The cooking-units package

Ben Vitecek
b.vitecek@gmx.at
2021/05/16

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

2 The Commands 3

3 Label & refs: Changing the amount of the recipe 5
  3.1 Rounding temperatures  6

4 Predefined units & some notes 6

5 Defining units 6

6 Defining options to change units 9
  6.1 Obsolete Commands 13

7 Language support 14
  7.1 Phrases 16

*This document corresponds to Benedikt Vitecek v2.00, dated 2021/05/16.
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{}2\epsilon) and here it is.
1.1 Supported languages

- German
- English
- French (currently suboptimal\(^1\))

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

2 The Commands

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

- \texttt{\textbackslash cunum}<\langle label\rangle>[\langle options\rangle]{\langle amount\rangle}\{\langle space\rangle}\{\langle unit-key\rangle}\}
- \texttt{\textbackslash cutext}<\langle label\rangle>[\langle options\rangle]{\langle amount\rangle}\{\langle unit-key\rangle}\}
- \texttt{\textbackslash Cutex}\langle label\rangle>[\langle options\rangle]{\langle amount\rangle}\{\langle unit-key\rangle}\}
- \texttt{\textbackslash cuam}<\langle label\rangle>[\langle options\rangle]{\langle amount\rangle}\}
- \texttt{\textbackslash cusetup}{\langle options\rangle}

Numbers and units are printed using \texttt{\textbackslash cunum}. The numerical part can interpret _ and / as (mixed) fractions and -- as a separator for ranges; to convert units use the option \langle old-unit\rangle=\langle new-unit\rangle\(^2\). 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\(^3\). \langle space\rangle adds a space between the number and the unit using \texttt{\phantom}. For a list of predefined units have a look at table 1. \langle label\rangle is explained in section 3.

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

\(^1\)You can only get limited information from the internet.
\(^2\)New keys can be added and defined, see section 4 and section 5 for further information.
\(^3\)You can customize this behavior, see section 8.
Decimal numbers are automatically rounded to 2 digits after the colon, temperatures (C, F, K and Re) are automatically rounded to integers.\footnote{You can – of course – change this behavior, see section 8.}

\texttt{\textbackslash cutext} and \texttt{\textbackslash Cutext} print the number and the written name of the unit. Since v1.10 it works similar\footnote{One could also say “exactly like”.} to \texttt{\textbackslash cunum}: it allows the conversion between units and interprets the numerical part (again _ and / are used for (mixed) fractions and \texttt{-} for ranges). Furthermore, \texttt{\textbackslash cutext} and \texttt{\textbackslash Cutext} allow the usages of numerals (see section 8.1 for more information).

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

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

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

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

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

\begin{verbatim}
\texttt{\textbackslash cusetup{1=m1}}
\end{verbatim}

\begin{verbatim}
1000 millilitres  \texttt{\textbackslash cutext{1}{ml}}
1000 millilitres  \texttt{\textbackslash Cutext{1}{ml}}
1000 to 2000 millilitres \texttt{\textbackslash cutext{1--2}{ml}}
12000 millilitres  \texttt{\textbackslash cutext{12}{ml}}
13000 millilitres  \texttt{\textbackslash Cutext{13}{ml}}
? litres        \texttt{\textbackslash Cutext{"?"}}
1/2 litres       \texttt{\textbackslash Cutext{1/2}}
\end{verbatim}

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

\begin{verbatim}
3             \texttt{\textbackslash cuam{3}}
2--3          \texttt{\textbackslash cuam{2--3}}
2/3           \texttt{\textbackslash cuam{2/3}}
1\frac{1}{2}  \texttt{\textbackslash cuam{1\_2/3}}
\end{verbatim}

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 \textit{use-phrases} (as I don’t know any english phrases, I switched the language to german for the following examples)

\begin{verbatim}
3             \texttt{\textbackslash cuam{3}}
2--3          \texttt{\textbackslash cuam{2--3}}
2/3           \texttt{\textbackslash cuam{2/3}}
1\frac{1}{2}  \texttt{\textbackslash cuam{1\_2/3}}
\end{verbatim}
3 Label & refs: Changing the amount of the recipe

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

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

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

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

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

```
\culabel{recipe}{2}
```

recipe for 2 persons:

- 10–20 dag flour,
- \(\frac{1}{2}\) ℓ water,
- 10 gramme nuts,
- 2–3 eggs,
- 180 °C (356 °F) open fire

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

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

recipe for 4 persons:

- 20–40 dag flour,
- 1 ℓ water,
- 20 gramme nuts,
- 4–6 eggs,
- 180 °C (356 °F) open fire
Note that fractions are automatically evaluated and that only values with a \texttt{\(\langle\text{label}\rangle\)} are changed (\texttt{\cunum{180}{C}} for example stays the same which also makes sense as the heat should be the same).

### 3.1 Rounding temperatures

By default temperatures are rounded to integers (using \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 8.2.1).

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

### 4 Predefined units & some notes

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

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

\begin{table}[h]
\centering
\begin{tabular}{|l|l|l|l|l|}
\hline
\textbf{description} & \textbf{key} & \textbf{default} & \textbf{description} & \textbf{key} & \textbf{default} \\
\hline
kilogramme & kg & kg & metre & m & m \\
decagramme & dag & dag & decimetre & dm & dm \\
gramme & g & g & centimetre & cm & cm \\
ounce & oz & oz & millimetre & mm & mm \\
pound & lb & lb & inch & in & in \\
stick (of butter) & stick & stick & & & \\
\hline
day & d & d & litre & l & l \\
hour & h & h & decilitre & dl & dl \\
minute & min & min & centilitre & cl & cl \\
second & s & s & millilitre & ml & ml \\
calorie & cal & cal & degree Celsius & °C & \\
kilocalorie & kcal & kcal & degree Fahrenheit & °F & \\
joule & J & J & degree Réaumur & °Ré & \\
kilojoule & kJ & kJ & kelvin & K & \\
electron volt & eV & eV & & & \\
\hline
\end{tabular}
\end{table}
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>pn</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>$c^2\hbar/eV$</td>
</tr>
<tr>
<td>(chbareV-1)3</td>
<td>$c^3\hbar^2/eV^3$</td>
</tr>
</tbody>
</table>

5 Defining units

New units can be defined using

- \texttt{\ declarecookingunit[(symbol/key-val-list)]{(unit-key)}}
- \texttt{\ newcookingunit[(symbol/key-val-list)]{(unit-key)}}
- \texttt{\ providecookingunit[(symbol/key-val-list)]{(unit-key)}}
These commands define the unit \texttt{(unit-key)}. Note that \texttt{(unit-key)} can neither contain \texttt{/} nor \	exttt{;} 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.

