Creating a Round Robin Database with RRDTool

Introduction

There are two essential components that should be defined when creating a **Round-Robin Database** (RRD) using rrdtool: the **Data Source** (DS) and the **Round-Robin Archive** (RRA). The Data Source definition dictates how the data is entered into the database and can provide rules for the interpretation, limits, and expected frequency of updates. The Round-Robin Archive definition dictates the storage of the data including how the data is consolidated, the time resolution, and the span of time the archive covers. A single RRD instance must contain at least one Data Source and Round-Robin Archive, but may contain more.

Data Source Definition

The general format for a DS specification is:

DS:Label:Type:Heartbeat:Min:Max

Label is the name of the Data Source. It may be from 1-19 characters long and consists of characters in the set [a-zA-Z0-9_]. The label should be descriptive of the data being collected. Example labels include “temperature”, “BytesIn”, “bytes_out”, etc.

Type is one of rrdtool’s defined data types: GAUGE, COUNTER, DERIVE, ABSOLUTE, or COMPUTE.

GAUGE is for storing simple measurements that are rate independent. Data types such as temperatures, prices, or system load are suitable for this data type.

COUNTER is for storing continuously incrementing counters. Data values input into this type of Data Source should never decrease except due to an overflow condition (which is handled by rrdtool). Data of this type is stored as a per-second rate. Example measurements that are suitable for this data type are velocity and throughput.

DERIVE stores the derivative of the line between the current and previous data points. This is useful for converting an input of simple measurements into a rate. For example, with an input of temperature readings, it will record the rate of temperature *change* instead of simply the raw temperature readings.

ABSOLUTE is used for counters which are reset upon reading. This behavior is often found in fast-moving counters that would otherwise overflow frequently. It is otherwise similar to the COUNTER type.

COMPUTE provides a means of applying a formula to *other* Data Sources in the RRD. As such, the DS specification for a COMPUTE type is differs than the general form:

DS:Label:COMPUTE:rpn-expression

The COMPUTE type can be thought of as a “virtual” Data Source that calculates its value based off of the values of other Data Sources.

Heartbeat defines the maximum number of seconds between two data updates before the value of the data is assumed to be unknown. This determines how rrdtool will interpolate data between readings.

Min and *Max* establish the expected range values for the data. Setting appropriate values provides additional safeguards against the accidental inclusion of invalid data. If the expected range values are unknown a value of “U” can be set instead.

Round Robin Archive Definition

The general format for a RRA specification is:

RRA:Consolidation Function:XFF:Steps:Rows

Consolidation Function (CF) is one of rrdtool’s defined functions: AVERAGE, MIN, MAX, and LAST. The different functions provide flexibility when aggregating Primary Data Points (PDP) into a RRA. Note that regardless of the CF chosen, aggregation will result in the loss of resolution for the data. For many applications, this is perfectly acceptable as maintaining fine-grained resolution on older data is often not necessary.

AVERAGE stores the average of all the Primary Data Points within the designated time range.

MIN stores the smallest value of all the Primary Data Points within the designated time range.

MAX stores the largest value of all the Primary Data Points within the designated time range.

LAST stores only the last (most recent) value of all the Primary Data Points within the designated time range.

XFF is the “X Files Factor”. This defines how many Primary Data Points may be unknown before the Consolidated Data Point is also defined as unknown. It is expressed as the ratio of the number of unknown PDP’s over the total number PDP’s for the aggregation period. As such, valid values are between 0 and 1. As an example, a XFF of 0.5 dictates that a CDP will be stored as unknown if the number of unknown PDP’s within the time period exceeds 50% of the total number of PDP’s.

Steps defines how many Primary Data Points will be aggregated into a Consolidated Data Point. The Consolidated Data Point is then recorded in the RRA. Each RRA Step is a multiple of the Step value associated with the Primary Data Point. For example, if the Primary Data Point Step has a resolution of 300 seconds and the RRA Step value is 12, then each Consolidated Data Point will be aggregated over 3600 seconds (12 x 300 seconds).

Rows defines how many Consolidated Data Point Steps are stored in the RRA. As each Row represents the time span defined by the RRA Step, it is a simple calculation to determine the maximum time represented in the archive. For example, if the Primary Data Point has a Step value of 300 seconds, and the RRA Step value is 12, and the Row value is 8760, then the RRA will be able to store 1 year’s worth of data. (300s x 12 x 8760 == 1 year)

rrdtool create

Before attempting to create a RRD, it is best to have fully defined the relevant Data Sources and Round Robin Archives the RRD will be storing. With those defined, there are only a couple other command line options to define in order to successfully create the RRD.

The general format for creating a RRD is as follows:

```
rrdtool create filename --start time --step secs \
DS:Label:Type:Heartbeat:Min:Max \
RRA:Consolidation Function:XFF:Steps:Rows
```

filename is any legitimate filename permitted by the filesystem. It is suggested that a suffix of ".rrd" be used to identify the file as a Round Robin Database.

start is the earliest time represented in the RRD. The RRD will not accept any times that predate (or equal) this value. The time is frequently presented as a standard Unix epoch value (number of seconds since Jan 1, 1970 UTC) although it it will also parse other formats including "now", "now – 1 hour", and more. (See the manual page for the at command for more details on the time format rrdtool supports.) The default value is the current time minus 10 seconds.

step is the number of seconds each Primary Data Point represents. The default value (300 seconds) provides a resolution of 1 PDP every 5 minutes.

Examples

Example 1

```
rrdtool create sysinfo.rrd --start now --step 300 \
DS:load:GAUGE:600:0:U \
RRA:AVERAGE:0.5:1:8928
```

This example creates a RRD with the filename of sysinfo.rrd. The starting time is "now" which rrdtool will determine at the time the command is executed. The step value for the Primary Data Points is 300 seconds.

The only Data Source defined in this RRD is labelled "load" and is of type GAUGE. It has a heartbeat value of 600 seconds, a minimum value of 0 and an unlimited ("U") maximum value.

There is only one Round Robin Archive defined, and it uses the AVERAGE method to aggregate Primary Data Points into Consolidated Data Points. It has an XFF of 0.5, which means that no more than 50% of the PDP's can be of the "unknown" type or the CDP will also be of "unknown" type. The RRA step value is 1, which in this case means there is a 1:1 correspondence between a PDP and a CDP. (i.e. no aggregation is actually performed) There are 8928 data rows in the RRA, so this archive represents a span of 31 days (8928 x 300 seconds).

Example 2

```
rrdtool create sysinfo.rrd --start now --step 300 \
    DS:load:GAUGE:600:0:U \
    RRA:AVERAGE:0.5:1:8928 \
    RRA:AVERAGE:0.5:12:8760
```

This example is identical to Example 1 with the exception that it contains more than one RRA.

In this case, the first RRA defines a “high-resolution” view of the data but only for the last 31 days. In many applications, the value of the data is highest for the most recent data, and so maintaining it with a high fidelity is warranted.

The second RRA defines a “low-resolution” view of the data for a substantially longer period of time. In this example, the RRA step value is 12, so the AVERAGE Consolidation Function will be applied to 12 Primary Data Points to generate a single Consolidated Data Point. As each PDP represents 300 seconds, each step in this RRA will represent 3600 seconds, or 1 hour (300s x 12). With 8760 rows for this RRA, this archive represents a total time span of 1 year (365 days x 24 hours = 8760).

When generating graphs, rrdtool is smart enough to use the appropriate RRA(s) to provide the appropriate resolution for the time span requested.

Example 3

```
    rrdtool create sysinfo.rrd --start now --step 300 \
    DS:load:GAUGE:600:0:U\
    DS:diskfree:DERIVE:600:U:U \
    RRA:AVERAGE:0.5:1:8928 \
    RRA:AVERAGE:0.5:12:8760
```

This example further extends the setup illustrated in Example 2. In this case, an additional Data Source has been defined to track the *rate of change* in the free disk space. As the amount of free disk space may increase or shrink, the DERIVE data type is required. Like the GAUGE Data Source, it has a heartbeat value of 600 seconds but both the minimum and maximum values are unknown for this DS. Note that the RRA’s apply to *both* of the Data Sources.

