Units Conversion

Edition 1.32 for `units` Version 1.54

Adrian Mariano

Copyright © 1996, 1997 Free Software Foundation, Inc

Permission is granted to make and distribute verbatim copies of this manual provided the copyright notice and this permission notice are preserved on all copies.

Permission is granted to copy and distribute modified versions of this manual under the conditions for verbatim copying, provided that the entire resulting derived work is distributed under the terms of a permission notice identical to this one.

Permission is granted to copy and distribute translations of this manual into another language, under the above conditions for modified versions, except that this permission notice may be stated in a translation approved by the Foundation.

1 Overview of `units`

The `units` program converts quantities expressed in various scales to their equivalents in other scales. The `units` program can only handle multiplicative scale changes. For example, it cannot convert Celsius to Fahrenheit but it can convert temperature differences between those temperature scales.

The units are defined in an external data file. You can use the extensive data file that comes with this program, or you can provide your own data file to suit your needs.

You can use the program interactively with prompts, or you can use it from the command line.

2 Interacting with `units`

To invoke units for interactive use, type *`units`* at your shell prompt. The program will print something like this:

```
1161 units, 53 prefixes

You have:
```

At the '`You have:`' prompt, type the quantity and units that you are converting *from*. For example, if you want to convert ten meters to feet, type *`10 meters`*. Next, `units` will print '`You want:`'. You should type the type of units you want to convert *to*. To convert to feet, you would type *`feet`*.

The answer will be displayed in two ways. The first line of output, which is marked with a '`*`' to indicate multiplication, gives the result of the conversion you have asked for. The second line of output, which is marked with a '`/`' to indicate division, gives the inverse of the conversion factor. If you convert 10 meters to feet, `units` will print

```
* 32.808399
/ 0.03048
```

which tells you that 10 meters equals about 32.8 feet. The second number gives the conversion in the reverse directions. In this case, it tells you that 1 foot is equal to about 0.03 dekameters since the dekameter is 10 meters. It also tells you that 1/32.8 is about .03.

The `units` program prints the inverse because sometimes it is a more convenient number. In the example above, for example, the inverse value is an exact conversion: a foot is exactly .03048 dekameters. But the number given the other direction is inexact.

If you try to convert grains to pounds, you will see the following:

```
You have: grains
You want: pounds
        * 0.00014285714
        / 7000
```

From the second line of the output you can immediately see that a grain is equal to a seven thousandth of a pound. This is not so obvious from the first line of the output. If you find the output format confusing, try using the '`--verbose`' option:

```
You have: grain
You want: aeginamina
        grain = 0.00010416667 aeginamina
        grain = (1 / 9600) aeginamina
```

If you request a conversion between units which measure reciprocal dimensions, then `units` will display the conversion results with an extra note indicating that reciprocal conversion has been done:

```
You have: 6 ohms
You want: siemens
        reciprocal conversion
        * 0.16666667
        / 6
```

Reciprocal conversion can be suppressed by using the '`--strict`' option. As usual, use the '`--verbose`' option to get more comprehensible output:

```
You have: tex
You want: typp
        reciprocal conversion
        1 / tex = 496.05465 typp
        1 / tex = (1 / 0.0020159069) typp

You have: 20 mph
You want: sec/mile
        reciprocal conversion
        1 / 20 mph = 180 sec/mile
        1 / 20 mph = (1 / 0.0055555556) sec/mile
```

If you enter incompatible unit types, the `units` program will print a message indicating that the units are not conformable and it will display the reduced form for each unit:

```
You have: ergs/hour
You want: fathoms kg^2 / day
conformability error
        2.7777778e-11 kg m^2 / sec^3
        2.1166667e-05 kg^2 m / sec
```

If you only want to find the reduced form or definition of a unit, simply press return at the '`You want:`' prompt. Here is an example:

```
You have: jansky
You want:
        Definition: fluxunit = 1e-26 W/m^2 Hz = 1e-26 kg / s^2
```

The output from `units` indicates that the jansky is defined to be equal to a fluxunit which in turn is defined to be a certain combination of watts, meters, and hertz. The fully reduced (and in this case somewhat more cryptic) form appears on the far right.

3 Using `units` non-interactively

The `units` program can perform units conversions non-interactively from the command line. To do this, type the command, type the original units expression, and type the new units you want. You will probably need to protect the units expressions from interpretation by the shell using single quote characters.

If you type

```
units '2 liters' 'quarts'
```

then `units` will print

```
    * 2.1133764
    / 0.47317647
```

and then exit. The output tells you that 2 liters is about 2.1 quarts, or alternatively that a quart is about 0.47 times 2 liters.

If the conversion is successful, then `units` will return success to the calling environment. If `units` is given non-conformable units to convert, it will print a message giving the reduced form of each unit and it will return failure to the calling environment.

When `units` is invoked with only one argument, it will print out the definition of the specified unit. It will return failure if the unit is not defined and success if the unit is defined.