\texttt{\newcookingunit} raises an error if the unit is already defined, \texttt{\declarecookingunit} creates or (if given) overwrites \texttt{(symbol)} and \texttt{\providecookingunit} does nothing if the unit is already defined.

All units have male gender \texttt{m} by default (unless you change it using a key below).

Some examples:

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

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

\begin{tabular}{l}
\texttt{symbol} & symbol = \{\texttt{(symbol)}\} \\
\texttt{gender} & gender = \{\texttt{m/f/n}\} \\
\texttt{set-option} & set-option = \{\texttt{(key-val-list)}\} \\
\texttt{add-to-group} & add-to-group = \{\texttt{(group)}\} \\
\texttt{natural-unit} & natural-unit = \{\texttt{true/false}\} \\
\end{tabular}

Those keys can only be used in the optional argument of \texttt{\declarecookingunit}, \texttt{\newcookingunit} or \texttt{\providecookingunit}. They can be used to define some properties of the unit during its initialization.

\texttt{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.

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

\texttt{set-option} allows to add some key-vars to the specific unit which are activated once the unit is used. See page 17.

\texttt{add-to-group} adds the unit defined to \texttt{(group)}. See section 8.2.1 for more information.

\texttt{natural-unit} is a simple \texttt{true/false} switch. If \texttt{true} the unit will be specified to be a “natural-unit”. This is more or less a joke option.
\declarecookingderivatives \declarecookingderivatives \{\langle unit-list\rangle\} \{\langle unit-key\rangle\} \{\langle mathematical-relation\rangle\} \{\langle unit-symbol\rangle\}

This function is experimental. Defines new units which are a combination of the units given in \langle unit-list \rangle and their key-chain. \langle unit-key \rangle, \langle mathematical-relation \rangle and \langle unit-symbol \rangle accept \#1 to \#n as arguments with \( n \) being the number of units given in \langle unit-list \rangle. \( n \) 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 \( \text{kg} \cdot \text{m}^2 \cdot \text{s}^{-2} \) into \( \text{oz} \cdot \text{in}^2 \cdot \text{min}^{-2} \). To check if you are correct you use \\declarecookingderivatives:

\makeatletter
defclarecookingderivatives\{
\makeatother
\text{kg, m, s}\}{\#1*\#2:#3} \{ \langle \#1\rangle \ast \langle \#2\rangle \cdot \langle \#3\rangle^{-2} \} \{ \text{sfrac}\{\langle \#1\rangle, \langle \#2\rangle\}^{-2}\{\langle \#3\rangle\}^{-2}\}

Using \\cunum[\text{kg} \cdot \text{m}^2 \cdot \text{s}^{-2} = \text{oz} \cdot \text{in}^2 \cdot \text{min}^{-2}]{1}{\text{kg} \cdot \text{m}^2 \cdot \text{s}^{-2}} shows that \( 1 \text{ kg m}^2 \cdot \text{s}^{-2} \) is equal to \( 196829101.34 \text{ oz in}^2 \cdot \text{min}^{-2} \).

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.

6 Defining options to change units

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

- \texttt{\cudefinekeychain}
- \texttt{\cuddefinekeychain}
- \texttt{\cuaddtokeychain}
- \texttt{\cuaddsinglekeys}
- \texttt{\cuaddsinglekeys}
\cudefinekeychain \cudefinesinglekey
\cudefinekeychain
{\langle unit-key-1 \rangle \{\langle value \rangle\}}
{\langle unit-key-2 \rangle \{\langle value \rangle unit-key-1\}}
{\langle unit-key-3 \rangle \{\langle value \rangle unit-key-1\}}
...
\cudefinesinglekey{\langle unit-key-1 \rangle}
{\langle unit-key-2 \rangle \{1 unit-key-2 are ... unit-key-1\}}
{\langle unit-key-3 \rangle \{1 unit-key-3 are ... unit-key-1\}}
...
If you define new units (see section 5) and cannot add them to already existing keys you 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
- ...

\begin{align*}
1 \text{ kg} & = 1 \text{ kg} \\
1 \text{ kg} & = 100 \text{ dag} \\
1 \text{ kg} & = 1000 \text{ g} \\
1 \text{ kg} & = 35.27399 \text{ oz} \\
1 \text{ kg} & = 2.2046226 \text{ lb}
\end{align*}

\cudefinekeychain
{\langle kg \rangle \{ 1 \}}
{\langle dag \rangle \{ 100 \} \% 1 \text{ kg are 100 dag}}
{\langle g \rangle \{ 1000 \} \% 1 \text{ kg are 1000 g}}
{\langle oz \rangle \{ 35.27399 \} \% 1 \text{ kg are 35.27399 oz}}
{\langle lb \rangle \{ 2.2046226 \} \% 1 \text{ kg are 2.2046226 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
}

\textbf{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 }
}

\textbf{Example:} To convert degree Fahrenheit to degree Celsius, kelvin and degree Réamur one needs the formulas\footnote{See Wikipedia.}

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

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 }
}

This defines the key F with the values F, C, K and Re.
first parses through its unit-list and searches for a base unit key which is already in a key-chain (aka. was defined by \cudefinekeychain). 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 (...) must be a number.

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

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

% defining new unit ’stone'
\newcookingunit{st}
\cuaddtokeychain
\{ {lb} { 14 } \} \% unit already in a key-chain.
\{ {st} { 1 } \} \% new unit. 1st = 14lb

\cunum{lb=st}{1}{lb}\n\cunum{st=lb}{1}{st}\n\cunum{st=g}{1}{st}\n\cunum{st=kg}{1}{st}\n\cunum{kg=st}{16}{kg}\n\cunum{kg=st}{1}{kg}\n
\Note: Of course using
\cuaddtokeychain
\{ {st} { 1/14 } \} \% 1lb = 1/14st
\{ {lb} { 1 } \} \% unit already in a key-chain.
\}
**Example:** Now you want to add degree Rømer and convert Celsius to degree Rømer:

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

\% defining new unit ‘degree Rømer’
\newcookingunit \ensuremath{ {} ^ { \circ } }\kern-\scriptspace R{o} \{Ro\}
\cuaddsinglekeys \{C\} \% adds value ‘Ro’ to key ‘C’.

