--2\}-\{kg\}\}\\ 
\texttt{\textbackslash cusetup\{cutext-to-cunum=true\}\}\ 
1 kg \texttt{\textbackslash cutext\{1\}-\{kg\}\}\ 
2 kg \texttt{\textbackslash Cutext\{2\}-\{kg\}\}\ 
⅓ kg \texttt{\textbackslash cutext\{1/2\}-\{kg\}\}\ 
½ kg \texttt{\textbackslash cutext\{?\}-\{kg\}\}\ 
1000–2000 g \texttt{\textbackslash cutext\{kg=g\}\{1--2\}-\{kg\}}

\texttt{\textbackslash cutext-change-unit}\cutext-change-unit = \langle\texttt{true/false}\rangle

Set this option to \texttt{false} if you do \texttt{not} want the units of \texttt{\textbackslash cutext} and \texttt{\textbackslash Cutext} to be changed. Set to \texttt{true} by default.

\texttt{\textbackslash cutext-change-unit}\cutext-change-unit = \langle\texttt{true/false}\rangle

\texttt{\textbackslash cuam-version}\cuam-version = \langle\texttt{old/new}\rangle

\texttt{\textbackslash cutext-version}\cutext-version = \langle\texttt{old/new}\rangle

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

9.2.3 Hooks

\texttt{\textbackslash commands-add-hook}\commands-add-hook = \langle\texttt{\langle\texttt{code}\rangle}\rangle

\texttt{\textbackslash cunum-add-hook}\cunum-add-hook = \langle\texttt{\langle\texttt{code}\rangle}\rangle

\texttt{\textbackslash cutext-add-hook}\cutext-add-hook = \langle\texttt{\langle\texttt{code}\rangle}\rangle

\texttt{\textbackslash Cutext-add-hook}\Cutext-add-hook = \langle\texttt{\langle\texttt{code}\rangle}\rangle

\texttt{\textbackslash cuam-add-hook}\cuam-add-hook = \langle\texttt{\langle\texttt{code}\rangle}\rangle

Adds \texttt{\langle\texttt{code}\rangle} to the respective command (or in case of the first key: to \texttt{all commands}). The hook is executed \texttt{after} setting the keys, but \texttt{before} parsing and processing the input.

Please be careful with spaces, they will be printed.

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

\texttt{\newcounter\{CookingUnitsCounter\} \% or however you like it} \texttt{\newcounter\{CookingUnitsCounter\} \% or however you like it} \texttt{\cusetup\{commands-add-hook=\{\textbackslash stepcounter\{CookingUnitsCounter\}\}\\}}

\texttt{\%\% 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):

<table>
<thead>
<tr>
<th>command</th>
<th>times</th>
</tr>
</thead>
<tbody>
<tr>
<td>\cunum</td>
<td>214</td>
</tr>
<tr>
<td>\cutext</td>
<td>63</td>
</tr>
<tr>
<td>\Cutext</td>
<td>29</td>
</tr>
<tr>
<td>\cuam</td>
<td>65</td>
</tr>
<tr>
<td>total</td>
<td>371</td>
</tr>
</tbody>
</table>

### 9.2.4 Input and Outputs

**expand-both** =\(\langle n/o/f/x\rangle\)

**expand-amount** =\(\langle n/o/f/x\rangle\)

**expand-unit** =\(\langle n/o/f/x\rangle\)

By default the commands \cunum, \cutext and \Cutext do not expand their input. You can change the expansion behavior of \(\langle\text{amount}\rangle\) and/or \(\langle\text{unit-key}\rangle\) using the options specified above. The meaning of the available values are the same as specified in the \LaTeX\ document “interface3”.

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

**set-special-sign** = \{\langle\text{character(s)}\rangle\}

**add-special-sign** = \{\langle\text{character(s)}\rangle\}

Allows \(\langle\text{character(s)}\rangle\) 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.

\[\begin{align*}
?\text{kg} & \quad \cunum{?}\{\text{kg}\}\backslash
10?--20?\text{kg} & \quad \cunum[g=\text{kg}]{10?--20?}\{\text{kg}\}\backslash
\text{cusetup}\{\text{add-special-sign}={\text{xX}}\}\backslash
\text{x}\text{kg} & \quad \cunum(x)\{\text{kg}\}\backslash
X--?\text{kg} & \quad \cunum(X--?)\{\text{kg}\}\backslash
\text{cusetup}\{\text{set-special-sign}={}\}\backslash
\text{1kg} & \quad \cunum\{1\}\{\text{kg}\}\backslash
\text{1--2kg} & \quad \cunum\{1--2\}\{\text{kg}\}\end{align*}\]

**set-unknown-message** = \{\langle\text{error/warning/none}\rangle\}

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

**set-cutext-translation-message** = \{\langle\text{error/warning/none}\rangle\}

If a translation for \cutext and \Cutext is not available for the language, 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 than `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`:

```latex
one kilogramme \text{(\texttt{1}kg)}

two kilogramme \text{(\texttt{2}kg)}

twelve kilogramme \text{(\texttt{12}kg)}

13 kilogramme \text{(\texttt{13}kg)}

\text{\texttt{\textbackslash cusetup\{print-numerals=false\}}}

1 kilogramme \text{(\texttt{1}kg)}

2 kilogramme \text{(\texttt{2}kg)}

12 kilogramme \text{(\texttt{12}kg)}

13 kilogramme \text{(\texttt{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`.

