

NAME

rrdgraph_graph – rrdtool graph command reference

SYNOPSIS

PRINT:*vname:format[:strftime|:valstrftime|:valstrfduration]*

GPRINT:*vname:format*

COMMENT:*text*

VRULE:*time#color[:[legend][:dashes[=on_s[,off_s[,on_s,off_s]...]][:dash–offset=offset]]]*

HRULE:*value#color[:[legend][:dashes[=on_s[,off_s[,on_s,off_s]...]][:dash–offset=offset]]]*

LINE[*width*]:*value[#color[:[legend][:STACK][:skipscale][:dashes[=on_s[,off_s[,on_s,off_s]...]][:dash–offset=offset]]]*

AREA:*value[#color[:[legend][:STACK][:skipscale]]]*

TICK:*vname#rrggb[aa][:fraction[:legend]]*

SHIFT:*vname:offset*

TEXTALIGN:{*left|right|justified|center*}

PRINT:*vname:CF:format* (deprecated)

GPRINT:*vname:CF:format* (deprecated)

STACK:*vname#color[:legend]* (deprecated)

DESCRIPTION

These instructions allow you to generate your image or report. If you don't use any graph elements, no graph is generated. Similarly, no report is generated if you don't use print options.

PRINT

PRINT:*vname:format[:strftime|:valstrftime|:valstrfduration]*

Depending on the context, either the value component (no suffix, *valstrftime* or *valstrfduration*) or the time component (*strftime*) of a **VDEF** is printed using *format*. It is an error to specify a *vname* generated by a **DEF** or **CDEF**.

Any text in *format* is printed literally with one exception: The percent character introduces a formatter string. This string can be:

For printing values:

% %

just prints a literal '%' character

%#.#le

prints numbers like 1.2346e+04. The optional integers # denote field width and decimal precision.

%#.#lf

prints numbers like 12345.6789 (%5.4lf), with optional field width and precision.

%s place this after **%le**, **%lf** or **%lg**. This will be replaced by the appropriate SI magnitude unit and the value will be scaled accordingly (123456 → 123.456 k).

%S is similar to **%s**. It does, however, use a previously defined magnitude unit. If there is no such unit yet, it tries to define one (just like **%s**) unless the value is zero, in which case the magnitude unit stays undefined. Thus, formatter strings using **%S** and no **%s** will all use the same magnitude unit except for zero values.

If you **PRINT** a **VDEF** value, you can also print the time associated with it by appending the string **:strftime** to the format. Note that RRDtool uses the *strftime* function of your OS's C library. This means that the conversion specifier may vary. Check the manual page if you are uncertain. The following is a list of conversion specifiers usually supported across the board. Formatting values interpreted as timestamps with **:valstrftime** is done likewise.

- %a** The abbreviated weekday name according to the current locale.
- %A** The full weekday name according to the current locale.
- %b** The abbreviated month name according to the current locale.
- %B** The full month name according to the current locale.
- %c** The preferred date and time representation for the current locale.
- %d** The day of the month as a decimal number (range 01 to 31).
- %H** The hour as a decimal number using a 24-hour clock (range 00 to 23).
- %I** The hour as a decimal number using a 12-hour clock (range 01 to 12).
- %j** The day of the year as a decimal number (range 001 to 366).
- %m** The month as a decimal number (range 01 to 12).
- %M** The minute as a decimal number (range 00 to 59).
- %p** Either 'AM' or 'PM' according to the given time value, or the corresponding strings for the current locale. Noon is treated as 'pm' and midnight as 'am'. Note that in many locales a 'pm' notation is unsupported and in such cases %p will return an empty string.
- %s** The second as a decimal number (range 00 to 61).
- %S** The seconds since the epoch (1.1.1970) (libc dependent non standard!)
- %U** The week number of the current year as a decimal number, range 00 to 53, starting with the first Sunday as the first day of week 01. See also %V and %W.
- %V** The ISO 8601:1988 week number of the current year as a decimal number, range 01 to 53, where week 1 is the first week that has at least 4 days in the current year, and with Monday as the first day of the week. See also %U and %W.
- %w** The day of the week as a decimal, range 0 to 6, Sunday being 0. See also %u.
- %W** The week number of the current year as a decimal number, range 00 to 53, starting with the first Monday as the first day of week 01.
- %x** The preferred date representation for the current locale without the time.
- %X** The preferred time representation for the current locale without the date.
- %y** The year as a decimal number without a century (range 00 to 99).
- %Y** The year as a decimal number including the century.
- %Z** The time zone or name or abbreviation.
- %%**
A literal '%' character.

You can format values as duration using **%valstrfduration** with the value being interpreted as milliseconds. Using printf like conversion specifications you can extract properties from the duration:

- All non-conversion specification chars are copied unchanged
- A conversion specification has format '%' [['0'] minwidth] ['.' precision]

With conversion-specifier being one of:

- %** A raw '%' is output, width and precision are ignored
- W** Number of weeks
- d** Number of days, modulus number of weeks
- D** Number of days

h Number of hours, modulus number of days
H Number of hours
m Number of minutes, modulus number of hours
M Number of minutes
s Number of seconds, modulus number of minutes
S Number of seconds
f Number of milliseconds, modulus seconds

PRINT:*vname*:*CF*:*format*

Deprecated. Use the new form of this command in new scripts. The first form of this command is to be used with **CDEF** *vnames*.