\{
    \{Ro\} \{ #1 \times 21/40 + 7.5 \}
\}
\cusetoptionfor\{Ro\}\{ round-precision = 0 \} \% round to integer automatically

10^\circ C \quad \cunum\{10\}\{C\}\\$
13^\circ Ro \quad \cunum[C=Ro]\{10\}\{C\}\$

### 6.1 Obsolete Commands

\cudefinekeys\{\cudefinekeys\{\unit-key-1\}\}
\{
    \{\unit-key-2\}\{1 \unit-key-1 \text{ are } ... \unit-key-2\}\$
    \{\unit-key-3\}\{1 \unit-key-1 \text{ are } ... \unit-key-3\}\$
    \{\unit-key-4\}\{1 \unit-key-1 \text{ are } ... \unit-key-4\}\$
    \ldots
\}

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

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

\cuaddkeys\{\cuaddkeys\{\unit-key-1\}\}
\{
    \{\unit-key-2\}\{1 \unit-key-1 \text{ are } ... \unit-key-2\}\$
    \{\unit-key-3\}\{1 \unit-key-1 \text{ are } ... \unit-key-3\}\$
    \{\unit-key-4\}\{1 \unit-key-1 \text{ are } ... \unit-key-4\}\$
    \ldots
\}

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

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

Unit names and symbols depend on the language. To change the name and symbol for given language you can use \cdefine; to only change symbols use \definesymbol. Those are special keys (as they cannot be used as units). Not only are printed units language depending, but as is the decimal mark (, or .) and the text which substitutes the range-sign. To set the decimal mark use decimal-mark (see examples below), to set the range-sign for \cutext and \Cutext use cutext-range-sign.

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

Furthermore if you are using numerals you may also use the keys one(m), one(f) and one(n). Integers below a certain value (see option use-numerals-below) are written-out. The 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).

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

This command defines the names (and optionally the symbol) of the units printed in \cutext and \Cutext (and \cnum regarding the symbol) for the specific ⟨Language⟩.

For details regarding ⟨language⟩ see the translations documentation.

If the plural form of the name differs from the singular form use \{⟨plural⟩\} to specify the plural form, else it will be equal to its singular form. The singular form is only used if the number in \cutext and \Cutext is equal to 1.

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

\cdefine{English}
{
  {kg} {kilogramme} 
  {oz} {ounce} 
  {h} {hour} [hours] 
  {C} {degree space Celsius} [degrees space Celsius]  
  {decimal-marker} {} 
  {cutext-range-sign} {~to~} 
  {one(m)} {one} 
  {one(f)} {one} 
  {one(n)} {one} 
}

\cdefine{German}
{
  {kg} {Kilogramm} <n> 
  {oz} {Unze} <f>
\cudefinesymbol \cudefinesymbol{\textbf{\textit{Language}}} \\
{\langle\text{unit-key-1} \rangle} \{\langle\text{symbol-1}\rangle\} \\
{\langle\text{unit-key-2} \rangle} \{\langle\text{symbol-2}\rangle\} \\
\ldots \\

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

\cudefinesymbol \cudefinesymbol{\textbf{\textit{English}}} \\
{\{\text{decimal-mark}\} \{.\}} \\
{\{\text{cutext-range-sign}\} \{-to-\}} \\
{\{\text{one(m)}\} \{\text{one(f)}\} \{\text{one(n)}\} \{\text{one}\}} \\

\cudefinesymbol \cudefinesymbol{\textbf{\textit{German}}} \\
{\{\text{decimal-mark}\} \{,\}} \\
{\{\text{cutext-range-sign}\} \{-bis-\}} \\
{\{\text{one(m)}\} \{\text{ein}\}} \\
{\{\text{one(f)}\} \{\text{eine}\}} \\
{\{\text{one(n)}\} \{\text{ein}\}} \\

\cudefinesymbol \cudefinesymbol{\textbf{\textit{French}}} \\
{\{1\} \{L\}} \\
{\{dl\} \{dL\}} \\
{\{cl\} \{cL\}} \\
{\{ml\} \{mL\}} \\
{\{\text{decimal-mark}\} \{.\}} \\
{\{\text{one(m)}\} \{\text{un}\}} \\
{\{\text{one(f)}\} \{\text{une}\}} \\
{\{\text{one(n)}\} \{\text{un}\}} \\

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

\begin{verbatim}
12 ducks weed
3 ducks nuts
10 ducks duckmeat
\end{verbatim}

### 7.1 Phrases

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

\begin{verbatim}
\cudefinephrase{⟨Language⟩}
{ ⟨integer-1⟩ } { ⟨phrase-1⟩ } {⟨phrase-1-plural⟩} {⟨gender-1⟩} \\
{ ⟨integer-2⟩ } { ⟨phrase-2⟩ } {⟨phrase-2-plural⟩} {⟨gender-2⟩} \\
... 
\end{verbatim}

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 $\texttt{cuam}$ 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”:

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

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

\begin{verbatim}
\selectlanguage{ngerman}
\cusetup{use-phrases=true}
1 Dutzend \quad \texttt{cuam{12}}
2 Dutzend \quad \texttt{cuam{24}}
25 \quad \texttt{cuam{25}}
1 Schock \quad \texttt{cuam{60}}
2 Schock \quad \texttt{cuam{120}}
121 \quad \texttt{cuam{121}}
1 halbes Dutzend \quad \texttt{cuam{6}}
18 \quad \texttt{cuam{18}}
\end{verbatim}
As you can see, “Schock” (60) is preferred over “Dutzend” (12) as it linked to the higher number. Furthermore, for 6 the phrase “halbes Dutzend” (half a dozen) is used, but because it is a star version it is not used for 18.

8 Options

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

\cusetup \cusetup{(options)}

Options can be set using \cusetup\{(options)\}.

\cusetoptionfor \cusetoptionfor\{(unit-list)\}{(options)}
\cuaddoptionfor \cuaddoptionfor\{(unit-list)\}{(options)}
\cuclearoptionfor \cuclearoptionfor\{(unit-list)\}

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

\cusetoptionfor sets \{(options)\} to each unit in \{(unit-list)\} overwriting the old ones.
\cuaddoptionfor adds \{(options)\} to each unit in \{(unit-list)\}.
\cuclearoptionfor clears all options given to each unit in \{(unit-list)\}.