```latex
one kilogramme \text{(\texttt{1}kg)}

two kilogramme \text{(\texttt{2}kg)}

twelve kilogramme \text{(\texttt{12}kg)}

13 kilogramme \text{(\texttt{13}kg)}

\text{\texttt{\textbackslash cusetup\{use-numerals-below=10\}}}

one kilogramme \text{(\texttt{1}kg)}

two kilogramme \text{(\texttt{2}kg)}

twelve kilogramme \text{(\texttt{12}kg)}

13 kilogramme \text{(\texttt{13}kg)}

\text{\texttt{\textbackslash cusetup\{use-numerals-below=0\}}}

1 kilogramme \text{(\texttt{1}kg)}

2 kilogramme \text{(\texttt{2}kg)}

12 kilogramme \text{(\texttt{12}kg)}

13 kilogramme \text{(\texttt{13}kg)}

\text{\texttt{\textbackslash cusetup\{use-numerals-below=12001\}}}

one thousand gramme \text{(\texttt{1}kg)}

two thousand gramme \text{(\texttt{2}kg)}

twelve thousand gramme \text{(\texttt{12}kg)}

13000 gramme \text{(\texttt{13}kg)}
```
**numeral-function**

\[ \text{numeral-function} = \langle \text{function} \rangle \]

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

\[
\cusetup{
    \text{numeral-function} = \text{\numberstringnum} ,
    \text{Numeral-function} = \text{\Numberstringnum}
}
\]

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

**parse-number**

\[ \text{parse-number} = \langle \text{true/false} \rangle \]

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}
\cunum{[kg=g]{1}}{kg}\
\cunum{1--2}{kg}\
\cunum{1----------2}{kg}\
\cunum{1.2}{kg}\
\cunum{1/2}{kg}\
\cunum{1_2/3}{kg}\
\cunum{1/2_3}{kg}\
\cunum{someweirdstuff}{kg}\
\cutext{1}{kg}\
\cutext{100}{kg}\
\cutext{gjfak}{kg}\
\cutext{12}{kg}\
\cuam{1--------2}\
\cuam{1,2}\
\cuam{1_1/2}\
\cuam{kwflk}\
```

**range-sign**

\[ \text{range-sign} = \langle \text{string} \rangle \]

**cunum-range-sign**

\[ \text{cunum-range-sign} = \langle \text{string} \rangle \]

**cutext-range-sign**

\[ \text{cutext-range-sign} = \langle \text{string} \rangle \]

\text{cunum-range-sign} sets the printed range sign used in `\cunum` (and `\cuam`) to \langle string \rangle, \text{cutext-range-sign} sets the printed range sign used in `\cutext` and `\Cutext` to \langle string \rangle. Using \text{range-sign} sets the range signs for both `\cunum` (and `\cuam`) and \text{cutext}/\Cutext to \langle string \rangle.

The default for \langle string \rangle is -- (for both).

Since version 1.45 there also exists the language symbol \text{cutext-range-sign} (see section 8). If the option `cutext-range-sign` is set the language symbol will be ignored.
use-phrases = \langle true/false \rangle

Setting this option to true replaces certain integers (see section 8.1 for more information) with their phrase counterpart. This option is set to false by default.

Example: For the German language:

\selectlanguage{ngerman}
\begin{verbatim}
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 bis zwei Dutzend \cuam{12--24}\
drei Dutzend \cuam{36}\
\end{verbatim}

9.2.5 Rounding options

round-precision = \langle integer \rangle

Rounds the amount automatically to \langle integer \rangle digits after the colon. Note that units like C, F, K and Re are still rounded to integers due to \texttt{\cusetoptionfor}.
1.23457 kg
0.01259 kg
194 kg
392–410 °F
−273 °C

csetup{round-precision=5}
\cunum{1.23456789}{kg}\n\cunum{12.587}{g}\n\cunum{194}{kg}\n\cunum{200--210}{C}\n\cunum{0.0012}{K}

cunum{1.23456789}{kg}\n\cunum{12.58}{kg}\n\cunum{194}{g}\n\cunum{200--210}{C}\n\cunum{0.0012}{K}

\cunum{-271,2}{C}\n\cunum{0.0012}{K}\n\cunum{185}{C}\n\cunum{180--200}{C}

\cunum{0.005}{kg}\n\cunum{-0.005}{kg}\n\cunum{1.245}{kg}\n\cunum{1.245}{kg}

\cunum{0.005}{kg}\n\cunum{-0.005}{kg}\n\cunum{1.245}{kg}\n\cunum{1.245}{kg}

\cunum{0.005}{kg}\n\cunum{-0.005}{kg}\n\cunum{1.245}{kg}\n
Note: Negative numbers are also allowed.

\cusetoptionfor{C,F}{round-precision=-1}

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.

\cusetup{round-half=default}
0 kg \cunum{0.005}{kg}\n−0 kg \cunum{-0.005}{kg}\n1.24 kg \cunum{1.245}{kg}\n\cunum{0.005}{kg}\n\cunum{-0.005}{kg}\n\cunum{1.245}{kg}

9.2.6 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.
\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}\\
−1500 g
\cunum[kg=g]{1_?/3}{kg}\\
1½ kg
\cunum[kg=g]{1_2/?}{kg}\\

convert-fraction \hspace{1cm} 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).