RRDTool Example Data

Introduction

Many of the postings about using rrdtool reference this example setup. This example is representative of the type of data frequently gathered as part of routine system monitoring. It has been designed to facilitate the illustration of the techniques and problems referenced in the other postings and is not necessarily representative of actual system monitoring conditions. While a more typical usage would probably separate out groups of the statistics into several files, this example collects them all in a single file for the sake of simplicity.

The rrdtool must be installed prior to any attempt to create or load data. It is readily available as a package for installation for a number of Unix distributions or may be downloaded directly from the developer's website and built locally.

Creating the RRD

The example RRD creates a file named sysinfo.rrd, a start date of Jan 1, 2009 00:00:00 PST (Unix epoch time 1230796800)and a Primary Data Point (PDP) step value of 300 seconds. There are a number of different measurements that it retains: the system load, temperature, CPU fan speed, amount of used disk space, rate of change in the use of disk space, and the number of bytes written or read from the disk. For these measurements, it maintains five Round Robin Archives (RRA): three RRAs with a resolution of 5 minutes spanning 31 days using the AVERAGE, MIN, and MAX consolidation functions, a RRA with a resolution of 15 minutes for 90 days, and a RRA with a resolution of 1 hour for 365 days.

```
rrdtool create sysinfo.rrd –start 1230796800 –step 300 \
DS:load:GAUGE:600:0:U \
DS:temperature:GAUGE:600:0:500 \
DS:cpu_fan:GAUGE:600:0:U \
DS:disk_used:GAUGE:600:0:U \
DS:disk_change:DERIVE:600:U:U \
DS:disk2_used:GAUGE:600:0:U \
DS:disk2_change:DERIVE:600:0:U \
DS:bytes_written:COUNTER:600:0:U \
DS:bytes_read:COUNTER:600:0:U \
RRA:AVERAGE:0.5:1:8928 \
RRA:MIN:0.5:1:8928 \
RRA:MAX:0.5:1:8928 \
RRA:AVERAGE:0.5:3:8640 \
RRA:AVERAGE:0.5:12:8760
```

Adding Data

The general format for adding data to the RRD is through the update command. The general format for adding a set of datapoints for the same time value is as follows:

rrdtool update filename timestamp:DS1[:DS2][:DS3]…

filename is the name of the RRD to be updated. An update command can update only a single RRD file at a time.

timestamp is the time of the event. This is typically specified in the standard Unix epoch format (number of seconds elapsed since Jan 1, 1970 UTC), however it may also be represented as "N" (for Now) which will insert the data using the current time value. An alternate format may also be specified using the formats as accepted by the at command. (See the manual page for the at command for more details on the time format rrdtool supports.) If the at format is used, the "@" symbol should be used to separate the timestamp from the data values instead of the ":" character.

DS is the value of the data point(s) that are to be inserted into the RRD for the time specified. The data points are processed in the same order as the Data Sources were specified when the RRD was created. If there is no data for a particular DS, then a "U" (for Unknown) should be used instead. If there is more than one DS to be updated in the RRD, the values should be separated by a ":" character.

An alternate update format is also available which may assign a unique time value for each data point:

rrdtool update filename timestamp:DS1 [timestamp:DS2] [timestamp:DS3]…

Data Generation

The perl script sysinfo-sample.pl is available which will create the RRD as well as populate it with a full year's worth of data. The data is randomly generated but should bear enough semblance to real data to help illustrate the concepts. Also, because it is randomly generated it will result in different results each time it is run.

Please note that this script will automatically overwrite any previous instance of the sysinfo.rrd file in the working directory.

1. *Create a directory that will hold the example data and associated files.*
   ```
   mkdir rrd_example
   ```
2. *Download the sysinfo-sample.pl file into the newly created directory.*
3. Verify the permissions on the perl script are set as executable.
   ```
   chmod 0755 sysinfo-sample.pl
   ```
4. *Execute the perl script and wait for the command prompt to reappear. (This may take 5-15 minutes depending on the speed of your system.)*
   ```
   ./sysinfo-sample.pl
   ```

Upon completion, a file named `sysinfo.rrd` should be created in the working directory. It will be filled with one year's worth of data starting on Jan 1, 2009.

Simple Line Graphs with RRDTool

Introduction

This post covers some of the basic techniques for generating graphs using RRD tool. The demonstration examples assumes that the data has been created as described in the RRD Example Data posting. It should provide a basic understanding of how to specify data sources and transform them into basic line graphs using some of RRDTool's built-in graphing options.

Data Definitions

The data definition is the source of the data to be graphed. It provides a label that is referenced by the graphing directives. There can be multiple data definitions in a single graph and the data may be manipulated extensively by other operations (CDEFs and VDEFs). In addition, it is possible to specify data from multiple files to be combined in a single graph.

The basic format for a Data Definition (DEF) is as follows:

DEF:Label=RRD File:Data Source:Consolidation Function

Label is the name of the data definition. It may be from 1-19 characters long and consists of characters in the set [a-zA-Z0-9_]. The label should be descriptive of the data being graphed and may be the same label as used in the original Data Source. Example labels include "temperature", "BytesIn", "bytes_out", etc.

RRD File is the path and name of the data file. If the full path is not specified, then rrdtool assumes the current working directory.

Data Source is the name of the data source (DS) within the RRD file. See the posting on Creating a RRD File for a reference of the DS definition if necessary.

Consolidation Function specifies how rrdtool may consolidate data prior to display. This does *not* affect the actual data stored in the RRD file; it is used to aggregate data points if the resolution of the graph is too low to display each data point individually. The available consolidation functions are MIN, MAX, AVERAGE, and LAST.

> MIN uses only the smallest value of the Primary Data Points within the resolution's time span.
>
> MAX uses only the largest value of the Primary Data Points within the resolution's time span.
>
> AVERAGE uses the average of the all Primary Data Points within the resolution's time span.
>
> LAST uses only the last (most recent) Primary Data Point within the resolution's time span.

Graphing Directive for Lines

With the data definitions established, graphing directives can be used to specify how the data will be displayed. Note that the graphing directives should be specified only *after* the data definitions (DEF) have been declared. The general format for a line directive is as follows:

LINE[Width]:Label#Color:Legend

Width is an optional element for line graphs. It defines the width of the line used to draw the graph.

Label refers to the label given to a previously defined data definition (DEF). This defines the source of the data to be graphed.

Color defines the color of the line or area graph. This is expressed in the web-standard RGB hexadecimal triplet and must be separated from the label by a '#'. Example color definitions are: 333399 (blue), 33FF00 (bright green), and CC0000 (red). If the color is not specified, then the line will be "invisible".

Legend is the text associated with the legend entry for the graph. It is an optional element, and if it is omitted the ‘:’ separator should also be omitted.

Display Options

There are a number of options available that define how the graph is rendered. These options can define the size, format, time span, legend, title, etc. of the graph. None of them are explicitly required (default values will be used), but they help provide some basic customization of the graphs.

`--width` specifies the width (in pixels) of the *graphing canvas*. The actual image width may be wider to accommodate any border or label elements.

`--height` specifies the height (in pixels) of the *graphing canvas*. The actual image height may be taller to accommodate any border, title, or legend elements.

`--imgformat` specifies the desired output format of the graph. Available options include: PNG, SVG, EPS, and PDF.

`--start` specifies the start time (in absolute or relative terms) for the data displayed on the graphing canvas. Frequently, the start time is a relative offset of the end time (e.g. “end-24 hours”, “end-1 month”, “end-2 weeks”)

`--end` specifies the end time (in absolute or relative terms) for the data displayed on the graphing canvas. The most frequently used value for the end time is “now”, which results in a graph showing the latest data.

`--title` specifies the text to be displayed above the graphing canvas.