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

\begin{itemize}
\item \texttt{\cusetoptionfor{C,F,K,Re}{round-precision=-1}}
\end{itemize}

\begin{verbatim}
75 °C  \cunum{75.23}{C}\n75 °F  \cunum{75.23}{F}\n75 K   \cunum{75.23}{K}\n75 °Ré \cunum{75.23}{Re}\n
80 °C  \cunum{75.23}{C}\n80 °F  \cunum{75.23}{F}\n80 K   \cunum{75.23}{K}\n80 °Ré \cunum{75.23}{Re}\n
75.23 °C \cunum{75.23}{C}\n75.23 °F \cunum{75.23}{F}\n75.23 K  \cunum{75.23}{K}\n75.23 °Ré \cunum{75.23}{Re}
\end{verbatim}
8.1 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 \texttt{to write-out numbers below use-numerals-below (13 by
default), integers above are printed as numbers. You can decide to not print any numerals
by setting print-numerals to false.

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

\begin{verbatim}
one kilogramme \texttt{1}\texttt{kg}\\
One kilogramme \texttt{1}\texttt{kg}\\
two kilogramme \texttt{2}\texttt{kg}\\
Two kilogramme \texttt{2}\texttt{kg}\\
twelve kilogramme \texttt{12}\texttt{kg}\\
Twelve kilogramme \texttt{12}\texttt{kg}\\
13 kilogramme \texttt{13}\texttt{kg}\\
13 kilogramme \texttt{13}\texttt{kg}\\
14 kilogramme \texttt{14}\texttt{kg}
\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 one(m), one(f) and one(n) to change the printed “one” (as “one” is in many languages dependent on the gender of the following word. E.g in
German: Masculine: ein Baum, Feminin: eine Pflanze, Neutrum: ein Auto).

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

8.2 Normal options

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

8.2.1 Unit Specific options

\begin{verbatim}
<unit> (unit-key-1) = (unit-key-2)
Change (unit-key-1) to (unit-key-2) (see section 6 to define new options).
\end{verbatim}
<group> \( \langle \text{group} \rangle = \langle \text{unit-key} \rangle \)

Changes each unit contained in \( \langle \text{group} \rangle \) to \( \langle \text{unit-key} \rangle \) \((\langle \text{unit-key} \rangle \) must be part of \( \langle \text{group} \rangle \)).

<table>
<thead>
<tr>
<th>( \langle \text{group} \rangle )</th>
<th>( \text{default } \langle \text{unit-key} \rangle )</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 \cudeclareunitgroup: \cudeclareunitgroup \( \langle \text{group-name} \rangle \) \{ \langle \text{unit-list} \rangle \}

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

Example: Define the group “weight”: \cudeclareunitgroup \{weight\} \{ kg , dag, g, oz, lb, stick \}

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

You can also add units to an existing group using \cuaddtounitgroup: \cuaddtounitgroup \langle \text{group} \rangle \{ \langle \text{unit-list} \rangle \}

Adds \( \langle \text{unit-list} \rangle \) to an already existing \( \langle \text{group} \rangle \) \((\langle \text{group} \rangle \) and \( \langle \text{unit-list} \rangle \) both need to exist).

Example: Adding st to the group weight

\cuaddtounitgroup \{weight\} \{st\}

\cusetup{weight=\text{g}}

1000 g \cunum{1}{kg}\
10 g \cunum{1}{dag}\
1 g \cunum{1}{g}\
28.35 g \cunum{1}{oz}\
453.59 g \cunum{1}{lb}\
113.4 g \cunum{1}{stick}\
6350.29 g \cunum{1}{st}
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⟩ =
set-option-for⟨unit-key⟩ = {
    ⟨key1 = value1, ...
}

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 \cuersetoptionfor and \cuaddoptionfor.

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

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 \cuersetoptionfor and \cuaddoptionfor.

erase-all-options
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).

8.2.2 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 \text{\cutext{1}{kg}}
2 kilogramme \text{\Cutext{2}{kg}}
\frac{1}{2} kilogramme \text{\cutext{1/2}{kg}}
? kilogramme \text{\cutext{?}{kg}}
1000 to 2000 gramm\text{\cutext[kg=g]{1--2}{kg}}
1 kg \text{\cutext{1}{kg}}
2 kg \text{\Cutext{2}{kg}}
\frac{1}{2} kg \text{\cutext{1/2}{kg}}
? kg \text{\cutext{?}{kg}}
1000–2000 g \text{\cutext[kg=g]{1--2}{kg}}

\text{cutext-change-unit} \quad \text{cutext-change-unit = \langle true/false \rangle}
Set this option to \text{false} if you do not want the units of \text{\cutext} and \text{\Cutext} to be changed. Set to \text{true} by default.

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

8.2.3 Hooks

\text{commands-add-hook} \quad \text{commands-add-hook = \langle \langle code \rangle \rangle}
\text{cunum-add-hook} \quad \text{cunum-add-hook = \langle \langle code \rangle \rangle}
\text{cutext-add-hook} \quad \text{cutext-add-hook = \langle \langle code \rangle \rangle}
\text{Cutext-add-hook} \quad \text{Cutext-add-hook = \langle \langle code \rangle \rangle}
\text{cuam-add-hook} \quad \text{cuam-add-hook = \langle \langle code \rangle \rangle}

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

Please be 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{\cusetup{commands-add-hook={\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>206</td>
</tr>
<tr>
<td>\cutext</td>
<td>62</td>
</tr>
<tr>
<td>\Cutext</td>
<td>27</td>
</tr>
<tr>
<td>\cuam</td>
<td>61</td>
</tr>
<tr>
<td>total</td>
<td>356</td>
</tr>
</tbody>
</table>

### 8.2.4 Input and Outputs

| expand-both | expand-both = \{n/o/f/x\} |
| expand-amount| expand-amount = \{n/o/f/x\} |
| expand-unit  | expand-unit = \{n/o/f/x\} |

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

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

| set-special-sign | set-special-sign = \{(character(s))\} |
| add-special-sign | add-special-sign = \{(character(s))\} |

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

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

| set-unknown-message | set-unknown-message = \{error/warning/none\} |

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

| set-cutext-translation-message | set-cutext-translation-message = \{error/warning/none\} |

If a translation for \cutext and \Cutext is not available 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 = \langle true/false \rangle