\cusetup{convert-fraction=true}

½ kg
\cunum{1/3}{kg}\\
333.33 g
\cunum[kg=g]{1/3}{kg}\\
1⅛ kg
\cunum{1_1/2}{kg}\\
1500 g
\cunum[kg=g]{1_1/2}{kg}\\
1⅛ kg
\cunum[kg=g]{1_?/3}{kg}\\

fraction-command \hspace{1cm} 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}\\
12/3 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}\\
1½ kg
\cunum{1_2/3}{kg}\\

fraction-inline \hspace{1cm} fraction-inline = {\langle input containing #1 and #2\rangle}

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

9.2.7 Spaces

\begin{verbatim}
mixed-fraction-space \texttt{mixed-fraction-space = \{length\}}
\end{verbatim}

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

\begin{verbatim}
1\frac{1}{3}
\cuam{1_2/3}
\cunum{1_2/3}{kg}
\cusetup{mixed-fraction-space=1em}
\cuam{1_2/3}
\cunum{1_2/3}{kg}
\cusetup{mixed-fraction-space=0em}
\cuam{1_2/3}
\cunum{1_2/3}{kg}
\cusetup{mixed-fraction-space=qwe}
\cuam{1_2/3}
\cunum{1_2/3}{kg}
\end{verbatim}

\begin{verbatim}
cutext-space \texttt{cutext-space = \{\texttt{string}\}}
\end{verbatim}

\texttt{\{} \texttt{string} \texttt{\}} is inserted between the numeral part and the unit part when using \texttt{\cutext} and \texttt{\Cutext}. By default it is set an unbreakable space \texttt{-}.

\begin{verbatim}
1 kilogramme \cutext{1}{kg}
10 kilogramme \Cutext{10}{kg}
\cusetup{cutext-space=\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={qwe}}
1 kilogramme \cutext{1}{kg}
10 kilogramme \Cutext{10}{kg}
\end{verbatim}
phrase-space phrase-space = \{\langle\text{string}\rangle\}

\langle\text{string}\rangle\text{ is inserted between the numeral part and the phrase part while using } \texttt{\cuam}. \text{ By default it is set to the unbreakable space \texttt{-}. Use this option if you want to e.g. insert a normal space.}

(Switching to german)

\begin{verbatim}
1 Dutzend \cuam{12}\{}\ \\
12 Dutzend \cuam{144}\{}\ \\
\cusetup{phrase-space=\texttt{\{}\}
1 Dutzend \cuam{12}\{}\ \\
12 Dutzend \cuam{144}\{}\ \\
\cusetup{phrase-space=\texttt{}}
1Dutzend \cuam{12}\{}\ \\
12Dutzend \cuam{144}\{}\ \\
\cusetup{phrase-space=\texttt{qwe}}
1qweDutzend \cuam{12}\{}\ \\
12qweDutzend \cuam{144}\{}
\end{verbatim}

amount-unit-space amount-unit-space = \{\langle\text{string}\rangle\}

Change the spacing for \texttt{\cunum} between the printed amount(s) and the unit. The default value is \texttt{\thsinspace}.

\begin{verbatim}
1 kg \cunum{1}{kg}\{}
1/2 kg \cunum{1/2}{kg}\{}
1–2 kg \cunum{1--2}{kg}\{}
\cusetup{amount-unit-space=\texttt{\{}\texttt{hspace}{1em}}}\{}
1 kg \cunum{1}{kg}\{}
1/2 kg \cunum{1/2}{kg}\{}
1–2 kg \cunum{1--2}{kg}\{}
\cusetup{amount-unit-space=\texttt{}}\{}
1 kg \cunum{1}{kg}\{}
1/2 kg \cunum{1/2}{kg}\{}
1–2 kg \cunum{1--2}{kg}\{}
\cusetup{amount-unit-space=\texttt{qwe}}\{}
1 qwe kg \cunum{1}{kg}\{}
1/2 qwe kg \cunum{1/2}{kg}\{}
1–2 qwe kg \cunum{1--2}{kg}\{}
\end{verbatim}

9.2.8 label & refs

recalculate-amount recalculate-amount = \{true/false\}

Set this option to \texttt{true} if you want to change your recipes to the given number of people set by \texttt{set-number-of-persons}. Note that only those values who have a label are changed.
**set-number-of-persons**  
\[ \text{set-number-of-persons = \langle integer \rangle} \]

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 \langle label \rangle given. It is set to 4 by default. Please also note the use of **recalculate-amount**.

