NAME

rrdgraph − Round Robin Database tool graphing functions

SYNOPSIS

rrdtool graph|graphv *filename* [*option* ...] [*data definition* ...] [*data calculation* ...] [*variable definition* ...] [*graph element* ...] [*print element* ...]

DESCRIPTION

The **graph** function of **RRDtool** is used to present the data from an **RRD** to a human viewer. Its main purpose is to create a nice graphical representation, but it can also generate a numerical report.

OVERVIEW

rrdtool graph needs data to work with, so you must use one or more **data definition** statements to collect this data. You are not limited to one database, it's perfectly legal to collect data from two or more databases (one per statement, though).

If you want to display averages, maxima, percentiles, etcetera it is best to collect them now using the **variable definition** statement. Currently this makes no difference, but in a future version of RRDtool you may want to collect these values before consolidation.

The data fetched from the **RRA** is then **consolidated** so that there is exactly one data point per pixel in the graph. If you do not take care yourself, **RRDtool** will expand the range slightly if necessary. Note, in that case the first and/or last pixel may very well become unknown!

Sometimes data is not exactly in the format you would like to display it. For instance, you might be collecting **bytes** per second, but want to display **bits** per second. This is what the **data calculation** command is designed for. After **consolidating** the data, a copy is made and this copy is modified using a rather powerful **RPN** command set.

When you are done fetching and processing the data, it is time to graph it (or print it). This ends the **rrdtool graph** sequence.

Use **graphv** instead of **graph** to get detailed information about the graph geometry and data once it is drawn. See the bottom of the document for more information.

OPTIONS

filename

The name and path of the graph to generate. It is recommended to end this in `.png`, `.svg` or `.eps`, but **RRDtool** does not enforce this.

filename can be '`-`' to send the image to `stdout`. In this case, no other output is generated.

Time range

[**−s**|**−−start** *time*] [**−e**|**−−end** *time*] [**−S**|**−−step** *seconds*]

The start and end of the time series you would like to display, and which **RRA** the data should come from. Defaults are: 1 day ago until now, with the best possible resolution. **Start** and **end** can be specified in several formats, see AT-STYLE TIME SPECIFICATION and rrdgraph_examples. By default, **rrdtool graph** calculates the width of one pixel in the time domain and tries to get data from an **RRA** with that resolution. With the **step** option you can alter this behavior. If you want **rrdtool graph** to get data at a one-hour resolution from the **RRD**, set **step** to 3'600. Note: a step smaller than one pixel will silently be ignored.

Labels

[**−t**|**−−title** *string*] [**−v**|**−−vertical−label** *string*]

A horizontal string at the top of the graph and/or a vertically placed string at the left hand side of the graph.

Size

[**−w**|**−−width** *pixels*] [**−h**|**−−height** *pixels*] [**−j**|**−−only−graph**] [**−D**|**−−full−size−mode**]

By default, the width and height of the **canvas** (the part with the actual data and such). This defaults to 400 pixels by 100 pixels.

If you specify the **−−full−size−mode** option, the width and height specify the final dimensions of the output image and the canvas is automatically resized to fit.

If you specify the **--only-graph** option and set the height < 32 pixels you will get a tiny graph image (thumbnail) to use as an icon for use in an overview, for example. All labeling will be stripped off the graph.

Limits

[**-u**|**--upper-limit** *value*] [**-l**|**--lower-limit** *value*] [**-r**|**--rigid**]

By default the graph will be autoscaling so that it will adjust the y-axis to the range of the data. You can change this behavior by explicitly setting the limits. The displayed y-axis will then range at least from **lower-limit** to **upper-limit**. Autoscaling will still permit those boundaries to be stretched unless the **rigid** option is set.

[**-A**|**--alt-autoscale**]

Sometimes the default algorithm for selecting the y-axis scale is not satisfactory. Normally the scale is selected from a predefined set of ranges and this fails miserably when you need to graph something like `260 + 0.001 * sin(x)`. This option calculates the minimum and maximum y-axis from the actual minimum and maximum data values. Our example would display slightly less than `260-0.001` to slightly more than `260+0.001` (this feature was contributed by Sasha Mikheev).