4 Unit expressions

In order to enter more complicated units or fractions, you will need to use operations such as powers, products and division. Powers of units can be specified using the '`^`' character as shown in the following example, or by simple concatenation: '`cm3`' is equivalent to '`cm^3`'. If the exponent is more than one digit, the '`^`' is required.

```
You have: cm^3
You want: gallons
        * 0.00026417205
        / 3785.4118

You have: arabicfoot-arabictradepound-force
You want: ft lbf
        * 0.7296
        / 1.370614
```

Multiplication of units can be specified by using spaces, a hyphen ('`-`') or an asterisk ('`*`'). Division of units is indicated by the slash ('`/`').

```
You have: furlongs/fortnight
You want: m/s
        * 0.00016630986
        / 6012.8727
```

Multiplication has a higher precedence than division and is evaluated left to right, so '`m/s * s/day`' is equivalent to '`m / s s day`' and has dimensions of length per time cubed. In effect, the first '`/`' character marks the beginning of the denominator of your unit. In particular, this means that writing '`1/2 meter`' refers to a unit of reciprocal length equivalent to .5/meter, which is probably not what you would intend if you entered that expression. To indicate division of numbers, use the vertical dash ('`|`'). No spaces area permitted on either side of the vertical dash character.

```
You have: 1|2 inch
You want: cm
        * 1.27
        / 0.78740157
```

Prefixes are defined separately from base units. In order to get centimeters, the units database defines '`centi-`' and '`c-`' as prefixes. Prefixes can appear alone with no unit following them. An exponent applies only to the immediately preceding unit and its prefix so that '`cm^3`' or '`centimeter^3`' refer to cubic centimeters but '`centi-meter^3`' refers to hundredths of cubic meters. Only one prefix is permitted per unit, so '`micromicrofarad`' will fail, but '`micro-microfarad`' will work.

For `units`, numbers are just another kind of unit. They can appear as many times as you like and in any order in a unit expression. For example, to find the volume of a box which is 2 ft by 3 ft by 12 ft in steres, you could do the following:

```
You have: 2 ft 3 ft 12 ft
You want: stere
        * 2.038813
        / 0.49048148

You have: $ 5 / yard
You want: cents / inch
        * 13.888889
        / 0.072
```

And the second example shows how the dollar sign in the units conversion can precede the five. Be careful: `units` will interpret '`$5`' with no space as equivalent to dollars^5.

Outside of the SI system, it is often desirable to add values of different units together. Sums of conformable units are written with the '`+`' character.

```
You have: 2 hours + 23 minutes + 32 seconds
You want: seconds
        * 8612
        / 0.00011611705

You have: 12 ft + 3 in
You want: cm
        * 373.38
        / 0.0026782366

You have: 2 btu + 450 ft-lbf
You want: btu
        * 2.5782804
        / 0.38785542
```

The expressions which are added together must reduce to identical expressions in primitive units, or an error message will be displayed:

```
You have: 12 printerspoint + 4 heredium
Illegal sum of non-conformable units:
        12 printerspoint reduces to 0.0042175176 m
        4 heredium reduces to 20145.828 m^2
```

Because '`-`' is used for products, it cannot also be used to form differences of units. If a '`-`' appears before numerical digits as the very first character on the input line or if it appears immediately after a '+' then the number will be evaluated as a negative number. So you can compute 20 degrees minus 12 minutes by entering '`20 degrees+-12 arcmin`'. The '+' character is sometimes used in exponents like '`3.43e+8`'. Exponents of this form cannot be used when forming sums of units, but they may be used otherwise.

5 Invoking `units`

You invoke `units` like this:

`units` *options* `[`*from-unit* `[`*to-unit*`]]`

If the *from-unit* and *to-unit* are omitted, then the program will use interactive prompts to determine which conversions to perform. See Chapter 2 [Interactive use], page 1. If both *from-unit* and *to-unit* are given, `units` will print the result of that single conversion and then exit. If only *from-unit* appears on the command line, `units` will display the definition of that unit and exit. Units specified on the command line will need to be quoted to protect them from shell interpretation and to group them into two arguments. See Chapter 3 [Command line use], page 2.

The following options allow you to read in an alternative units file, check your units file, or change the output format:

'`-c`'
'`--check`'
: Check that all units and prefixes defined in the units file reduce to primitive units. The program will print a list of all units that cannot be reduced.

'`--check-verbose`'
: Like the '`-check`' option, this option prints a list of units that cannot be reduced. But to help find unit definitions that cause endless loops, it lists the units as they are checked. If `units` hangs, then the last unit to be printed has a bad definition.

'`-o format`'
'`--output-format format`'
: Use the specified format for numeric output. Format is the same as that for the printf function in the ANSI C standard. For example, if you want more precision you might use '`-o %.15g`'.

'`-f filename`'
'`--file filename`'
: Use filename as the units data file rather than the default units data file.

'`-h`'
'`--help`'
: Print out a summary of the options for `units`.

'`-q`'
'`--quiet`'
'`--silent`'
: Suppress prompting of the user for units and the display of statistics about the number of units loaded.