\[
\begin{align*}
\text{2 persons} & & \text{\texttt{\textbackslash curef\{anotherrecipe\}-persons}} \\
\text{1 kg} & & \text{\texttt{\textbackslash cuam\{anotherrecipe\}\{1\}\{kg\}}} \\
\text{10 kilogramme} & & \text{\texttt{\textbackslash cutext[ref=anotherrecipe]\{10\}\{kg\}}} \\
\text{4 persons} & & \text{\texttt{\textbackslash curef\{anotherrecipe\}-persons}} \\
\text{2 kg} & & \text{\texttt{\textbackslash cuam\{anotherrecipe\}\{1\}\{kg\}}} \\
\text{20 kilogramme} & & \text{\texttt{\textbackslash cutext[ref=anotherrecipe]\{20\}\{kg\}}} \\
\text{3 persons} & & \text{\texttt{\textbackslash curef\{anotherrecipe\}-persons}} \\
\text{1.5 kg} & & \text{\texttt{\textbackslash cuam\{anotherrecipe\}\{1\}\{kg\} \textbackslash cuam\{anotherrecipe\}\{1\}}} \\
\text{15 kilogramme} & & \text{\texttt{\textbackslash cutext[ref=anotherrecipe]\{15\}\{kg\}}} \\
\text{2 persons} & & \text{\texttt{\textbackslash curef\{anotherrecipe\}-persons}} \\
\text{1 kg} & & \text{\texttt{\textbackslash cuam\{anotherrecipe\}\{1\}\{kg\} \textbackslash cuam\{anotherrecipe\}\{1\}}} \\
\text{10 kilogramme} & & \text{\texttt{\textbackslash cutext[ref=anotherrecipe]\{10\}\{kg\}}} \\
\text{1 persons} & & \text{\texttt{\textbackslash curef\{anotherrecipe\}-persons}} \\
\text{0.5 kg} & & \text{\texttt{\textbackslash cuam\{anotherrecipe\}\{1\}\{kg\} \textbackslash cuam\{anotherrecipe\}\{1\}}} \\
\text{5 kilogramme} & & \text{\texttt{\textbackslash cutext[ref=anotherrecipe]\{5\}\{kg\}}} \\
\end{align*}
\]

**label**  
\[ \text{label = \langle\langle string \rangle\rangle*\langle integer \rangle} \]

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

\[
\begin{align*}
\text{1 person} & & \text{\texttt{\textbackslash curef\{Toast\}-person}} \\
\text{2} & & \text{\texttt{\textbackslash cuam\{Toast\}\{2\}}} \\
\text{2 dag} & & \text{\texttt{\textbackslash cuam\{Toast\}\{2\}\{dag\}}} \\
\text{4 persons} & & \text{\texttt{\textbackslash cuam\{Toast\}-persons}} \\
\text{8} & & \text{\texttt{\textbackslash cuam\{Toast\}\{8\}}} \\
\text{8 dag} & & \text{\texttt{\textbackslash cuam\{Toast\}\{8\}\{dag\}}} \\
\end{align*}
\]

**get-label**  
\[ \text{get-label = \langle\langle label \rangle\rangle} \]

The key-value version of \texttt{\textbackslash curef}. Note that this key doesn’t save the value inside a macro but rather prints it directly into the document.
\cuelabel{Schinken}{3}
3 \curef{Schinken}\
3 \cusetup{get-label=Schinken}\
\curef{Schinken}\
\cusetup{recalculate-amount=true} 
4 \curef{Schinken} \\
4 \curef{Schinken}

\textbf{Note:} \curef is expendable.

\texttt{ref} \texttt{ref} = \texttt{⟨/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 \texttt{⟨/label⟩} is not defined.

\cuelabel{Kaese}{3}
10 dm \cunum{Kaese}[m=dm]{1}{m}\
10 dm \cunum{Kaese,m=dm}{1}{m}\
\cusetup{recalculate-amount=true} 
13.33 dm \cunum{Kaese}[m=dm]{1}{m}\
13.33 dm \cunum{Kaese,m=dm}{1}{m}

\texttt{curef-add-forbidden-unit \ curef-add-forbidden-unit} = \texttt{⟨unit list⟩}
\texttt{curef-remove-forbidden-unit \ curef-remove-forbidden-unit} = \texttt{⟨unit list⟩}
\texttt{curef-clear-forbidden-units} = \texttt{(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.

\cuelabel{check}{2}
\cusetup{recalculate-amount=true} 
2 m \cunum{check}{1}{m}\
2 kg \cunum{check}{1}{kg}\
1 °C \cunum{check}{1}{C}\
\cusetup{curef-add-forbidden-unit={m,kg}} 
1 m \cunum{check}[m=1\{m\}]{1}{m}\
1 kg \cunum{check}[m=1\{kg\}]{1}{kg}\
1 °C \cunum{check}[m=1\{C\}]{1}{C}\
\cusetup{curef-remove-forbidden-unit={C}} 
1 m \cunum{check}[m=1\{m\}]{1}{m}\
1 kg \cunum{check}[m=1\{kg\}]{1}{kg}\
2 °C \cunum{check}[m=1\{C\}]{1}{C}\
\cusetup{curef-clear-forbidden-units=true} 
2 m \cunum{check}[m=1\{m\}]{1}{m}\
2 kg \cunum{check}[m=1\{kg\}]{1}{kg}\
2 °C \cunum{check}[m=1\{C\}]{1}{C}
9.3 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 \texttt{\textbackslash{cunum}(0)}{K} is ok, \texttt{\textbackslash{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 =

\begin{verbatim}
{ 
  \langle unit-key-1 \rangle = \langle minimum-value-1 \rangle , 
  \langle unit-key-2 \rangle = \langle minimum-value-2 \rangle , 
  \ldots 
}
\end{verbatim}

This option adds \langle unit-key-1 \rangle 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 \langle unit-key \rangle = \langle minimum-value \rangle. This sets the allowed minimum value of \langle unit-key \rangle to \langle minimum-value \rangle.