[**-J**|**--alt-autoscale-min**]

Where `--alt-autoscale` will modify both the absolute maximum AND minimum values, this option will only affect the minimum value. The maximum value, if not defined on the command line, will be 0. This option can be useful when graphing router traffic when the WAN line uses compression, and thus the throughput may be higher than the WAN line speed.

[**-M**|**--alt-autoscale-max**]

Where `--alt-autoscale` will modify both the absolute maximum AND minimum values, this option will only affect the maximum value. The minimum value, if not defined on the command line, will be 0. This option can be useful when graphing router traffic when the WAN line uses compression, and thus the throughput may be higher than the WAN line speed.

[**-N**|**--no-gridfit**]

In order to avoid anti-aliasing blurring effects RRDtool snaps points to device resolution pixels, this results in a crisper appearance. If this is not to your liking, you can use this switch to turn this behavior off.

Grid-fitting is turned off for PDF, EPS, SVG output by default.

X-Axis

[**-x**|**--x-grid** *GTM*:*GST*:*MTM*:*MST*:*LTM*:*LST*:*LPR*:*LFM*]

[**-x**|**--x-grid none**]

The x-axis label is quite complex to configure. If you don't have very special needs it is probably best to rely on the auto configuration to get this right. You can specify the string `none` to suppress the grid and labels altogether.

The grid is defined by specifying a certain amount of time in the *?TM* positions. You can choose from `SECOND`, `MINUTE`, `HOUR`, `DAY`, `WEEK`, `MONTH` or `YEAR`. Then you define how many of these should pass between each line or label. This pair (*?TM:?ST*) needs to be specified for the base grid (*G??*), the major grid (*M??*) and the labels (*L??*). For the labels you also must define a precision in *LPR* and a *strftime* format string in *LFM*. *LPR* defines where each label will be placed. If it is zero, the label will be placed right under the corresponding line (useful for hours, dates etcetera). If you specify a number of seconds here the label is centered on this interval (useful for Monday, January etcetera).

```
--x-grid MINUTE:10:HOUR:1:HOUR:4:0:%X
```

This places grid lines every 10 minutes, major grid lines every hour, and labels every 4 hours. The labels are placed under the major grid lines as they specify exactly that time.

```
--x-grid HOUR:8:DAY:1:DAY:1:86400:%A
```

This places grid lines every 8 hours, major grid lines and labels each day. The labels are placed exactly

between two major grid lines as they specify the complete day and not just midnight.

[**−−week−fmt** *strftime format string*]

By default rrdtool uses ‘‘Week %V’’ to render the week number. With this option you can define your own format, without completely overriding the xaxis format.

Y−Axis

[**−y**|**−−y−grid** *grid step*:*label factor*]

[**−y**|**−−y−grid none**]

Y−axis grid lines appear at each *grid step* interval. Labels are placed every *label factor* lines. You can specify `−y none` to suppress the grid and labels altogether. The default for this option is to automatically select sensible values.

If you have set −−y−grid to ’none’ not only the labels get suppressed, also the space reserved for the labels is removed. You can still add space manually if you use the −−units−length command to explicitly reserve space.

[**−−left−axis−format** *format-string*]

By default the format of the axis labels gets determined automatically. If you want to do this your self, use this option with the same `%lf` arguments you know from the PRINT and GPRINT commands.

[**−Y**|**−−alt−y−grid**]

Place the Y grid dynamically based on the graph’s Y range. The algorithm ensures that you always have a grid, that there are enough but not too many grid lines, and that the grid is metric. That is the grid lines are placed every 1, 2, 5 or 10 units. This parameter will also ensure that you get enough decimals displayed even if your graph goes from 69.998 to 70.001. (contributed by Sasha Mikheev).