GRAPH

GPRINT:*vname*:*format*

This is the same as **PRINT**, but printed inside the graph.

GPRINT:*vname*:*CF*:*format*

Deprecated. Use the new form of this command in new scripts. This is the same as **PRINT**, but printed inside the graph.

COMMENT:*text*

Text is printed literally in the legend section of the graph. Note that in RRDtool 1.2 you have to escape colons in **COMMENT** text in the same way you have to escape them in ***PRINT** commands by writing '\:'.

VRULE:*time*#*color*[:*legend*][:*dashes*[=*on_s*[,*off_s*[,*on_s*[,*off_s*]...]][:*dash-offset*=*offset*]]

Draw a vertical line at *time*. Its color is composed from three hexadecimal numbers specifying the rgb color components (00 is off, FF is maximum) red, green and blue followed by an optional alpha. Optionally, a legend box and string is printed in the legend section. *time* may be a number or a variable from a **VDEF**. It is an error to use *vnames* from **DEF** or **CDEF** here. Dashed lines can be drawn using the **dashes** modifier. See **LINE** for more details.

HRULE:*value*#*color*[:*legend*][:*dashes*[=*on_s*[,*off_s*[,*on_s*[,*off_s*]...]][:*dash-offset*=*offset*]]

Draw a horizontal line at *value*. **HRULE** acts much like **LINE** except that will have no effect on the scale of the graph. If a **HRULE** is outside the graphing area it will just not be visible and it will not appear in the legend by default.

LINE[*width*]:*value*#[*color*][:*legend*][:**STACK**][:*skip scale*][:*dashes*[=*on_s*[,*off_s*[,*on_s*[,*off_s*]...]][:*dash-offset*=*offset*]]

Draw a line of the specified width onto the graph. *width* can be a floating point number. If the color is not specified, the drawing is done 'invisibly'. This is useful when stacking something else on top of this line. Also optional is the legend box and string which will be printed in the legend section if specified. The **value** can be generated by **DEF**, **VDEF**, and **CDEF**. If the optional **STACK** modifier is used, this line is stacked on top of the previous element which can be a **LINE** or an **AREA**.

Normally the graphing function makes sure that the entire **LINE** or **AREA** is visible in the chart. The scaling of the chart will be modified accordingly if necessary. Any **LINE** or **AREA** can be excluded from this process by adding the option **skip scale**.

The **dashes** modifier enables dashed line style. Without any further options a symmetric dashed line with a segment length of 5 pixels will be drawn. The dash pattern can be changed if the **dashes=** parameter is followed by either one value or an even number (1, 2, 4, 6, ...) of positive values. Each value provides the length of alternate *on_s* and *off_s* portions of the stroke. The **dash-offset** parameter specifies an *offset* into the pattern at which the stroke begins.

When you do not specify a color, you cannot specify a legend. Should you want to use **STACK**, use the "LINEx:<value>::STACK" form.

AREA:*value*[*#color*[*#color2*]][:*legend*][:**STACK**][:*skip scale*][:*gradheight=y*]

See **LINE**, however the area between the x-axis and the line will be filled.

If *color2* is specified, the area will be filled with a gradient.

The *gradheight* parameter can create three different behaviors. If *gradheight* > 0, then the gradient is a fixed height, starting at the line going down. If *gradheight* < 0, then the gradient starts at a fixed height above the x-axis, going down to the x-axis. If *height* == 0, then the gradient goes from the line to x-axis.

The default value for *gradheight* is 50.

TICK:*vname**#rrgbb[aa]*[:*fraction*][:*legend*]

Plot a tick mark (a vertical line) for each value of *vname* that is non-zero and not *UNKNOWN*. The *fraction* argument specifies the length of the tick mark as a fraction of the y-axis; the default value is 0.1 (10% of the axis). Note that the color specification is not optional. The **TICK** marks normally start at the lower edge of the graphing area. If the fraction is negative they start at the upper border of the graphing area.

SHIFT:*vname:offset*

Using this command **RRDtool** will graph the following elements with the specified offset. For instance, you can specify an offset of (7*24*60*60 =) 604'800 seconds to "look back" one week. Make sure to tell the viewer of your graph you did this ... As with the other graphing elements, you can specify a number or a variable here.

TEXTALIGN:{*left*|*right*|*justified*|*center*}

Labels are placed below the graph. When they overflow to the left, they wrap to the next line. By default, lines are justified left and right. The **TEXTALIGN** function lets you change this default. This is a command and not an option, so that you can change the default several times in your argument list.

STACK:*vname**#color*[:*legend*]

Deprecated. Use the STACK modifiers on the other commands instead!

Some notes on stacking

When stacking, an element is not placed above the X-axis but rather on top of the previous element. There must be something to stack upon.

You can use an **invisible** **LINE** or **AREA** to stacked upon.