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:

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

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

\begin{verbatim}
\cusetup
{ 
  add-temperature-to-check = { Ro = -135.90375 } 
}
\end{verbatim}

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).
add-natural-unit \ add-natural-unit = \mathbf{\langle\ unit-key\rangle}

This option adds \mathbf{\langle\ unit-key\rangle} to the list of units \texttt{convert-to-eV} uses to determine how a unit is transformed if set to \texttt{true}.

\begin{itemize}
  \item \texttt{42 = \langle\ true/false\rangle}
  \begin{itemize}
    \item Take a good guess.
  \end{itemize}
\end{itemize}

\begin{itemize}
  \item \texttt{42 kg}
  \begin{itemize}
    \item \texttt{\langle\ true/false\rangle}
  \end{itemize}
\end{itemize}

\begin{itemize}
  \item \texttt{42 g}
  \begin{itemize}
    \item \texttt{\langle\ true/false\rangle}
  \end{itemize}
\end{itemize}

\begin{itemize}
  \item \texttt{42 J}
  \begin{itemize}
    \item \texttt{\langle\ true/false\rangle}
  \end{itemize}
\end{itemize}

\begin{itemize}
  \item \texttt{42 °C}
  \begin{itemize}
    \item \texttt{\langle\ true/false\rangle}
  \end{itemize}
\end{itemize}

\begin{itemize}
  \item \texttt{42 s}
  \begin{itemize}
    \item \texttt{\langle\ true/false\rangle}
  \end{itemize}
\end{itemize}

\begin{itemize}
  \item \texttt{42 - 42 min}
  \begin{itemize}
    \item \texttt{\langle\ true/false\rangle}
  \end{itemize}
\end{itemize}

\begin{itemize}
  \item \texttt{42 × 42 × 42 min}
  \begin{itemize}
    \item \texttt{\langle\ true/false\rangle}
  \end{itemize}
\end{itemize}

\begin{itemize}
  \item \texttt{42 \ell}
  \begin{itemize}
    \item \texttt{\langle\ true/false\rangle}
  \end{itemize}
\end{itemize}

nothing-special \ nothing-special = \langle\ true/false\rangle
\\
going-bonkers \ going-bonkers = \langle\ true/false\rangle
\\
fully-bonkers \ fully-bonkers = \langle\ true/false\rangle
\\
xD-lol \ xD-lol = \langle\ true/false\rangle

Options that do \ldots stuff. The four stages of madness in option for.

nothing-special \ is your default. The package behaves as intended.

going-bonkers \ is a bit more strange. It converts an unit into another random unit (if it can) and does so throughout the document. So if \texttt{unit-A} is converted into \texttt{unit-B}, it is going to be converted this way the entire document through. For an unit to be converted it must have a key, see section 7.

fully-bonkers \ converts one unit into another random unit (if it can) and does so for each unit it encounters. So \texttt{unit-A} might be converted into \texttt{unit-B} the first time, but \texttt{unit-C} the second. Each conversion picks a random unit for the conversion (but the conversion itself makes sense, e.g. \texttt{kg} into \texttt{g}, but not into \texttt{cm}).

xD-lol \ is pure insanity. A unit is transformed into another, if it makes sense or not, and its value is replaced by a random number.
10 Public Commands

This section describes some public functions. Their main usage is the printing of stuff. They are primitives used between an \ExplSyntaxOn and \ExplSyntaxOff; for usage in a document you may do the following:

\ExplSyntaxOn
\NewDocumentCommand \cuprintfrac { O{} m m } { \cookingunits_print_fraction:nnn {#1} {#2} {#3} }
\NewDocumentCommand \cuprintrange { m m } { \cookingunits_print_range:nn {#1} {#2} }
\ExplSyntaxOff

\cookingunits_print_fraction:nnn \cookingunits_print_range:nnn

\langle \text{mixed-part} \rangle \langle \text{numerator} \rangle \langle \text{denominator} \rangle

Note: There is no parsing and processing done in this command, it just prints the input.

Uses the internal fraction printing command to print a fraction. \langle \text{mixed-part} \rangle can be empty if no mixed part is needed.

\cookingunits_print_range:nn \cookingunits_print_range:nn

\langle \text{left-part} \rangle \langle \text{right-part} \rangle

Note: There is no parsing and processing done in this command, it just prints the input.

Uses the internal range printing command to print a range (using the same range sign as \cunum would use)

11 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) 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.
12 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\textsuperscript{sicilian} (Rottel\textsuperscript{de}) Around 0.850 kg

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

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

\textit{"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)?

\textit{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 Réamur [°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* and – even worse – the weight changes depending the content (See Packerl).

Packerl* (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:

<table>
<thead>
<tr>
<th>Packerl Backpulver (baking powder)</th>
<th>16 g (0.56 oz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Natrium</td>
<td>14 g (0.49 oz)</td>
</tr>
<tr>
<td>Vanillin(-zucker) (vanillin(-sugar))</td>
<td>8 g (0.28 oz)</td>
</tr>
<tr>
<td>Germ*</td>
<td>7 g (0.25 oz)</td>
</tr>
</tbody>
</table>

*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.