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 \texttt{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:  \texttt{(Using the package option \texttt{use-fmtcount-numerals}:}

\begin{verbatim}
one kilogramme \texttt{cutext(1){kg}}
two kilogramme \texttt{cutext(2){kg}}
twelve kilogramme \texttt{cutext(12){kg}}
13 kilogramme \texttt{cutext(13){kg}}
\cusetup{print-numerals=false}
1 kilogramme \texttt{cutext(1){kg}}
2 kilogramme \texttt{cutext(2){kg}}
12 kilogramme \texttt{cutext(12){kg}}
13 kilogramme \texttt{cutext(13){kg}}
\end{verbatim}

use-numerals-below \, use-numerals-below = \langle integer \rangle

If print-numerals is true, prints the numerals in \texttt{cutext} and \texttt{Cutext} for integers smaller than \langle integer \rangle. \langle integer \rangle is by default 13. You can deactivate the printing of numerals by print-numerals=false.

\begin{verbatim}
one kilogramme \texttt{cutext(1){kg}}
two kilogramme \texttt{cutext(2){kg}}
twelve kilogramme \texttt{cutext(12){kg}}
13 kilogramme \texttt{cutext(13){kg}}
\cusetup{use-numerals-below=10}
one kilogramme \texttt{cutext(1){kg}}
two kilogramme \texttt{cutext(2){kg}}
12 kilogramme \texttt{cutext(12){kg}}
13 kilogramme \texttt{cutext(13){kg}}
\cusetup{use-numerals-below=0}
1 kilogramme \texttt{cutext(1){kg}}
2 kilogramme \texttt{cutext(2){kg}}
12 kilogramme \texttt{cutext(12){kg}}
13 kilogramme \texttt{cutext(13){kg}}
\cusetup{use-numerals-below=12001}
one thousand gramme \texttt{cutext[kg=g]{1}{kg}}
two thousand gramme \texttt{cutext[kg=g]{2}{kg}}
twelve thousand gramme \texttt{cutext[kg=g]{12}{kg}}
13000 gramme \texttt{cutext[kg=g]{13}{kg}}
\end{verbatim}
numeral-function = \langle 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{
  numeral-function = \numberstringnum ,
  Numeral-function = \Numberstringnum
}

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

parse-number = (true/false)

If set to false prints the number of \cunum, \cutext, \Cutext and \cuam as they are (after some ... well ... parsing due to “ _ ”). Is set to true by default.

\cusetup{parse-number=false}

\cunum[kg=g]{1}{kg}
\cunum{1--2}{kg}
\cunum{1----------2}{kg}
\cunum[kg=g]{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[kg=g]{12}{kg}
\cuam{1----------2}
\cuam{1,2}
\cuam{1_1/2}
\cuam{kwflk}

\cunum-range-sign = \langle string \rangle
\cutext-range-sign = \langle string \rangle
\cutext-range-sign = \langle string \rangle

\cunum-range-sign sets the printed range sign used in \cunum (and \cuam) to \langle string \rangle, \cutext-range-sign sets the printed range sign used in \cutext and \Cutext to \langle string \rangle. Using range-sign sets the range signs for both \cunum and \cuam and \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 cutext-range-sign (see section 7). If the option cutext-range-sign is set the language symbol will be ignored.
use-phrases = (true/false)

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

Example: For the German language:

\selectlanguage{ngerman}

12 \cuam{12}\%
12–24 \cuam{12--24}\%
36 \cuam{36}\%

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}\%

8.2.5 Rounding options

round-precision = (integer)

Rounds the amount automatically to (integer) digits after the colon. Note that units like C, F, K and Re are still rounded to integers due to \cusetoptionfor.
\cusetup{round-precision=5}
\cunum{1.23456789}{kg} \cunum{12.587}{g} \cunum{194}{kg} \cunum{200--210}{C} \cunum{0.0012}{K}

Note: Negative numbers are also allowed.
\cusetoptionfor{C,F}{round-precision=-1}
\cunum{-271,2}{C} \cunum{0.0012}{K} \cunum{185}{C} \cunum{180--200}{C}

round-to-int \qquad 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 \qquad 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 \dots 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}
\cunum{0.005}{kg} \cunum{-0.005}{kg} \cunum{1.245}{kg} \cunum{0.005}{kg} \cunum{-0.005}{kg} \cunum{1.245}{kg}

8.2.6 Fractions

eval-fraction \qquad 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.

26
### convert-fraction

**convert-fraction = (true/false)**

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

```latex
\csetup{convert-fraction=true}
```

<table>
<thead>
<tr>
<th>Fraction</th>
<th>Unit Conversion</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/3 kg</td>
<td>(\cunum{1/3}{kg})</td>
</tr>
<tr>
<td>1/2 kg</td>
<td>(\cunum{1/2}{kg})</td>
</tr>
<tr>
<td>1(\frac{1}{2}) kg</td>
<td>(\cunum{1\frac{1}{2}}{kg})</td>
</tr>
<tr>
<td>1(\frac{1}{2}) kg</td>
<td>(\cunum{1\frac{1}{2}}{kg})</td>
</tr>
<tr>
<td>1(\frac{1}{2}) kg</td>
<td>(\cunum{1\frac{1}{2}}{kg})</td>
</tr>
</tbody>
</table>

### fraction-command

**fraction-command = \command**

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

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

```latex
\newcommand{\myfrac}[2]{#1/#2}
\csetup{fraction-command=\myfrac}
```

<table>
<thead>
<tr>
<th>Fraction</th>
<th>Unit Conversion</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/8</td>
<td>(\cuam{1/8})</td>
</tr>
<tr>
<td>1/2 kg</td>
<td>(\cunum{1/2}{kg})</td>
</tr>
<tr>
<td>4/5 °C</td>
<td>(\cunum{4/5}{°C})</td>
</tr>
<tr>
<td>12/3 kg</td>
<td>(\cunum{12/3}{kg})</td>
</tr>
<tr>
<td>1/8</td>
<td>(\cuam{1/8})</td>
</tr>
<tr>
<td>1/2 kg</td>
<td>(\cunum{1/2}{kg})</td>
</tr>
<tr>
<td>4/5 °C</td>
<td>(\cunum{4/5}{°C})</td>
</tr>
<tr>
<td>12/3 kg</td>
<td>(\cunum{12/3}{kg})</td>
</tr>
</tbody>
</table>

### fraction-inline

**fraction-inline = \{input containing #1 and #2\}**

Similar to **fraction-command** only that you don’t have to define a command to alter the output of the fraction.
8.2.7 Spaces