[**−o**|**−−logarithmic**]

Logarithmic y−axis scaling.

[**−X**|**−−units−exponent** *value*]

This sets the 10**exponent scaling of the y−axis values. Normally, values will be scaled to the appropriate units (k, M, etc.). However, you may wish to display units always in k (Kilo, 10e3) even if the data is in the M (Mega, 10e6) range, for instance. Value should be an integer which is a multiple of 3 between −18 and 18 inclusively. It is the exponent on the units you wish to use. For example, use 3 to display the y−axis values in k (Kilo, 10e3, thousands), use −6 to display the y−axis values in u (Micro, 10e−6, millionths). Use a value of 0 to prevent any scaling of the y−axis values.

This option is very effective at confusing the heck out of the default RRDtool autoscaling function and grid painter. If RRDtool detects that it is not successful in labeling the graph under the given circumstances, it will switch to the more robust **−−alt−y−grid** mode.

[**−L**|**−−units−length** *value*]

How many digits should RRDtool assume the y−axis labels to be? You may have to use this option to make enough space once you start fiddling with the y−axis labeling.

[**−−units=si**]

With this option y−axis values on logarithmic graphs will be scaled to the appropriate units (k, M, etc.) instead of using exponential notation. Note that for linear graphs, SI notation is used by default.

Right Y Axis

[**−−right−axis** *scale*:*shift*] [**−−right−axis−label** *label*]

A second axis will be drawn to the right of the graph. It is tied to the left axis via the scale and shift parameters. You can also define a label for the right axis.

[**−−right−axis−format** *format-string*]

By default the format of the axis labels gets determined automatically. If you want to do this your self, use

this option with the same `%lf` arguments you know from the PRINT and GPRINT commands.

Legend

[**−g**|**−−no−legend**]

Suppress generation of the legend; only render the graph.

[**−F**|**−−force−rules−legend**]

Force the generation of HRULE and VRULE legends even if those HRULE or VRULE will not be drawn because out of graph boundaries (mimics behavior of pre 1.0.42 versions).

[**−−legend−position**=(north|south|west|east)]

Place the legend at the given side of the graph. The default is south. In west or east position it is necessary to add line breaks manually.

[**−−legend−direction**=(topdown|bottomup|bottomup2)]

Place the legend items in the given vertical order. The default is topdown. Using bottomup the legend items appear in the same vertical order as a stack of lines or areas. Using bottomup2 will keep leading and trailing COMMENT lines in order, this might be useful for generators that use them for table headers and the like.

Miscellaneous

[**−z**|**−−lazy**]

Only generate the graph if the current graph is out of date or not existent. Note, that all the calculations will happen regardless so that the output of PRINT and graphv will be complete regardless. Note that the behavior of lazy in this regard has seen several changes over time. The only thing you can really rely on before RRDtool 1.3.7 is that lazy will not generate the graph when it is already there and up to date, and also that it will output the size of the graph.

[**−−daemon** *address*]

Address of the rrdcached daemon. If specified, a `flush` command is sent to the server before reading the RRD files. This allows the graph to contain fresh data even if the daemon is configured to cache values for a long time. For a list of accepted formats, see the **−l** option in the rrdcached manual.

```
rrdtool graph [...] --daemon unix:/var/run/rrdcached.sock [...]
```

[**−f**|**−−imginfo** *printfstr*]

After the image has been created, the graph function uses printf together with this format string to create output similar to the PRINT function, only that the printf function is supplied with the parameters *filename*, *xsize* and *ysize*. In order to generate an **IMG** tag suitable for including the graph into a web page, the command line would look like this:

```
--imginfo '<IMG SRC="/img/%s" WIDTH="%lu" HEIGHT="%lu" ALT="Demo">'
```

[**−c**|**−−color** *COLORTAG#rrggbb*[*aa*]]

Override the default colors for the standard elements of the graph. The *COLORTAG* is one of `BACK` background, `CANVAS` for the background of the actual graph, `SHADEA` for the left and top border, `SHADEB` for the right and bottom border, `GRID`, `MGRID` for the major grid, `FONT` for the color of the font, `AXIS` for the axis of the graph, `FRAME` for the line around the color spots, and finally `ARROW` for the arrow head pointing up and forward. Each color is composed out of three hexadecimal numbers specifying its rgb color component (00 is off, FF is maximum) of red, green and blue. Optionally you may add another hexadecimal number specifying the transparency (FF is solid). You may set this option several times to alter multiple defaults.