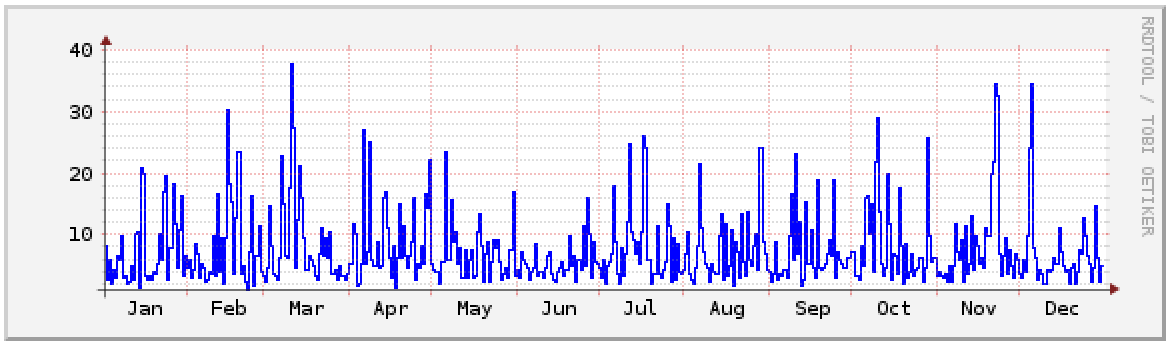
`--vertical-label` specifies the text to be displayed next to the y-axis. Typically, this indicates the units of the measurement.

Examples

Example 1

This basic example creates a graph with the filename “Example 1.png”. As specified by the start and end dates, it encompasses the entire year of 2009. The resulting PNG file will have a data canvas of 500 x 120 pixels, although the full size of the image will be larger to accommodate the legend and border. This example uses only one data definition (DEF) which references the sysinfo.rrd file’s “load” data element (but note that the label for the DEF itself is called “sysload”). The line directive (LINE1) in turn references the “sysload” label to draw a 1 pixel wide blue line.

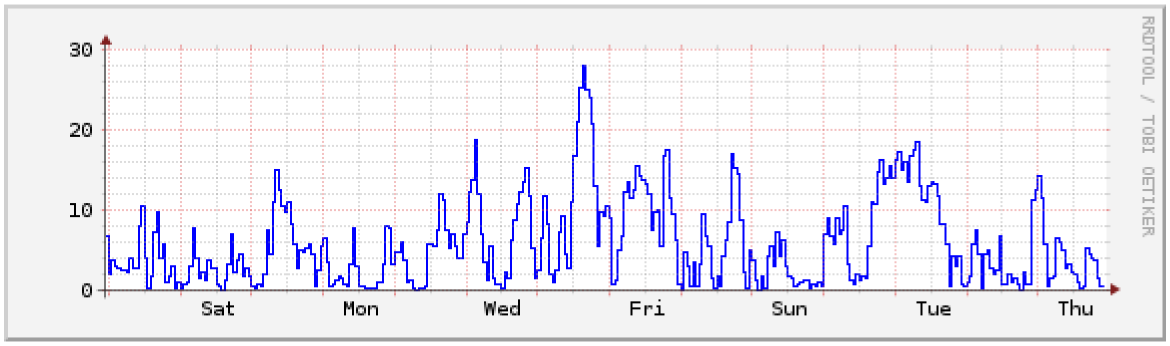
```
rrdtool graph "Example 1.png" \
--start="Jan 1, 2009" --end "Dec 31, 2009" \
--imgformat PNG --width 500 --height 120 \
DEF:sysload=sysinfo.rrd:load:AVERAGE \
LINE1:sysload#0000FF
```



Example 2

This example is similar to the first with the exception that it illustrates the use of relative dates to define the graphing period. In this case, the start time is defined as 2 weeks less than the end time. A frequent use of relative dates is specifying the end time as “now” and the start time relative to that (such as “end-24 hours” or “end-1 week”). Note also RRDTool adjusts the x-axis legend to reflect the different time span.

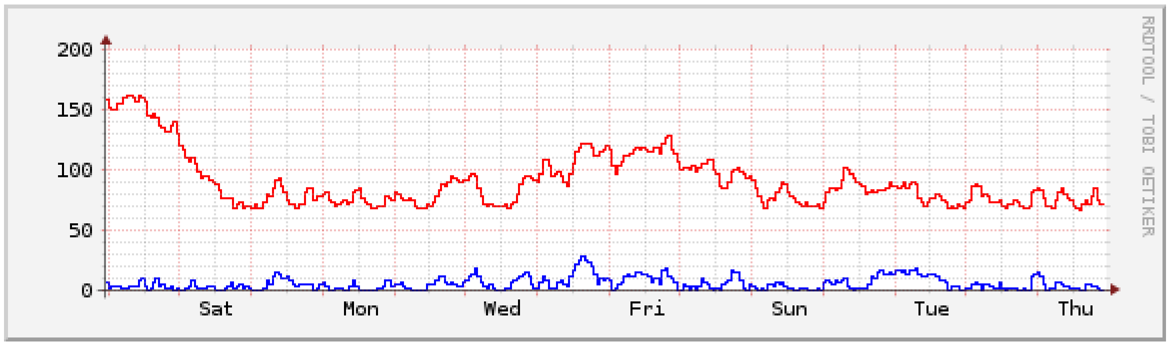
```
rrdtool graph "Example 2.png" \
--start="end-2 weeks" --end "Dec 31, 2009" \
--imgformat PNG --width 500 --height 120 \
DEF:load=sysinfo.rrd:load:AVERAGE \
LINE1:load#0000FF
```



Example 3

This example extends the previous example by adding an additional data definition and line. Note that the lines are drawn *in the order they are specified*, which means that where lines overlap, the line specified *latest* in the graph command will be on top.

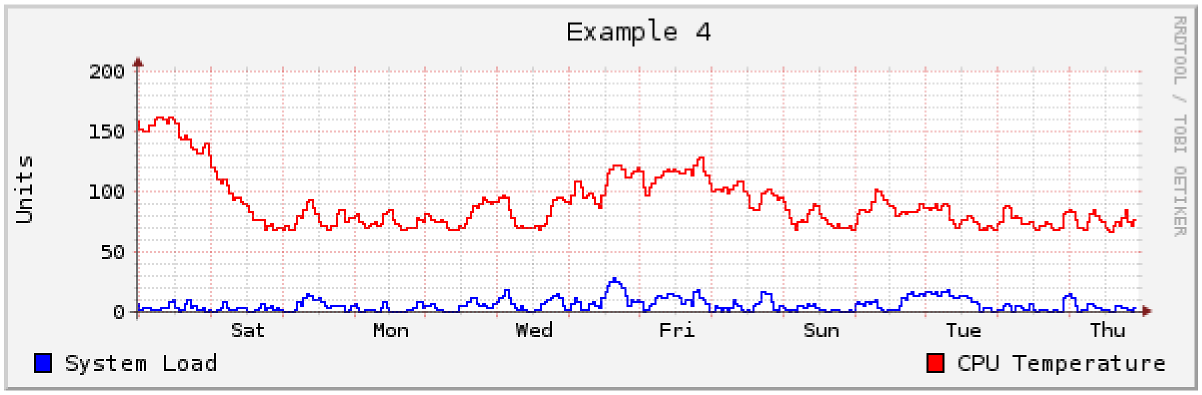
```
rrdtool graph "Example 3.png" \
--start="end-2 weeks" --end "Dec 31, 2009" \
--imgformat PNG --width 500 --height 120 \
DEF:load=sysinfo.rrd:load:AVERAGE \
DEF:temp=sysinfo.rrd:temperature:AVERAGE \
LINE1:load#0000FF \
LINE1:temp#FF0000
```



Example 4

Extending the example further, this iteration introduces some basic legend and title information. This example creates a title for the graph ("Example 4") and a label for the vertical axis ("Units"). In addition, each of the line directives are amended to include a legend specification. RRDTool will create the appropriate "color squares" for the legend entries.