*And maybe doesn’t even exist outside my family
### A.1 English

<table>
<thead>
<tr>
<th>Unit-key</th>
<th>Printed Unit</th>
<th>Unitname</th>
<th>(plural)</th>
<th>Gender</th>
</tr>
</thead>
<tbody>
<tr>
<td>kg</td>
<td>kg</td>
<td>kilogramme</td>
<td></td>
<td>m</td>
</tr>
<tr>
<td>dag</td>
<td>dag</td>
<td>decagramme</td>
<td></td>
<td>m</td>
</tr>
<tr>
<td>g</td>
<td>g</td>
<td>gramme</td>
<td></td>
<td>m</td>
</tr>
<tr>
<td>oz</td>
<td>oz</td>
<td>ounce</td>
<td></td>
<td>m</td>
</tr>
<tr>
<td>lb</td>
<td>lb</td>
<td>pound</td>
<td>(pounds)</td>
<td>m</td>
</tr>
<tr>
<td>°C</td>
<td>°C</td>
<td>degree Celsius</td>
<td>(degrees Celsius)</td>
<td>m</td>
</tr>
<tr>
<td>°F</td>
<td>°F</td>
<td>degree Fahrenheit</td>
<td>(degrees Fahrenheit)</td>
<td>m</td>
</tr>
<tr>
<td>°Ré</td>
<td>°Ré</td>
<td>degree Réaumur</td>
<td>(degrees Réaumur)</td>
<td>m</td>
</tr>
<tr>
<td>K</td>
<td>K</td>
<td>kelvin</td>
<td></td>
<td>m</td>
</tr>
<tr>
<td>d</td>
<td>d</td>
<td>day</td>
<td>(days)</td>
<td>m</td>
</tr>
<tr>
<td>h</td>
<td>h</td>
<td>hour</td>
<td>(hours)</td>
<td>m</td>
</tr>
<tr>
<td>min</td>
<td>min</td>
<td>minute</td>
<td>(minutes)</td>
<td>m</td>
</tr>
<tr>
<td>s</td>
<td>s</td>
<td>second</td>
<td>(seconds)</td>
<td>m</td>
</tr>
<tr>
<td>m</td>
<td>m</td>
<td>metre</td>
<td>(metres)</td>
<td>m</td>
</tr>
<tr>
<td>dm</td>
<td>dm</td>
<td>decimetre</td>
<td>(decimetres)</td>
<td>m</td>
</tr>
<tr>
<td>cm</td>
<td>cm</td>
<td>centimetre</td>
<td>(centimetres)</td>
<td>m</td>
</tr>
<tr>
<td>mm</td>
<td>mm</td>
<td>millimetre</td>
<td>(millimetres)</td>
<td>m</td>
</tr>
<tr>
<td>in</td>
<td>in</td>
<td>inch</td>
<td>(inches)</td>
<td>m</td>
</tr>
<tr>
<td>ℓ</td>
<td>ℓ</td>
<td>litre</td>
<td>(litres)</td>
<td>m</td>
</tr>
<tr>
<td>dl</td>
<td>dl</td>
<td>decilitre</td>
<td>(decilitres)</td>
<td>m</td>
</tr>
<tr>
<td>cl</td>
<td>cl</td>
<td>centilitre</td>
<td>(centilitres)</td>
<td>m</td>
</tr>
<tr>
<td>ml</td>
<td>ml</td>
<td>millilitre</td>
<td>(millilitres)</td>
<td>m</td>
</tr>
<tr>
<td>cal</td>
<td>cal</td>
<td>calorie</td>
<td>(calories)</td>
<td>m</td>
</tr>
<tr>
<td>kcal</td>
<td>kcal</td>
<td>kilocalorie</td>
<td>(kilocalories)</td>
<td>m</td>
</tr>
<tr>
<td>J</td>
<td>J</td>
<td>joule</td>
<td>(joules)</td>
<td>m</td>
</tr>
<tr>
<td>kJ</td>
<td>kJ</td>
<td>kilojoule</td>
<td>(kilojoules)</td>
<td>m</td>
</tr>
<tr>
<td>eV</td>
<td>eV</td>
<td>electron volt</td>
<td></td>
<td>m</td>
</tr>
<tr>
<td>pn</td>
<td>pinch</td>
<td>pinch</td>
<td>(pinches)</td>
<td>m</td>
</tr>
<tr>
<td>EL</td>
<td>tbsp.</td>
<td>tablespoon</td>
<td>(tablespoons)</td>
<td>m</td>
</tr>
<tr>
<td>TL</td>
<td>tsp.</td>
<td>teaspoon</td>
<td>(teaspoons)</td>
<td>m</td>
</tr>
<tr>
<td>csp</td>
<td>csp.</td>
<td>coffeespoonful</td>
<td></td>
<td>m</td>
</tr>
<tr>
<td>dsp</td>
<td>dsp.</td>
<td>dessertspoonful</td>
<td></td>
<td>m</td>
</tr>
<tr>
<td>ssp</td>
<td>ssp.</td>
<td>saltspoonful</td>
<td></td>
<td>m</td>
</tr>
<tr>
<td>Msp</td>
<td>Msp.</td>
<td>—</td>
<td></td>
<td>m</td>
</tr>
<tr>
<td>decimal-mark</td>
<td>—</td>
<td>.</td>
<td></td>
<td>m</td>
</tr>
<tr>
<td>one(m)</td>
<td>—</td>
<td>one</td>
<td></td>
<td>m</td>
</tr>
<tr>
<td>one(f)</td>
<td>—</td>
<td>one</td>
<td></td>
<td>m</td>
</tr>
<tr>
<td>one(n)</td>
<td>—</td>
<td>one</td>
<td></td>
<td>m</td>
</tr>
</tbody>
</table>
## A.2 american