A green arrow is made by: `--color ARROW#00FF00`

[**−−grid−dash** *on*:*off*]

by default the grid is drawn in a 1 on, 1 off pattern. With this option you can set this yourself

```
--grid-dash 1:3    for a dot grid
```

```
--grid-dash 1:0    for uninterrupted grid lines
```

[**--border** *width*]

Width in pixels for the 3d border drawn around the image. Default 2, 0 disables the border. See SHADEA and SHADEB above for setting the border color.

[**--dynamic-labels**]

Pick the shape of the color marker next to the label according to the element drawn on the graph.

[**-m**|**--zoom** *factor*]

Zoom the graphics by the given amount. The factor must be > 0

[**-n**|**--font** *FONTTAG*:*size*[:*font*]]

This lets you customize which font to use for the various text elements on the RRD graphs. DEFAULT sets the default value for all elements, TITLE for the title, AXIS for the axis labels, UNIT for the vertical unit label, LEGEND for the graph legend, WATERMARK for the watermark on the edge of the graph.

Use Times for the title: `--font TITLE:13:Times`

Note that you need to quote the argument to **--font** if the font-name contains whitespace: --font "TITLE:13:Some Font"

If you do not give a font string you can modify just the size of the default font: `--font TITLE:13:`.

If you specify the size 0 then you can modify just the font without touching the size. This is especially useful for altering the default font without resetting the default fontsizes: `--font DEFAULT:0:Courier`.

RRDtool comes with a preset default font. You can set the environment variable RRD_DEFAULT_FONT if you want to change this.

RRDtool uses Pango for its font handling. This means you can to use the full Pango syntax when selecting your font:

The font name has the form "[*FAMILY-LIST*] [*STYLE-OPTIONS*] [*SIZE*]", where *FAMILY-LIST* is a comma separated list of families optionally terminated by a comma, *STYLE_OPTIONS* is a whitespace separated list of words where each WORD describes one of style, variant, weight, stretch, or gravity, and *SIZE* is a decimal number (size in points) or optionally followed by the unit modifier "px" for absolute size. Any one of the options may be absent.

[**-R**|**--font-render-mode** {**normal**,**light**,**mono**}]

There are 3 font render modes:

normal: Full Hinting and Anti-aliasing (default)

light: Slight Hinting and Anti-aliasing

mono: Full Hinting and NO Anti-aliasing

[**-B**|**--font-smoothing-threshold** *size*]

(this gets ignored in 1.3 for now!)

This specifies the largest font size which will be rendered bitmapped, that is, without any font smoothing. By default, no text is rendered bitmapped.

[**-P**|**--pango-markup**]

All text in RRDtool is rendered using Pango. With the **--pango-markup** option, all text will be processed by pango markup. This allows to embed some simple html like markup tags using

```
<span key="value">text</span>
```

Apart from the verbose syntax, there are also the following short tags available.

```
b     Bold
big   Makes font relatively larger, equivalent to <span size="larger">
i     Italic
s     Strikethrough
sub   Subscript
sup   Superscript
small Makes font relatively smaller, equivalent to <span size="smaller">
tt    Monospace font
u     Underline
```

More details on <http://developer.gnome.org/pango/stable/PangoMarkupFormat.html>.

[**−G**|**−−graph−render−mode** {**normal**,**mono**}]

There are 2 render modes:

normal: Graphs are fully Anti-aliased (default)

mono: No Anti-aliasing

[**−E**|**−−slope−mode**]

RRDtool graphs are composed of stair case curves by default. This is in line with the way RRDtool calculates its data. Some people favor a more 'organic' look for their graphs even though it is not all that true.