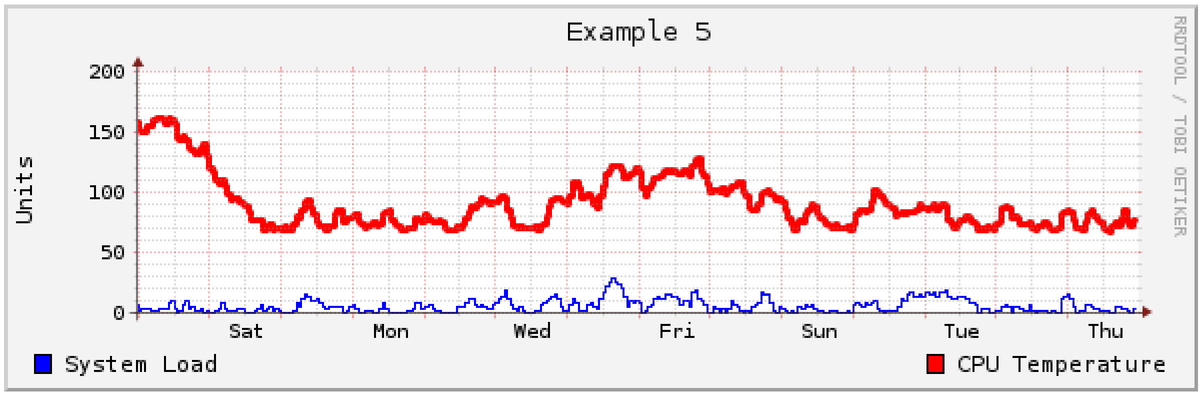
```
rrdtool graph "Example 4.png" \
--start="end-2 weeks" --end "Dec 31, 2009" \
--imgformat PNG --width 500 --height 120 \
--title "Example 4" \
--vertical-label "Units" \
DEF:load=sysinfo.rrd:load:AVERAGE \
DEF:temp=sysinfo.rrd:temperature:AVERAGE \
LINE1:load#0000FF:"System Load" \
LINE1:temp#FF0000:"CPU Temperature"
```



Example 5

This example simply illustrates a variation of the width of the line element. In this case, the line representing the temperature has been specified as being 3 pixels wide (LINE3). Line width can be specified as 1, 2 or 3 pixels wide.

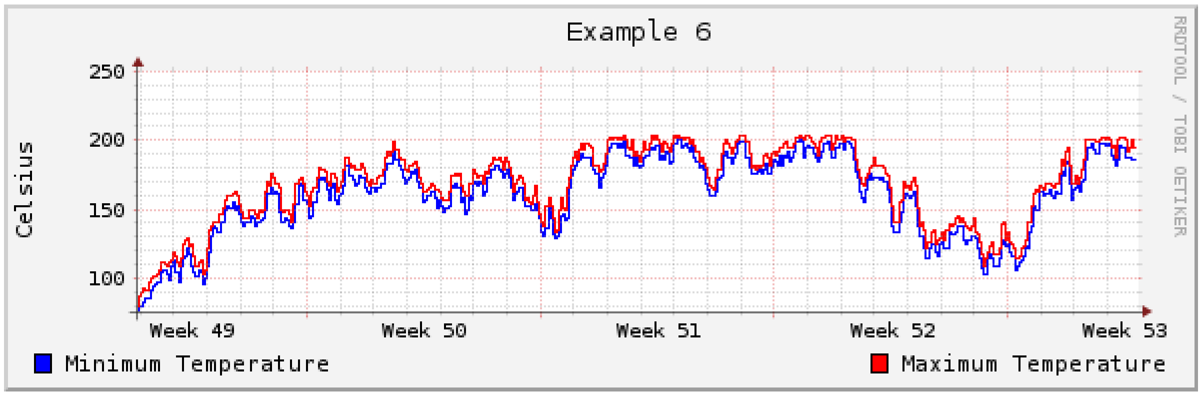
```
rrdtool graph "Example 5.png" \
--start="end-2 weeks" --end "Dec 31, 2009" \
--imgformat PNG --width 500 --height 120 \
--title "Example 5" \
--vertical-label "Units" \
DEF:load=sysinfo.rrd:load:AVERAGE \
DEF:temp=sysinfo.rrd:temperature:AVERAGE \
LINE1:load#0000FF:"System Load" \
LINE3:temp#FF0000:"CPU Temperature"
```



Example 6

This final example illustrates the use of different round-robin archives (RRAs) within a graph definition. In this case, the different RRAs use a different consolidation function (CF) in order to record different aspects of the data patterns. Specifying the MIN and MAX consolidation functions in the DEFs allow us to retrieve and plot the min and max temperature readings.

```
rrdtool graph "Example 6.png" \
--start="end-30 days" --end "Dec 31, 2009" \
--imgformat PNG --width 500 --height 120 \
--title "Example 6" \
--vertical-label "Celsius" \
DEF:mintemp=sysinfo.rrd:temperature:MIN \
DEF:maxtemp=sysinfo.rrd:temperature:MAX \
LINE1:mintemp#0000FF:"Minimum Temperature" \
LINE1:maxtemp#FF0000:"Maximum Temperature"
```



Basic Area Graphs with RRDTool

Introduction

This post describes some basic techniques for generating area charts using RRD tool. It builds upon the previous Line Graphs posting which describes DEFs and other basic graphing options. The data used for the examples was generated using the tool specified in the RRD Example Data posting.

Area Chart Graphing Directives

As specified in the Line Graphs posting, the data definitions (DEFs) must be declared prior to the display directives. The general format for an area directive is as follows:

AREA:Label#Color:Legend:STACK

Label refers to the label given to a previously defined data definition (DEF). This defines the source of the data to be graphed.

Color defines the color of the line or area graph. This is expressed in the web-standard RGB hexadecimal triplet and must be separated from the label by a '#'. Example color definitions are: 333399 (blue), 33FF00 (bright green), and CC0000 (red). If the color is not specified, then the area will be "invisible".

Legend is the text associated with the legend entry for the graph. It is an optional element, and if it is omitted the ':' separator should also be omitted.

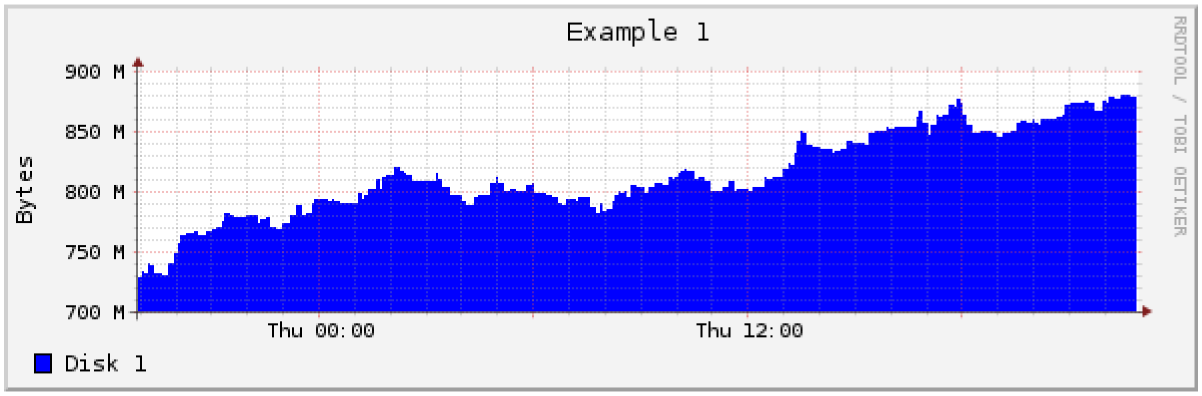
STACK specifies that the results in the graph element be offset from the top of the *previous* display output instead of the normal 0-based offset. It is case-sensitive and should always be specified in all-caps. It is an optional element, and if it is omitted the ':' separator should also be omitted. To specify a stacked element without a legend, simply omit any entry for the legend *but remember the colon*. (e.g. AREA:label#00FFAA::STACK)

Examples

Example 1

This is a simple example of an area graph. In this case, there is only one data element defined and displayed ("disk1").