'`-s`'
'`--strict`'
: Suppress conversion of units to their reciprocal units.

'-v'
'--verbose'
: Give slightly more verbose output when converting units. When combined with the '-c' option this gives the same effect as '--check-verbose'.

'-V'
'--version'
: Print program version number, tell whether the readline library has been included, and give the location of the default units data file.

6 Unit definitions

The conversion information is read from a units data file which is called 'units.dat' and is probably located in the '/usr/local/share' directory. If you invoke `units` with the '-V' option, it will print the location of this file. The default file includes definitions for all familiar units, abbreviations and metric prefixes. It also includes many obscure or archaic units.

Many constants of nature are defined, including these:

`pi`	ratio of circumference to diameter
`c`	speed of light
`e`	charge on an electron
`force`	acceleration of gravity
`mole`	Avogadro's number
`water`	pressure per unit height of water
`Hg`	pressure per unit height of mercury
`au`	astronomical unit
`k`	Boltzman's constant
`mu0`	permeability of vacuum
`epsilon0`	permitivity of vacuum
`G`	Gravitational constant
`mach`	speed of sound

The database includes atomic masses for all of the elements and numerous other constants. Also included are the densities of various ingredients used in baking so that '2 cups flour_sifted' can be converted to 'grams'. This is not an exhaustive list. Consult the units data file to see the complete list, or to see the definitions that are used.

The unit 'pound' is a unit of mass. To get force, multiply by the force conversion unit 'force' or use the shorthand 'lbf'. (Note that 'g' is already taken as the standard abbreviation for the gram.) The unit 'ounce' is also a unit of mass. The fluid ounce is 'fluidounce' or 'floz'. British capacity units that differ from their US counterparts, such as the British Imperial gallon, are prefixed with 'br'. Currency is prefixed with its country name: 'belgiumfranc', 'britainpound'.

The US Survey foot, yard, and mile can be obtained by using the 'US' prefix. These units differ slightly from the international length units. They were in use until 1959, but for geographic surveys, they are still used. The acre is officially defined in terms of the US Survey foot. If you want an acre defined according to the international foot, use 'intacre'. The difference between these units is about 4 parts per million. The British also used a slightly different length measure before 1959. These can be obtained with the prefix 'UK'.

When searching for a unit, if the specified string does not appear exactly as a unit name, then the `units` program will try to remove a trailing '`s`' or a trailing '`es`'. If that fails, `units` will check for a prefix. All of the standard metric prefixes are defined.

To find out what units and prefixes are available, read the standard units data file.

7 Defining new units

All of the units and prefixes that `units` can convert are defined in the units data file. If you want to add your own units, you can supply your own file.

A unit is specified on a single line by giving its name and an equivalence. Comments start with a '`#`' character, which can appear anywhere in a line. Be careful to define new units in terms of old ones so that a reduction leads to the primitive units, which are marked with '`!`' characters. When adding new units, be sure to use the '`-c`' option to check that the new units reduce properly. If you define any units which contain '`+`' characters, carefully check them because the '`-c`' option will not catch non-conformable sums. If you create a loop in the units definitions, then `units` will hang when invoked with the '`-c`' options. You will need to use the '`--check-verbose`' option which prints out each unit as it checks them. The program will still hang, but the last unit printed will be the unit which caused the infinite loop.

Here is an example of a short units file that defines some basic units:

```
m       !               # The meter is a primitive unit
sec     !               # The second is a primitive unit
micro-  1e-6            # Define a prefix
minute  60 sec          # A minute is 60 seconds
hour    60 min          # An hour is 60 minutes
inch    0.0254 m        # Inch defined in terms of meters
ft      12 inches       # The foot defined in terms of inches
mile    5280 ft         # And the mile
```

A unit which ends with a '`-`' character is a prefix. If a prefix contains a '`/`' character, then when it is applied to a unit, that unit will be in the denominator. So the definition '`half- 1/2`' would give a '`halfmeter`' equal to '`1 / 2 meter`'. For this reason, the '`/`' character should be avoided in prefixes.

8 Readline support

If the `readline` package has been compiled in, then when `units` is used interactively, numerous command line editing features are available. To check if your version of `units` includes the readline, invoke the program with the '`--version`' option.

For complete information about readline, consult the documentation for the readline package. Without any configuration, `units` will allow editing in the style of emacs. Of particular use with `units` are the completion commands.

If you type a few characters and then hit '`ESC`' followed by the *`?`* key then `units` will display a list of all the units which start with the characters typed. For example, if you type *`metr`* and then request completion, you will see something like this:

```
You have: metr
metre            metriccup        metrichorsepower  metrictenth
metretes         metricfifth      metricounce       metricton
metriccarat      metricgrain      metricquart       metricyarncount
You have: metr
```

If there is a unique way to complete a unitname, you can hit the tab key and `units` will provide the rest of the unit name. If `units` beeps, it means that there is no unique completion. Pressing the tab key a second time will print the list of all completions.

Index

(Index is nonexistent)

Table of Contents