<table>
<thead>
<tr>
<th>(unit-key)</th>
<th>printed unit</th>
<th>unitname</th>
<th>(plural)</th>
<th>gender</th>
</tr>
</thead>
<tbody>
<tr>
<td>kg</td>
<td>kg</td>
<td>kilogram</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>dag</td>
<td>dag</td>
<td>decagram</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>g</td>
<td>g</td>
<td>gram</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>oz</td>
<td>oz</td>
<td>ounce</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>lb</td>
<td>lb</td>
<td>pound</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>°C</td>
<td>degree Celsius</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>°F</td>
<td>degree Fahrenheit</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>Re</td>
<td>°Ré</td>
<td>degree Réaumur</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>K</td>
<td>K</td>
<td>kelvin</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>d</td>
<td>day</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>h</td>
<td>h</td>
<td>hour</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>min</td>
<td>min</td>
<td>minute</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>s</td>
<td>s</td>
<td>second</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>m</td>
<td>m</td>
<td>meter</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>dm</td>
<td>dm</td>
<td>decimeter</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>cm</td>
<td>cm</td>
<td>centimeter</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>mm</td>
<td>mm</td>
<td>millimeter</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>in</td>
<td>in</td>
<td>inch</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>ℓ</td>
<td>ℓ</td>
<td>liter</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>dl</td>
<td>dl</td>
<td>deciliter</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>cl</td>
<td>cl</td>
<td>centiliter</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>ml</td>
<td>ml</td>
<td>milliliter</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>cal</td>
<td>cal</td>
<td>calorie</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>kcal</td>
<td>kcal</td>
<td>kilocalorie</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>J</td>
<td>J</td>
<td>joule</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>kJ</td>
<td>kJ</td>
<td>kilojoule</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>eV</td>
<td>eV</td>
<td>electron volt</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>pn</td>
<td>pn.</td>
<td>pinch</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>EL</td>
<td>tbsp.</td>
<td>tablespoon</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>TL</td>
<td>tsp.</td>
<td>teaspoon</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>csp</td>
<td>csp.</td>
<td>coffeespoonful</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>dsp</td>
<td>dsp.</td>
<td>dessertspoonful</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>ssp</td>
<td>ssp.</td>
<td>saltspoonful</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>Msp</td>
<td>Msp.</td>
<td>—</td>
<td>m</td>
<td></td>
</tr>
</tbody>
</table>

| decimal-mark | — | | m |
| one(m)       | — | one | m |
| one(f)       | — | one | m |
| one(n)       | — | one | m |
## A.3 German

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

<table>
<thead>
<tr>
<th>(Phrase-key)</th>
<th>phrase</th>
<th>(plural)</th>
<th>gender</th>
</tr>
</thead>
<tbody>
<tr>
<td>12’</td>
<td>Dutzend</td>
<td></td>
<td>n</td>
</tr>
</tbody>
</table>
Some further phrases, just to write them down (they are not implemented, as they are barely used).

<table>
<thead>
<tr>
<th>(number)</th>
<th>name</th>
<th>Note</th>
<th>(plural)</th>
<th>gender</th>
</tr>
</thead>
<tbody>
<tr>
<td>60</td>
<td>Schock</td>
<td>(5 Dutzend, 12 * 5)</td>
<td>n</td>
<td></td>
</tr>
<tr>
<td>144</td>
<td>Gros</td>
<td>(12 Dutzend, 12 * 12)</td>
<td>n</td>
<td></td>
</tr>
<tr>
<td>1728</td>
<td>Großgros</td>
<td>(12 Groß, 12 * 144)</td>
<td>n</td>
<td></td>
</tr>
</tbody>
</table>

Note that Großgros has other (probably more common) synonyms.
### A.4 French

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

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 m (exact) |
| 1 yard = 3 foot |
| 1 yard = 36 Inch |
| 1 Inch = 0.0254 m (also exact) |

| 1 liter = 1 dm³ |
| 1 gallon = 4.54609 liter (exact) |
| 1 gallon = 4 Quart |
| 1 gallon = 8 Pint |
| 1 gallon = 32 Gill |
| 1 gallon = 160 fl. oz |
| 1 fl. oz = 0.0284130625 liter |

Note 1: I think the American fl. oz 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 fl. oz 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 “flous”, “floiz” and “floez”? “flous” would be “fl. oz U.S.”, “floiz” would be “Imperial fl. oz” and “floez” would simply be equal to 30 ml? |

For “stick” see [6].

| 1 lb = 0.45359237 kg (exact) |
| 1 lb = 16 oz |
| 1 lb = ½ st |
| 1 lb = 17½/12 ounce troy |
| 1 lb = 4 stick |

| 1 cup ≈ 0.25 litre = 250 ml |
| 1 tablespoon ≈ 0.015 litre = 15 ml |
| 1 teaspoon ≈ 0.005 litre = 5 ml |

Note 1: I tested the approximation for tablespoon with water (1 mg ≈ 1 mg) and the approximation looks good enough. It of course depends on how full you fill your spoon. |

If you ever encounter in a german cookery book the word “Packerl”, check out its entry in section 12.