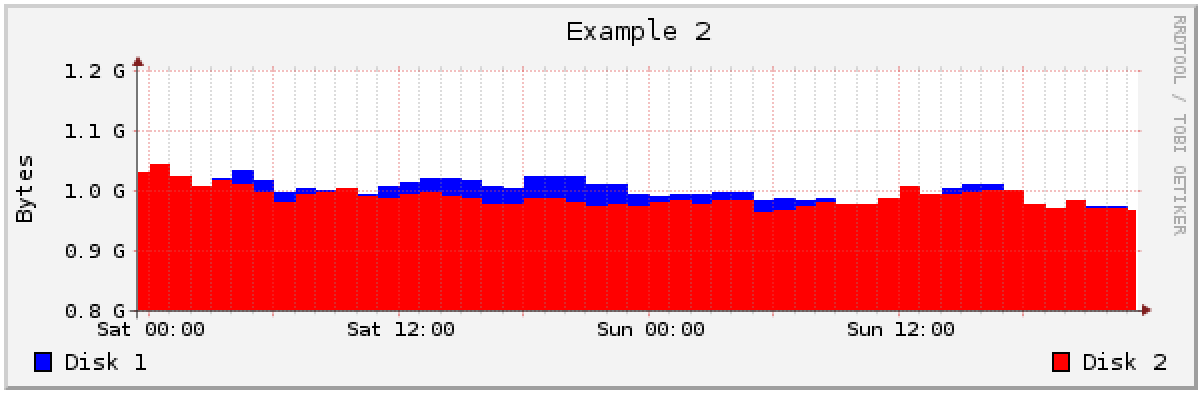
```
rrdtool graph "Example 1.png" \
--start "end-28 hours" --end "Dec 31, 2009" \
--imgformat PNG --width 500 --height 120 \
--title "Example 1" \
--vertical-label "Bytes" \
DEF:disk1=sysinfo.rrd:disk_used:AVERAGE \
AREA:disk1#0000FF:"Disk 1"
```



Example 2

This example illustrates more than one display element. In this case, the elements are *not* stacked and so both use the default 0-based offset. Note that the order of display directives also defines the drawing order, which may result in some (or all) of an area graph in being hidden.

```
rrdtool graph "Example 2.png" \
--start "end-48 hours" --end "Jan 4, 2009" \
--imgformat PNG --width 500 --height 120 \
--title "Example 2" \
--vertical-label "Bytes" \
DEF:disk1=sysinfo.rrd:disk_used:AVERAGE \
DEF:disk2=sysinfo.rrd:disk2_used:AVERAGE \
AREA:disk1#0000FF:"Disk 1" \
AREA:disk2#FF0000:"Disk 2"
```

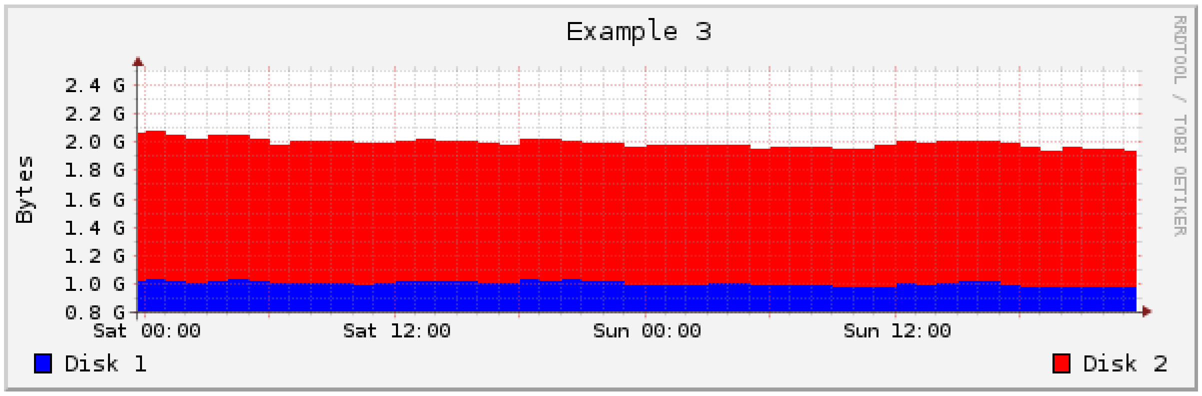


Example 3

This example illustrates the use of the STACK option for the AREA directive. Remember that the STACK offsets the data display to start from the top of the *previously* displayed element. If there is no previous element, then the STACK option uses the default 0-based offset. Any number of display directives can use the STACK option. A frequent use of the STACK option is to illustrate a “summation” of a set of servers or services.

Compare this graph with that of Example 2, which covers the same set of data.

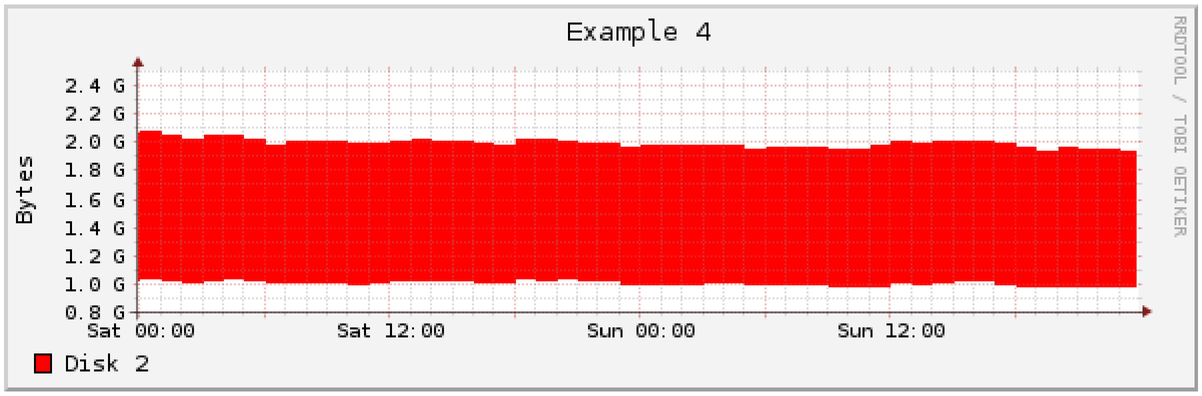
```
rrdtool graph "Example 3.png" \
--start "end-48 hours" --end "Jan 4, 2009" \
--imgformat PNG --width 500 --height 120 \
--title "Example 3" \
--vertical-label "Bytes" \
DEF:disk1=sysinfo.rrd:disk_used:AVERAGE \
DEF:disk2=sysinfo.rrd:disk2_used:AVERAGE \
AREA:disk1#0000FF:"Disk 1":STACK \
AREA:disk2#FF0000:"Disk 2":STACK
```



Example 4

This example illustrates the technique of making an “invisible” element. In this case, the “disk1” directive has no color or legend specified. However, despite it not being displayed, it is still present as applicable to the STACK option for the “disk2” element.

```
rrdtool graph "Example 4.png" \
--start "end-48 hours" --end "Jan 4, 2009" \
--imgformat PNG --width 500 --height 120 \
--title "Example 4" \
--vertical-label "Bytes" \
DEF:disk1=sysinfo.rrd:disk_used:AVERAGE \
DEF:disk2=sysinfo.rrd:disk2_used:AVERAGE \
AREA:disk1::STACK \
AREA:disk2#FF0000:"Disk 2":STACK
```



Rules, Legends and Scales with RRDTool

Introduction

This section covers some of the basic options and directives that can be used to personalize the graph. There are a number of options available to control how the legends and other text are displayed, adjustments for the scaling behavior, as well as means for demarcating significant levels or marking points in time.

Rule Directives

Rules are simply straight lines that are drawn in the graphing area. There are two variations: Horizontal Rules (HRULEs) and Vertical Rules (VRULEs). Typical usages of rules may include displaying a bar across the graph that may indicate an important threshold or a delimiter that reflects a system change. Note that rules will *not* be drawn if they are not within the scope of the actual data ranges being displayed. The general format for rules are as follows:

HRULE:Value#Color:Legend

VRULE:Time#Color:Legend

Value represents the value on the y-axis that will apply to a horizontal rule. Note that horizontal rules can only be perfectly horizontal; it is not possible to supply a formula for a sloped line. If the values of the data being displayed are not within the range of the horizontal line, then the line will not be displayed. A frequent use of a HRULE is to demarcate a critical threshold in the data.