\texttt{mixed-fraction-space = \{length\}}

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

\begin{verbatim}
1\%3 \texttt{\cuam{1\_2/3}}
1\%4\% kg \texttt{\cunum{1\_2/3}{kg}}
10\%\% kg \texttt{\cunum{10\_2/3}{kg}}
\texttt{\cusetup{mixed-fraction-space=1em}}
\texttt{\cuam{1\_2/3}}
\texttt{\cunum{1\_2/3}{kg}}
\texttt{\cunum{10\_2/3}{kg}}
\texttt{\cusetup{mixed-fraction-space=0em}}
\texttt{\cuam{1\_2/3}}
\texttt{\cunum{1\_2/3}{kg}}
\texttt{\cunum{10\_2/3}{kg}}
\end{verbatim}

\texttt{cutext-space = \{(string)\}}

\texttt{\langle string \rangle} 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 ~.

\begin{verbatim}
1 kilogramme \texttt{\cutext{1}{kg}}
10 kilogramme \texttt{\Cutext{10}{kg}}
\texttt{\cusetup{cutext-space=\space}}
1 kilogramme \texttt{\cutext{1}{kg}}
10 kilogramme \texttt{\Cutext{10}{kg}}
\texttt{\cusetup{cutext-space=\{}\}}
1 kilogramme \texttt{\cutext{1}{kg}}
10 kilogramme \texttt{\Cutext{10}{kg}}
\texttt{\cusetup{cutext-space=\}}
1 kilogramme \texttt{\cutext{1}{kg}}
10 kilogramme \texttt{\Cutext{10}{kg}}
\end{verbatim}

28
phrase-space = \{(string)\}

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

(Switching to german)

\selectlanguage{ngerman}

\cuam{12}\\\cuam{144}\\cuam{12}\\cuam{144}\\cuam{12}\\cuam{144}\cuam{12}\cuam{144}\\cuam{12}\cuam{144}\cuam{12}\cuam{144}\cuam{12}\cuam{144}\cuam{12}\cuam{144}

amount-unit-space = \{(string)\}

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

\cunum{1}{kg}\\cunum{1/2}{kg}\cunum{1--2}{kg}\cunum{1}{kg}\cunum{1/2}{kg}\cunum{1--2}{kg}\cunum{1}{kg}\cunum{1/2}{kg}\cunum{1--2}{kg}\cunum{1}{kg}\cunum{1/2}{kg}\cunum{1--2}{kg}\cunum{1}{kg}\cunum{1/2}{kg}\cunum{1--2}{kg}

8.2.8 label & refs

recalculate-amount = \{true/false\}

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

Set the number of persons = ⟨integer⟩

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

<table>
<thead>
<tr>
<th>Persons</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1 kg</td>
</tr>
<tr>
<td>1</td>
<td>10 kg</td>
</tr>
<tr>
<td>4</td>
<td>2 kg</td>
</tr>
<tr>
<td>2</td>
<td>20 kg</td>
</tr>
<tr>
<td>3</td>
<td>1.5 kg</td>
</tr>
<tr>
<td>1.5</td>
<td>15 kg</td>
</tr>
<tr>
<td>2</td>
<td>1 kg</td>
</tr>
<tr>
<td>1</td>
<td>10 kg</td>
</tr>
<tr>
<td>1</td>
<td>1 kg</td>
</tr>
<tr>
<td>10</td>
<td>kg</td>
</tr>
<tr>
<td>1</td>
<td>0.5 kg</td>
</tr>
<tr>
<td>0.5</td>
<td>5 kg</td>
</tr>
</tbody>
</table>

**label**

Define label = ⟨(string)*⟨integer⟩⟩

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

<table>
<thead>
<tr>
<th>Persons</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 person</td>
<td>2 dag</td>
</tr>
<tr>
<td>2</td>
<td>2 dag</td>
</tr>
<tr>
<td>2</td>
<td>10 kg</td>
</tr>
<tr>
<td>4</td>
<td>8 dag</td>
</tr>
<tr>
<td>8</td>
<td>10 kg</td>
</tr>
</tbody>
</table>

**get-label**

Define get-label = ⟨(label)⟩

The key-value version of **\curef**. Note that this key doesn’t save the value inside a macro but rather prints it directly into the document.
\textbf{Note:} \texttt{\texttt{curef}} is expendable.

\begin{itemize}
\item \texttt{ref} = \{⟨label⟩\}
\end{itemize}

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

\begin{itemize}
\item \texttt{curef-add-forbidden-unit} = \{(unit list)\}
\item \texttt{curef-remove-forbidden-unit} = \{(unit list)\}
\item \texttt{curef-clear-forbidden-units} = (true/false)
\end{itemize}

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.
8.3 Weird options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
</table>
| `check-temperature`     | Checks if the used temperature is below absolute zero. Currently C, F, K and Re are supported. While \cunum{0}{K} is ok, \cunum{-1}{K} raises an error, same for the others. Is set to false by default. To add new units see `add-temperature-to-check`.
| `add-temperature-to-check` | This option adds \(\langle \text{unit-key-1} \rangle\) 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 \(\langle \text{unit-key} \rangle = \langle \text{minimum-value} \rangle\). This sets the allowed minimum value of \(\langle \text{unit-key} \rangle\) to \(\langle \text{minimum-value} \rangle\). 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: |
| `convert-to-eV`         | Converts (nearly) every unit in table 1 to electron volt or the respective derivative (if possible). Note that this option is: a) experimental and probably will forever be and (b) just a joke, you are not supposed to use this units in a cookery book (and as you see this package doesn’t support the arrangement of such huge numbers). Also you may want to check the values if you really want to use them, just to be sure (I’ve checked them several times and hope they are finally correct, but mistakes happen). |
\cusetup{convert-to-eV=true}

\cunum{1}{kg}\\n\cunum{1}{l}\\n\cunum{1}{J}\\n\cunum{1}{m}\\n\cunum{1}{C}\\n\cunum{1}{s}

\begin{verbatim}
add-natural-unit = \langle\unit-key\rangle
\end{verbatim}

This option adds \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{verbatim}
42 = \langle\true/false\rangle
\end{verbatim}

Take a good guess.

\begin{verbatim}
42 kg \cunum{1}{kg}\\n42 \text{g} \cunum[kg=g]{1}{kg}\\n42 J \cunum{1.5}{J}\\n42^\circ\text{C} \cunum{180}{C}\\n42\frac{3}{4}s \cunum{15/4}{s}\\n42\text{--}2\text{ min} \cunum{1--2}{min}\\n42(!) \ell \cunum{?}{l}
\end{verbatim}

\begin{verbatim}
nothing-special = \langle\true/false\rangle
\end{verbatim}

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

\begin{verbatim}
going-bonkers = \langle\true/false\rangle
\end{verbatim}

\begin{verbatim}
fully-bonkers = \langle\true/false\rangle
\end{verbatim}

\begin{verbatim}
xD-lol = \langle\true/false\rangle
\end{verbatim}

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

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

\texttt{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 6.