An **unknown** value makes the entire stack unknown from that moment on. You don't know where to begin (the unknown value) and therefore do not know where to end.

If you want to make sure you will be displaying a certain variable, make sure never to stack upon the unknown value. Use a CDEF instruction with **IF** and **UN** to do so.

NOTES on legend arguments

Escaping the colon

A colon ':' in a *legend* argument will mark the end of the legend. To enter a ':' as part of a legend, the colon must be escaped with a backslash '\:'. Beware that many environments process backslashes themselves, so it may be necessary to write two backslashes in order to one being passed onto rrd_graph.

String Formatting

The text printed below the actual graph can be formatted by appending special escape characters at the end of a text. Whenever such a character occurs, all pending text is pushed onto the graph according to the character specified.

Valid markers are: **\j** for justified, **\l** for left aligned, **\r** for right aligned, and **\c** for centered. In the next section there is an example showing how to use centered formatting.

\n is a valid alias for **\l** since incomplete parsing in earlier versions of RRDtool lead to this behavior and a number of people has been using it.

Normally there are two space characters inserted between every two items printed into the graph. The space

following a string can be suppressed by putting a `\g` at the end of the string. The `\g` also ignores any space inside the string if it is at the very end of the string. This can be used in connection with `%s` to suppress empty unit strings.

```
GPRINT:a:MAX:%lf%s\g
```

A special case is `COMMENT:\s` which inserts some additional vertical space before placing the next row of legends.

If you want to have left and right aligned legends on the same line use `COMMENT:\u` to go one line back like this:

```
COMMENT:left\l
COMMENT:\u
COMMENT:right\r
```

There is also a 'nop' control for situations where you want a string to actually end in a backslash character sequence `\`.

```
COMMENT:OS\2\.
```

When using a proportional font in your graph, the tab characters or the sequence `\t` will line-up legend elements. Note that the tabs inserted are relative to the start of the current legend element!

Since RRDtool 1.3 is using Pango for rendering text, you can use Pango markup. Pango uses the xml **span** tags for inline formatting instructions.

A simple example of a marked-up string might be:

```
<span foreground="blue" size="x-large">Blue text</span> is <i>cool</i>!
```

The complete list of attributes for the span tag (taken from the pango documentation):

font_desc

A font description string, such as "Sans Italic 12"; note that any other span attributes will override this description. So if you have "Sans Italic" and also a `style="normal"` attribute, you will get Sans normal, not italic.

font_family

A font family name

face

Synonym for `font_family`

size

Font size in 1024ths of a point, or one of the absolute sizes 'xx-small', 'x-small', 'small', 'medium', 'large', 'x-large', 'xx-large', or one of the relative sizes 'smaller' or 'larger'. If you want to specify an absolute size, it's usually easier to take advantage of the ability to specify a partial font description using 'font_desc'; you can use `font_desc='12.5'` rather than `size='12800'`.

style

One of 'normal', 'oblique', 'italic'

weight

One of 'ultralight', 'light', 'normal', 'bold', 'ultrabold', 'heavy', or a numeric weight

variant

'normal' or 'smallcaps'

stretch

One of 'ultracondensed', 'extracondensed', 'condensed', 'semicondensed', 'normal', 'semiexpanded', 'expanded', 'extraexpanded', 'ultraexpanded'

foreground

An RGB color specification such as '#00FF00' or a color name such as 'red'

background

An RGB color specification such as '#00FF00' or a color name such as 'red'

underline

One of 'none', 'single', 'double', 'low', 'error'

underline_color

The color of underlines; an RGB color specification such as '#00FF00' or a color name such as 'red'

rise

Vertical displacement, in 10000ths of an em. Can be negative for subscript, positive for superscript.

striketrough

'true' or 'false' whether to strike through the text

striketrough_color

The color of crossed out lines; an RGB color specification such as '#00FF00' or a color name such as 'red'

fallback

'true' or 'false' whether to enable fallback. If disabled, then characters will only be used from the closest matching font on the system. No fallback will be done to other fonts on the system that might contain the characters in the text. Fallback is enabled by default. Most applications should not disable fallback.

lang

A language code, indicating the text language

letter_spacing

Inter-letter spacing in 1024ths of a point.

gravity

One of 'south', 'east', 'north', 'west', 'auto'.

gravity_hint

One of 'natural', 'strong', 'line'.

To save you some typing, there are also some shortcuts:

b Bold

big Makes font relatively larger, equivalent to

i Italic

s Strike through

sub

Subscript

sup

Superscript

small

Makes font relatively smaller, equivalent to

tt Monospace font

u Underline

SEE ALSO

rrdgraph gives an overview of how **rrdtool graph** works. rrdgraph_data describes **DEF,CDEF** and **VDEF** in detail. rrdgraph_rpn describes the **RPN** language used in the **?DEF** statements. rrdgraph_graph page describes all of the graph and print functions.

Make sure to read rrdgraph_examples for tips&tricks.

AUTHOR

Program by Tobias Oetiker <tobi@oetiker.ch>

This manual page by Alex van den Bogaerd <alex@vandenbogaerd.nl> with corrections and/or additions by several people