Value represents the value on the y-axis that will apply to a horizontal rule. Note that horizontal rules can only be perfectly horizontal; it is not possible to supply a formula for a sloped line. If the values of the data being displayed are not within the range of the horizontal line, then the line will not be displayed. A frequent use of a HRULE is to demarcate a critical threshold in the data.

Time represents the value on the x-axis that will apply to a vertical rule. The time must be presented as a standard Unix epoch value (number of seconds since Jan 1, 1970 UTC). Note that if the time range of the data displayed does not contain the time specified for the VRULE, the rule will not be displayed. A common use of a VRULE is to flag a “state change” in the measured systems, such as a new software release.

Color defines the color of the line or area graph. This is expressed in the web-standard RGB hexadecimal triplet and must be separated from the label by a ‘#’. Example color definitions are: 333399 (blue), 33FF00 (bright green), and CC0000 (red). If the color is not specified, then the area will be “invisible”.

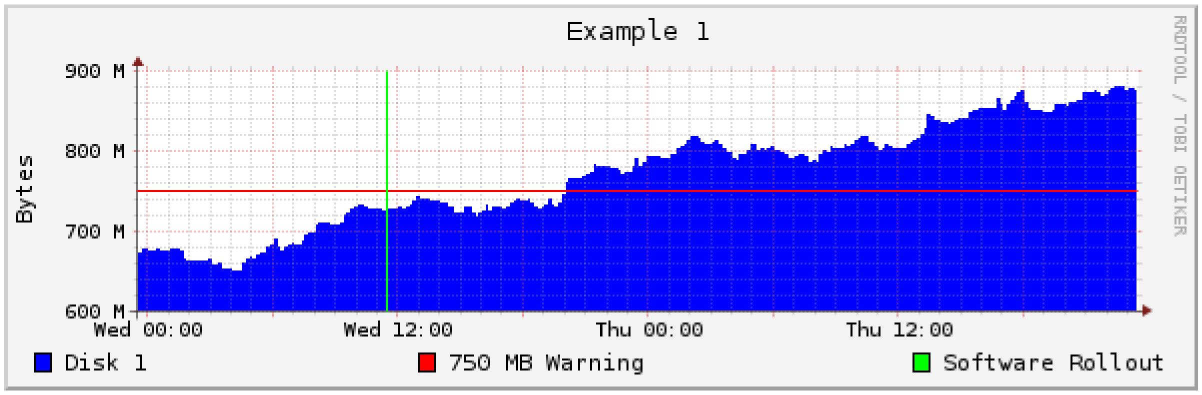
Legend is the text associated with the legend entry for the graph. It is an optional element, and if it is omitted the ‘:’ separator should also be omitted.

Rule Examples

Example 1

The following example is a simple example of the different rule types. The HRULE specifies a red horizontal line drawn at the value of 750,000,000 on the y-axis. The VRULE specifies a light green vertical line drawn at the value of 11:30am Dec 30, 2009 (1262201400 in Unix epoch format).

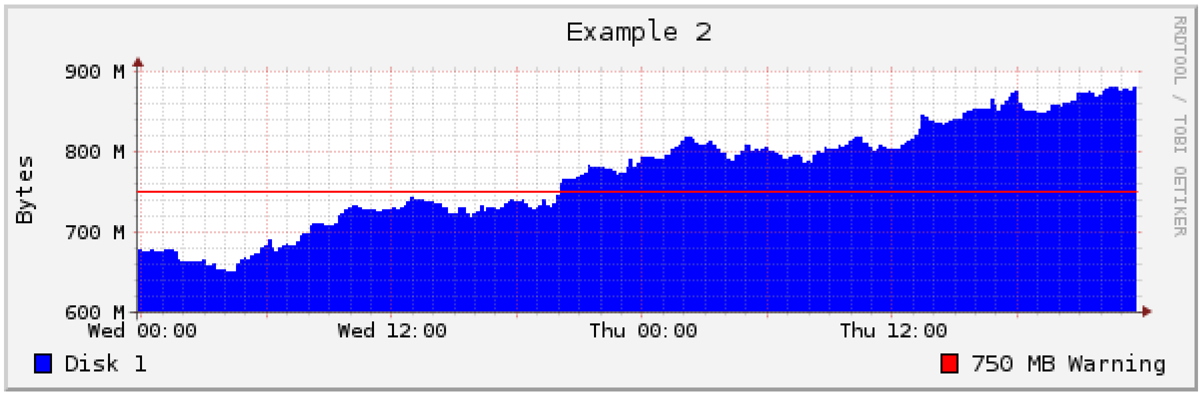
```
rrdtool graph "Example 1.png" \
--start "end-48 hours" --end "Dec 31, 2009" \
--imgformat PNG --width 500 --height 120 \
--title "Example 1" \
--vertical-label "Bytes" \
DEF:disk1=sysinfo.rrd:disk_used:AVERAGE \
AREA:disk1#0000FF:"Disk 1" \
HRULE:750000000#FF0000:"750 MB Warning" \
VRULE:1262201400#00FF00:"Software Rollout"
```



Example 2

The following example illustrates use of a time value for the VRULE (Dec 18, 2009) that is not within the scope of the graphed data. Note that the rule in this case is *not* displayed nor is there a legend element. If the scope of the graphed data is changed to encompass the value of the VRULE, then the rule will again be displayed. This behavior provides some utility in demarcating significant events without causing the y-axis to be “pinned” to a specified time.

```
rrdtool graph "Example 2 Rule.png" \
--start "end-48 hours" --end "Dec 31, 2009" \
--imgformat PNG --width 500 --height 120 \
--title "Example 2" \
--vertical-label "Bytes" \
DEF:disk1=sysinfo.rrd:disk_used:AVERAGE \
AREA:disk1#0000FF:"Disk 1" \
HRULE:750000000#FF0000:"750 MB Warning" \
VRULE:1261201400#00FF00:"Software Rollout"
```



Legend and Title Options

`--title` specifies the text to be displayed above the graphing canvas.

`--vertical-label` specifies the text to be displayed next to the y-axis. Typically, this indicates the units of the measurement.

`--right-axis` specifies an alternate y-axis scale that will be displayed on the right side of the graph. This can be used to realign the units displayed which may have been adjusted to increase the legibility of the graph. This parameter requires both a scale and offset (scale:offset) be specified.

`--right-axis-label` specifies the text to be displayed on the right axis.

`--no-legend` omits the legend information from being drawn.

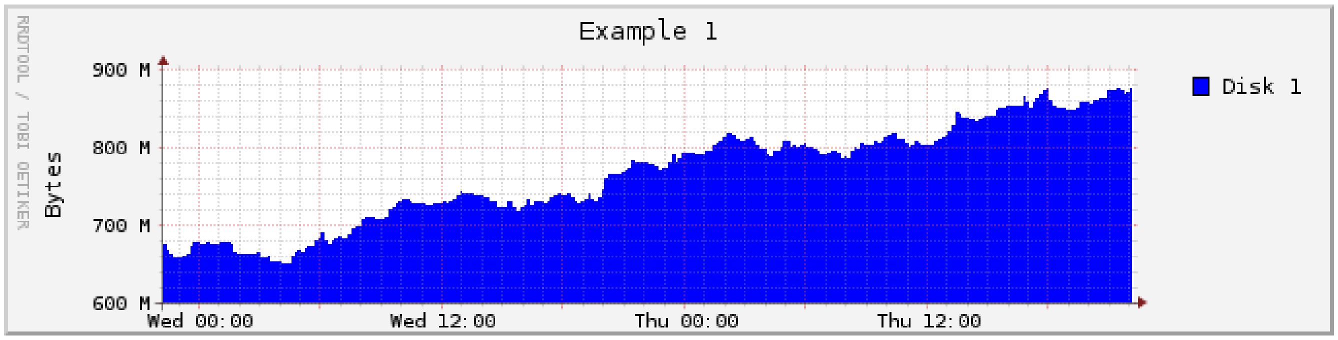
`--legend-position` defines where the legend will be displayed in the graph. Acceptable values are `north`, `south`, `east` and `west`. The default value is `south`.