\texttt{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}).

\texttt{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.
9 Bugs & Feedback

Bug reports are always welcome. If you are sending a bug report please include a minimal working example showing the bug and a short description. If you use mail please add cooking-units to the e-mail header. GMX has the habit of putting e-mails into the spam account and adding cooking-units to the header makes it easier to recognize those e-mails. It can also take longer of GitHub, but I hope I figured out how to get a mail if a new issue is created (by not me).

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

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

*sicilian* (Rottel*de*) Around 0.850 kg

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

**Canna**

*sicilian* (Rute*de*, *rod*en) About 2 m bzw. about 6 foot.

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

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

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

(Story “The missing Three Quarters”)

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

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

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

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

*Tockengerm (dry Germ) to be precise

For what kind of thing do I need Natrium for?

A  Translations

This section contains the list of available translations. Each table shows the available translations regarding the unit symbol, the unit name (printed if \text or \textit 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>(plurals)</td>
<td>m</td>
</tr>
<tr>
<td>dag</td>
<td>dag</td>
<td>decagramme</td>
<td>(plurals)</td>
<td>m</td>
</tr>
<tr>
<td>g</td>
<td>g</td>
<td>gramme</td>
<td>(plurals)</td>
<td>m</td>
</tr>
<tr>
<td>oz</td>
<td>oz</td>
<td>ounce</td>
<td>(plurals)</td>
<td>m</td>
</tr>
<tr>
<td>lb</td>
<td>lb</td>
<td>pound</td>
<td>(plurals)</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>(plurals)</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>(electrons)</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>(coffeespoonfuls)</td>
<td>m</td>
</tr>
<tr>
<td>dsp</td>
<td>dsp.</td>
<td>dessertspoonful</td>
<td>(dessertspoonfuls)</td>
<td>m</td>
</tr>
<tr>
<td>ssP</td>
<td>ssP.</td>
<td>saltspoonful</td>
<td>(saltspoonfuls)</td>
<td>m</td>
</tr>
<tr>
<td>Msp</td>
<td>Msp.</td>
<td>—</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>decimal-mark</td>
<td>—</td>
<td>.</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>one(m)</td>
<td>—</td>
<td>one</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>one(f)</td>
<td>—</td>
<td>one</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>one(n)</td>
<td>—</td>
<td>one</td>
<td>m</td>
<td></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></td>
<td>m</td>
</tr>
<tr>
<td>dag</td>
<td>dag</td>
<td>decagram</td>
<td></td>
<td>m</td>
</tr>
<tr>
<td>g</td>
<td>g</td>
<td>gram</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>Re</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>meter</td>
<td>(meters)</td>
<td>m</td>
</tr>
<tr>
<td>dm</td>
<td>dm</td>
<td>decimeter</td>
<td>(decimeters)</td>
<td>m</td>
</tr>
<tr>
<td>cm</td>
<td>cm</td>
<td>centimeter</td>
<td>(centimeters)</td>
<td>m</td>
</tr>
<tr>
<td>mm</td>
<td>mm</td>
<td>millimeter</td>
<td>(millimeters)</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>liter</td>
<td>(liters)</td>
<td>m</td>
</tr>
<tr>
<td>dl</td>
<td>dl</td>
<td>deciliter</td>
<td>(deciliters)</td>
<td>m</td>
</tr>
<tr>
<td>cl</td>
<td>cl</td>
<td>centiliter</td>
<td>(centiliters)</td>
<td>m</td>
</tr>
<tr>
<td>ml</td>
<td>ml</td>
<td>milliliter</td>
<td>(milliliters)</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>pn.</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.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>n</td>
<td></td>
</tr>
<tr>
<td>dag</td>
<td>dag</td>
<td>Dekagramm</td>
<td>n</td>
<td></td>
</tr>
<tr>
<td>g</td>
<td>g</td>
<td>Gramm</td>
<td>n</td>
<td></td>
</tr>
<tr>
<td>oz</td>
<td>oz</td>
<td>Unze</td>
<td>f</td>
<td></td>
</tr>
<tr>
<td>lb</td>
<td>lb</td>
<td>Pfund</td>
<td>n</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>°C</td>
<td>Grad Celsius</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>°F</td>
<td>Grad Fahrenheit</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>Ré</td>
<td>°Ré</td>
<td>Grad Réamur</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>K</td>
<td>K</td>
<td>Kelvin</td>
<td>n</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>d</td>
<td>Tag (Tage)</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>h</td>
<td>h</td>
<td>Stunde (Stunden)</td>
<td>f</td>
<td></td>
</tr>
<tr>
<td>min</td>
<td>min</td>
<td>Minute (Minuten)</td>
<td>f</td>
<td></td>
</tr>
<tr>
<td>s</td>
<td>s</td>
<td>Sekunde (Sekunden)</td>
<td>f</td>
<td></td>
</tr>
<tr>
<td>m</td>
<td>m</td>
<td>Meter</td>
<td>n</td>
<td></td>
</tr>
<tr>
<td>dm</td>
<td>dm</td>
<td>Dezimeter</td>
<td>n</td>
<td></td>
</tr>
<tr>
<td>cm</td>
<td>cm</td>
<td>Centimeter</td>
<td>n</td>
<td></td>
</tr>
<tr>
<td>mm</td>
<td>mm</td>
<td>Millimeter</td>
<td>n</td>
<td></td>
</tr>
<tr>
<td>in</td>
<td>in</td>
<td>Zoll</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>Deziliter</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>Kalorie (Kalorien)</td>
<td>f</td>
<td></td>
</tr>
<tr>
<td>kcal</td>
<td>kcal</td>
<td>Kilokalorie (Kilokalorien)</td>
<td>f</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>Elektronenvolt</td>
<td>n</td>
<td></td>
</tr>
<tr>
<td>pn</td>
<td>Prise</td>
<td>Prise (Prisen)</td>
<td>f</td>
<td></td>
</tr>
<tr>
<td>EL</td>
<td>EL</td>
<td>Esslöffel</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>TL</td>
<td>TL</td>
<td>Teelöffel</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>csp</td>
<td>KL</td>
<td>Mokkalöffel</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>dsp</td>
<td>dsp.</td>
<td>—</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>ssp</td>
<td>ssp.</td>
<td>—</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>Msp</td>
<td>Msp.</td>
<td>Messerspitze (Messerspitzen)</td>
<td>f</td>
<td></td>
</tr>
<tr>
<td>decimal-mark</td>
<td>—</td>
<td>,</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>one(m)</td>
<td>—</td>
<td>ein</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>one(f)</td>
<td>—</td>
<td>eine</td>
<td>m</td>
<td></td>
</tr>
<tr>
<td>one(n)</td>
<td>—</td>
<td>ein</td>
<td>m</td>
<td></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>n</td>
<td></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>décagramme</td>
<td>(décagrammes)</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>once</td>
<td></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>Re</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></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>électron-volt</td>
<td>(électron-volts)</td>
<td>m</td>
</tr>
<tr>
<td>pn</td>
<td>pinch</td>
<td>pincée</td>
<td></td>
<td>f</td>
</tr>
<tr>
<td>EL</td>
<td>c.à.s.</td>
<td>cuillère à soupe</td>
<td></td>
<td>f</td>
</tr>
<tr>
<td>TL</td>
<td>c.à.c.</td>
<td>cuillère à café</td>
<td></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 Quart
1 gallon = 8 Pint
1 gallon = 32 Gill
1 gallon = 160 fl. oz
1 fl. oz = 0.0284130625 liter
1 U.S. gallon = 231 inch³
1 U.S. gallon = 4 Quart
1 U.S. gallon = 8 Pint
1 U.S. gallon = 32 Gill
1 U.S. gallon = 128 fl. oz
1 fl. oz U.S. = 0.0295735295625 liter