[**−a**|**−−imgformat PNG**|**SVG**|**EPS**|**PDF**|**XML**|**XMLENUM**|**JSON**|**JSONTIME**|**CSV**|**TSV**|**SSV**]

Image format for the generated graph. For the vector formats you can choose among the standard Postscript fonts Courier-Bold, Courier-BoldOblique, Courier-Oblique, Courier, Helvetica-Bold, Helvetica-BoldOblique, Helvetica-Oblique, Helvetica, Symbol, Times-Bold, Times-BoldItalic, Times-Italic, Times-Roman, and ZapfDingbats.

For Export type you can define XML, XMLENUM (enummerates the value tags <v0>,<v1>,<v2>,...), JSON, JSONTIME (adds a timestamp to each data row), CSV (=comma separated values), TSV (=tab separated values), SSV (=semicolon separated values), (for comma/tab/semicolon separated values the time format by default is in the form of unix time. to change it to something else use: −−x−grid MINUTE:10:HOUR:1:HOUR:4:0:''%Y−%m−%d %H:%M:%S'')

[**−i**|**−−interlaced**]

(this gets ignored in 1.3 for now!)

If images are interlaced they become visible on browsers more quickly.

[**−T**|**−−tabwidth** *value*]

By default the tab-width is 40 pixels, use this option to change it.

[**−b**|**−−base** *value*]

If you are graphing memory (and NOT network traffic) this switch should be set to 1024 so that one Kb is 1024 byte. For traffic measurement, 1 kb/s is 1000 b/s.

[**−W**|**−−watermark** *string*]

Adds the given string as a watermark, horizontally centered, at the bottom of the graph.

[**−Z**|**−−use−nan−for−all−missing−data**]

If one DS is missing, either because the RRD is not available or because it does not contain the requested DS name, just assume that we got empty values instead of raising a fatal error.

Data and variables

DEF:*vname*=*rrdfile*:*ds-name*:*CF*[**:step=***step*][**:start=***time*][**:end=***time*]

CDEF:*vname*=*RPN expression*

VDEF:*vname*=*RPN expression*

You need at least one **DEF** and one **LINE**, **AREA**, **GPRINT**, **PRINT** statement to generate anything useful.

See rrdgraph_data and rrdgraph_rpn for the exact format.

NOTE: **Graph and print elements**

You need at least one graph element to generate an image and/or at least one print statement to generate a report. See rrdgraph_graph for the exact format.

graphv

Calling RRDtool with the graphv option will return information in the RRDtool info format. On the command line this means that all output will be in key=value format. When used from the Perl and Ruby bindings a hash pointer will be returned from the call.

When the filename '−' is given, the contents of the graph itself will also be returned through this interface (hash key 'image'). On the command line the output will look like this:

```
print[0] = "0.020833"
print[1] = "0.0440833"
graph_left = 51
graph_top = 22
graph_width = 400
graph_height = 100
graph_start = 1232908800
graph_end = 1232914200
image_width = 481
image_height = 154
value_min = 0.0000000000e+00
value_max = 4.0000000000e−02
image = BLOB_SIZE:8196
[... 8196 bytes of image data ...]
```

There is more information returned than in the standard interface. Especially the 'graph_*' keys are new. They help applications that want to know what is where on the graph.

ENVIRONMENT VARIABLES

The following environment variables may be used to change the behavior of `rrdtool graph`:

RRDCACHED_ADDRESS

If this environment variable is set it will have the same effect as specifying the `−−daemon` option on the command line. If both are present, the command line argument takes precedence.

RRD_DEFAULT_FONT

RRDtool comes with a preset default font. You can set the environment variable RRD_DEFAULT_FONT if you want to change this.

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 Bogaerdt <alex@vandenbogaerdt.nl> with corrections and/or additions by several people