Legend and Title Examples

Example 1

This simple example illustrates the title, vertical-label, and legend-position parameters in use. Note that the text for the label and title is quoted to ensure proper parsing. The legend-position overrides the default value of south with east.

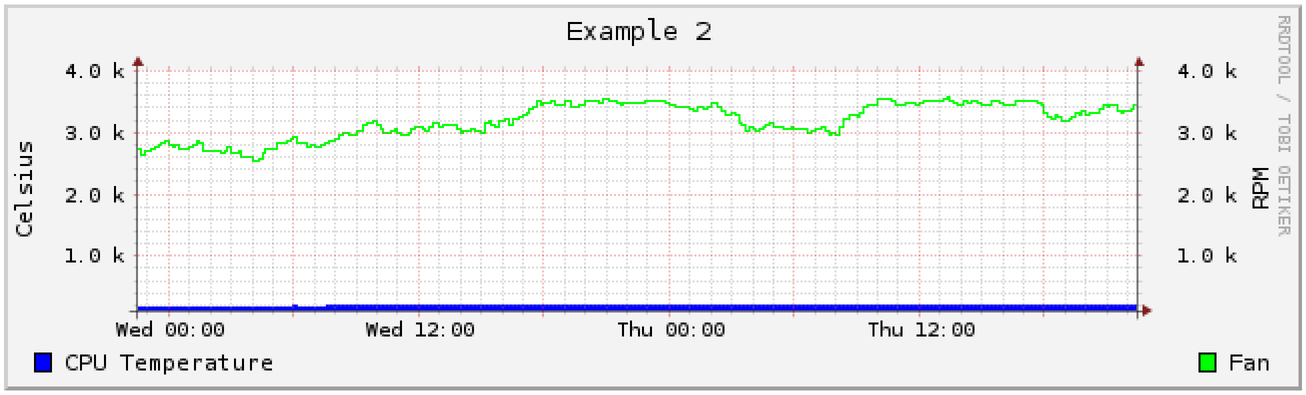
```
rrdtool graph "Example 1 Legend.png" \
--start "end-48 hours" --end "Dec 31, 2009" \
--imgformat PNG --width 500 --height 120 \
--title "Example 1" \
--vertical-label "Bytes" \
--legend-position east \
DEF:disk1=sysinfo.rrd:disk_used:AVERAGE \
AREA:disk1#0000FF:"Disk 1"
```



Example 2

This example illustrates the use of the right-axis to specify an alternate scale. In this case, the y-axis scale on the left (the default) reflects the CPU temperature measured in Celsius. The y-axis scale on the right reflects the fan speed measured in RPM. For this example, the scale has been set to 1 and the offset specified as 0.

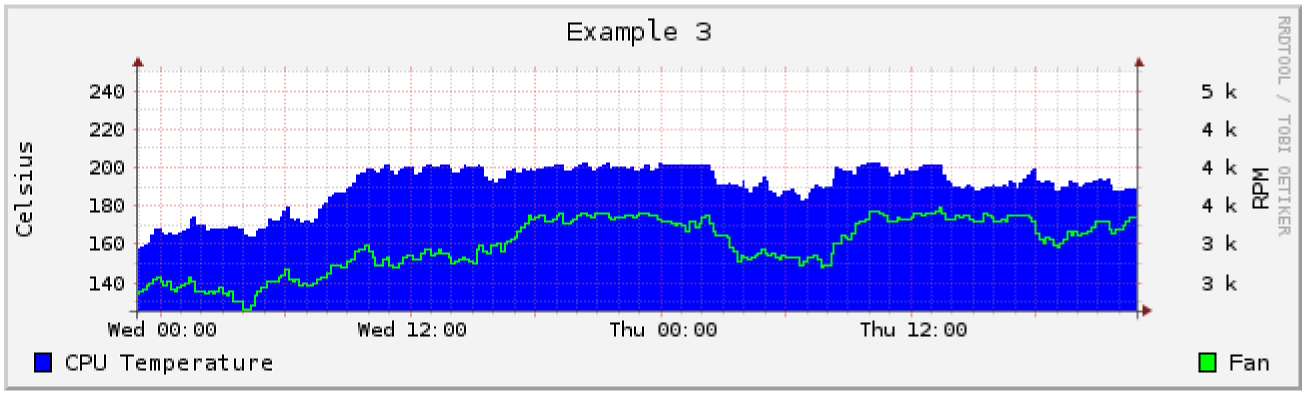
```
rrdtool graph "Example 2 Legend.png" \
--start "end-48 hours" --end "Dec 31, 2009" \
--imgformat PNG --width 500 --height 120 \
--title "Example 2" \
--vertical-label "Celsius" \
--right-axis 1:0 \
--right-axis-label RPM \
DEF:temp=sysinfo.rrd:temperature:AVERAGE \
DEF:fan=sysinfo.rrd:cpu_fan:AVERAGE \
AREA:temp#0000FF:"CPU Temperature" \
LINE1:fan#00FF00:"Fan"
```



Example 3

This example further refines the graph defined in Example 2 by adjusting the values of the fan so that they are closer to the range of the temperatures. (This is performed by the CDEF directive, which will be further described in later examples.) To keep the units correct, the scale for the right-label is adjusted to re-compensate. Note that in this example the variations in temperature and RPM are now both clearly visible and the left and right scales on the y-axis reflect the different scales of the datatypes.

```
rrdtool graph "Example 3 Legend.png" \
--start "end-48 hours" --end "Dec 31, 2009" \
--imgformat PNG --width 500 --height 120 \
--title "Example 3" \
--vertical-label "Celsius" \
--right-axis 20:0 \
--right-axis-label RPM \
DEF:temp=sysinfo.rrd:temperature:AVERAGE \
DEF:fan=sysinfo.rrd:cpu_fan:AVERAGE \
CDEF:fan20=fan,20,/ \
AREA:temp#0000FF:"CPU Temperature" \
LINE1:fan20#00FF00:"Fan"
```



Scale Options

`--upper-limit` provides an override of the default auto-scaling behavior by setting an explicit upper limit value for the y-axis. Note that the value provided by this option will continue to be overridden if an actual data value in the graph exceeds the limit specified.

`--lower-limit` provides an override of the default auto-scaling behavior by setting an explicit lower limit value for the y-axis. Note that the value provided by this option will continue to be overridden if an actual data value in the graph is lower than the limit specified.

`--rigid` is used in conjunction with the upper-limit and lower-limit options to provide a definitive maximum and minimum y-axis value. With this option specified, the auto-scaling behavior will not adjust the scale even if a data value would exceed (or fall below) the upper or lower limits specified.

`--logarithmic` changes the y-axis to use a logarithmic scale instead of the default linear scale. This can be useful to visualize fine-grain patterns in the data that may otherwise be obscured if the values are wide-ranging.

`--units-exponent` sets the exponent expressed in the y-axis scale to a fixed value. For example, setting this to 3 would lead to the scale to consistently use units of 1000 (10^3).

`--units-length` defines how many characters rrdtool should assume are present in the y-axis scale labels. This may be necessary to specify when deviating from the default values for the expression of the units (via the `units-exponent`, `logarithmic`, or `units=si` options) in order to prevent rrdtool from overlapping the vertical label and scale labels.