Note 1: I think the American fl. oz U.S. is more common. Maybe. Most bottles have something like 10 fl. oz, which they say is equal to 30 mL. This would work really well with 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 “flouz”, “floiz” and “floez”? “flouz” 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 = \frac{1}{14} st
1 lb = \frac{17}{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
1 cup U.S. = 8 fl. oz U.S.
1 tablespoon U.S. = \frac{1}{2} fl. oz U.S.
1 teaspoon U.S. = \frac{1}{6} fl. oz U.S.

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 10.

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 \cufrac by \cuam

2016/09/09
General: \@@_calculate_input_and_store_in:nN optimiert durch neue property-key: single.
Add ‘single’ to property list of singlekeys.
Changed name from \@@_cunum_parse_range (and derivatives) to \@@_cutext_parse_range.
Changed name from \@@_parse_fraction_in_input:www to \@@_parse_mixed_fraction_in_input:www.
Corrected mistake: ‘ELektronenvolt’ (note uppercase L) to ‘Elektronenvolt’ in german.

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
New option: ‘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 \cuam 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.

2016/10/30
General: Fractions should now deal correctly with minus signs.

2016/11/07
General: Finished writing v1.10.

2016/11/13
General: \( \text{cutext}, \text{Cutext} \) and \( \text{cuam} \) check their input, allows conversion of units.

2017/03/10
General: \texttt{\textbackslash curlref} is now defined by \texttt{\textbackslash NewExpandableDocumentCommand} instead of the Declare variant.

2017/10/23
General: Added “phrases”.

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

2018/04/20

2018/06/05

44
This version introduces major 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

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> .......................... 19
<unit> ................................ 18

@@ commands:
\@@_calculate_input_and_store_in:nN .................. 43
\@@_cunum_parse_range .................. 43
\@@_cutext_default:nnn .................. 43
\@@_cutext_parse_range .................. 43
\@@_parse_fraction_in_input:ww ....... 43
\@@_parse_mixed_fraction_in_:ww ....... 43

\cutext ...., 3, 4, 6, 14, 18, 20–24, 28, 36
\cuaddkeys ................................ 13
\cuaddoptionfor ....................... 17, 20
\cuaddsinglekeys ...................... 9, 12
\cuaddtokeychain ...................... 9, 12, 13, 45
\cuaddtokeys ......................... 13
\cuaddtounitgroup .................... 19, 20
\cuam ......................... 3, 16, 21, 22, 24, 29
\curefoptionfor ...................... 17, 20
\cudeclareunitgroup ............. 19
\cudefinekeychain ................... 9, 10, 12, 13, 45
\cudefinekeys ....................... 13
\cudefinename ....................... 14
\cudefinephrase ...................... 9, 10, 12
\cudefinesinglekey ................... 9, 10, 12

\cudefinesymbol ..................... 15
\culabel .......................... 5, 30
\cunum ............. 3, 4, 10, 20, 22, 24, 29
\curef .......................... 5, 30, 31
\cusetoptionfor .................... 6, 17, 20, 25
\cusetup .......................... 3, 17
\cutext ...., 3, 4, 6, 14, 18, 20–24, 28, 36
\declarecookingderivatives ........ 9
\declarecookingunit ............... 7, 8
\newcookingunit .................... 7, 8
\providecookingunit ............... 7, 8

Numbers

42 ............................... 33

A
add-natural-unit ............. 33
add-option-for .................. 20
add-option-for<unit-key> ....... 20
add-special-sign ................ 22
add-temperature-to-check ..... 32
add-to-group .................... 8
add-unit-to-group .............. 20
amount-unit-space .......... 29

C
check-temperature .............. 32
\command .......................... 27
<table>
<thead>
<tr>
<th>Command</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>set-number-of-persons</td>
<td>30</td>
</tr>
<tr>
<td>set-option</td>
<td>8</td>
</tr>
<tr>
<td>set-option-for</td>
<td>20</td>
</tr>
<tr>
<td>set-option-for-unit-key</td>
<td>20</td>
</tr>
<tr>
<td>set-special-sign</td>
<td>22</td>
</tr>
<tr>
<td>set-unknown-message</td>
<td>22</td>
</tr>
<tr>
<td>symbol</td>
<td>8</td>
</tr>
<tr>
<td>\translate</td>
<td>44</td>
</tr>
<tr>
<td>use-fmtcount-numerals</td>
<td>18</td>
</tr>
<tr>
<td>use-numerals-below</td>
<td>23</td>
</tr>
<tr>
<td>use-phrases</td>
<td>25</td>
</tr>
<tr>
<td>usepackage</td>
<td>18</td>
</tr>
<tr>
<td>\thinspace</td>
<td>42</td>
</tr>
<tr>
<td>xD-lol</td>
<td>33</td>
</tr>
</tbody>
</table>