References


### Change History

2016/06/11
General: Added the package option to load ‘fmtcount’

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

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

2016/09/05
General: New message:
‘obsolete-command’
Replaced \cnum by \cuam

2016/09/09
Add ‘single’ to property list of singlekeys.

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

2016/09/26
General: \cuaddsinglekeys now tests if the unit exists (it didn’t before).
New option (and needed macros):

- add-temperature-to-check.
- round-half.

Recalculated all electron volt values for conversion (as ‘kg’ was wrong before). Let’s hope they are correct this time.

Replaced \prop_clear_new:c by \prop_clear:c.

2016/10/19

General: ‘convert-to-eV’ now also as optional argument available.

Option ‘load-time-option’ now spells ‘available’ correct.

Update of documentation.

Use \keys_set:nn only if second argument is not empty.

2016/10/28

General: \cutext (and \Cutext) and \cuan now parse their input like \cunum. This is needed as they also need to be changed.

Start implementation of “Change recipe from $n$ to $m$ persons.”

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.

2016/11/07

General: Finished writing v1.10.

2016/11/13

General: \cutext, \Cutext and \cuan check their input, allows conversion of units.

Change amounts for specific number of persons.

New commands: \cualabel and \curef.

New commands:

- \declarecookingunit and \providecookingunit.
- cuam-version and \cutext-version.
- cuam-version and \cutext-version.
- cuam-version and \cutext-version.
- cuam-version and \cutext-version.
- cuam-version and \cutext-version.

New options:

- set-cutext-translation-message.
- use-phrases.
- phrase-space.
- print-numerals.
- add-unit-to-group.
- use-phrases.
- set-cutext-translation-message.
- use-phrases.
- phrase-space.
- print-numerals.
- add-unit-to-group.

Now checks for ranges if both values can be printed as numerals (if activated) (bug fix).

Replaced translator by translations.

Reworked quite a lot of code.

2018/04/20

Allow round precision to be negative.
Change large portions of code.
Cooking Units-keys are not allowed to contain either “,” or “/”.
Fix argument specifiers.
Introduce key-groups (weight, volume, etc.).
New feature: Hooks
New Option: 42
New option: add-unit-to-group.
New option:

2018/06/05

Add “range-sign” for translations (not usable yet).

2017/03/10

General: \curef is now defined by \NewExpandableDocumentCommand instead of the Declare variant.

Removed \translate and others from code and replaced them with wrapper-macros.

Removed things like ‘cu-unit’ from translate input and placed them into separate tl’s.

2017/10/23

General: Added “phrases”.
 Added unit “stick” (of butter).
New option: amount-unit-space.
New option: phrase-space.
New option: print-numerals.
New option:

2018/06/05

Add “range-sign” for translations (not usable yet).
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
2021/03/21
General: Adding keys to unit
definition. ............ 1
Much better error handling. ......... 1
New commands to define keys:
\cudefinekeychain and \cuadtokeychain. ............ 1
New joke options:
nothing-special, going-bonkers,
fully-bonkers and xD-lol. ....... 1
New options: definition/symbol,
definition/gender,
definition/set-option,
definition/add-to-group. ....... 1
Overall reworking of internal code. 1
Remove exhaustive expansion from translations. Shouldn’t really change anything. ............ 1
Using commands as unit-keys now works. ............ 1
2022/03/26
General: Bugfix: Fixing property list retrieval error (storing property lists via translations did not work too well). ............ 1
Bugfix: Remove \peek_meaning_-
ignore_spaces:NTF. ............ 1
2022/06/06
General: Add two public commands
\cookingunits_print_-fraction:nnn and
\cookingunits_print_range:nnn. 1
Correction of french language.
Thanks to Alexis Jeandeau ....... 1
Implement a way to define the plural-finding algorithm. ....... 1
2022/11/27
General: Allow ranges and fractions to be combined|hdpindex ............ 1
New input type: products|hdpindex 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.

Symbols
<group> ............................... 20
<unit> ................................. 19
@@ commands:
@@_calculate_input_and_store_- \@_cunum_parse_range ............ 44
@@_cutext_default:nnn ............ 44
@@_cutext_parse_range ............ 44
@@_parse_fraction_in_input:www . 44

in:NN ................................. 44
one(m) .......................... 15
one(n) .......................... 15
parse-number ....................... 25
peek commands:
\peek_meaning_ignore_spaces:NTF  . 46
\phantom .......................... 44
phrase-space ........................ 30
print-numerals ..................... 24
prop commands:
\prop_clear:N  ....................... 45
\prop_clear_new:N  ............... 45
\providecookingunit  ............ 9,45
range-sign ........................ 25
recalculate-amount ................ 30
ref ................................. 32
round-half ........................ 27
round-precision .................... 26
round-to-int ....................... 27
S
set-context-translation-message .... 23
set-number-of-persons .............. 31
set-option .......................... 9
set-option-for ........................ 21
set-option-for<unit-key> ........... 21
set-special-sign ................... 23
set-unknown-message ............... 23
symbol ............................. 9
T
\thinspace .......................... 43
\translate ........................... 45
U
use-fmcount-numerals ............... 19
use-numerals-below ................ 24
use-phrases .......................... 26
\usepackage ......................... 19
X
xD-lol ............................... 34
48