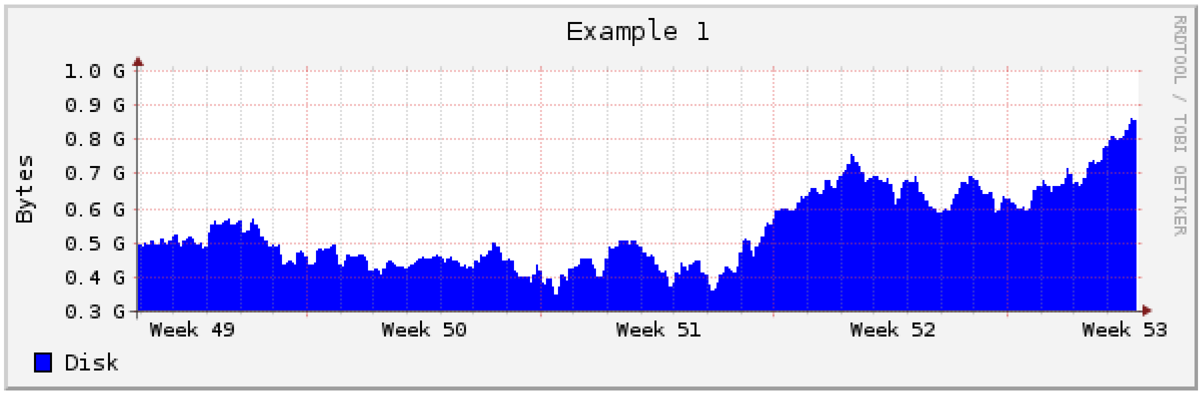
`--units=si` overrides the exponential notation with the standard SI unit symbols (k, M, etc.) Note that the exponential notation is the default only for logarithmic graphs; linear graphs already use the SI notation.

Scale Examples

Example 1

This example illustrates the use of the upper and lower limit options. Note that in this case the specified lower-limit is *ignored*, as an actual data value is lower than the value specified and so the auto-scaler adjusts the lower range of the scale to compensate.

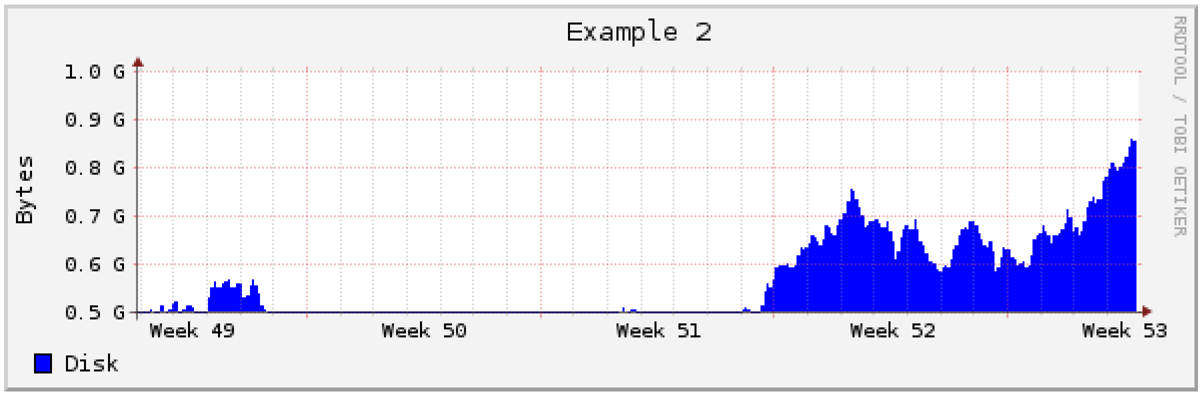
```
rrdtool graph "Example 1 Scale.png" \
--start "end-1 month" --end "Dec 31, 2009" \
--imgformat PNG --width 500 --height 120 \
--title "Example 1" \
--vertical-label "Bytes" \
--upper-limit 1000000000 \
--lower-limit 500000000 \
DEF:disk1=sysinfo.rrd:disk_used:AVERAGE \
AREA:disk1#0000FF:"Disk"
```



Example 2

This example demonstrates how the rigid option can be used to enforce the specified upper and lower limits. Note that this strict adherence to the specified limits may prevent data that is out of range from being displayed.

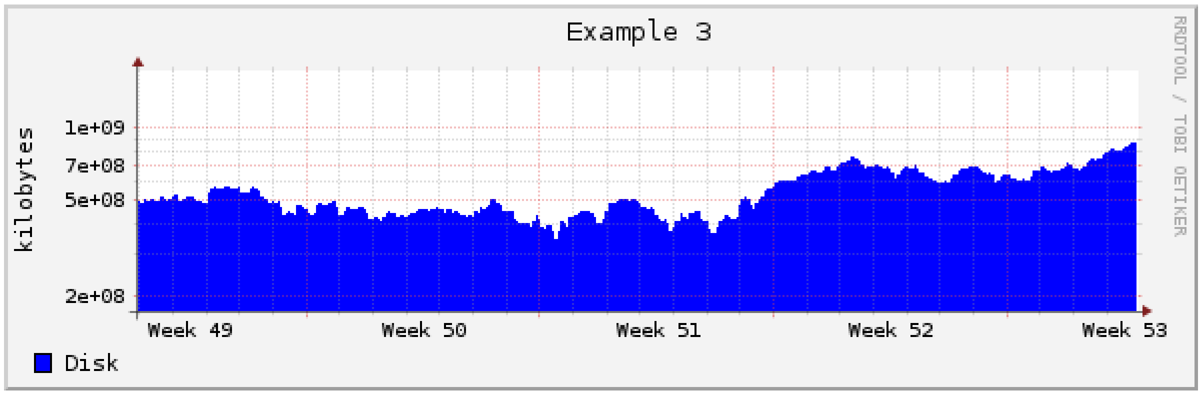
```
rrdtool graph "Example 2 Scale.png" \
--start "end-1 month" --end "Dec 31, 2009" \
--imgformat PNG --width 500 --height 120 \
--title "Example 2" \
--vertical-label "Bytes" \
--upper-limit 1000000000 \
--lower-limit 500000000 \
--rigid \
DEF:disk1=sysinfo.rrd:disk_used:AVERAGE \
AREA:disk1#0000FF:"Disk"
```



Example 3

In this example, the logarithmic option is specified to alter the y-axis scaling behavior. The units-exponent is also specified which maintains the expression of the scale at the fixed rate of 10^3 (1000's).

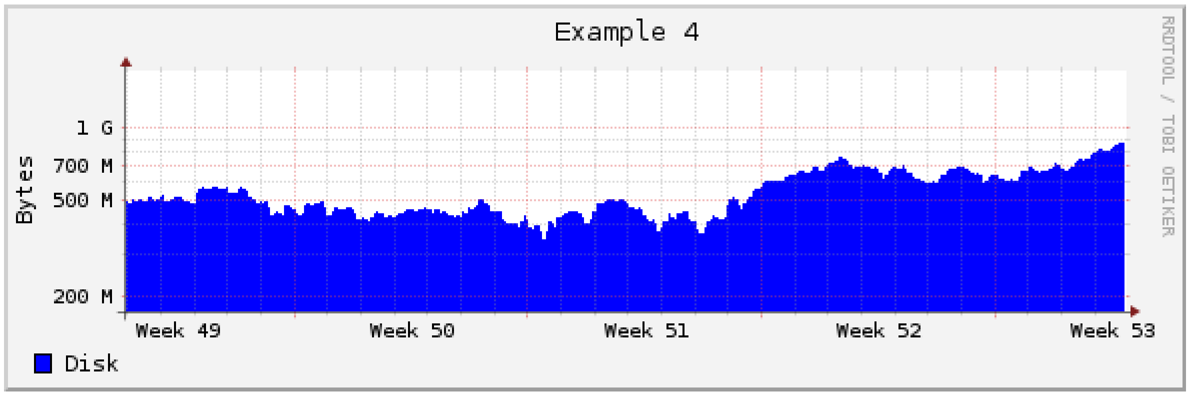
```
rrdtool graph "Example 3 Scale.png" \
--start "end-1 month" --end "Dec 31, 2009" \
--imgformat PNG --width 500 --height 120 \
--title "Example 3" \
--vertical-label "kilobytes" \
--logarithmic \
--units-exponent 3 \
DEF:disk1=sysinfo.rrd:disk_used:AVERAGE \
AREA:disk1#0000FF:"Disk"
```



Example 4

This is an alternate view of the previous graph using the SI notation instead of the default exponential notation for a logarithmic graph. In addition, the units-length option is specified to facilitate the alignment of the axis label and scale units.

```
rrdtool graph "Example 4 Scale.png" \
--start "end-1 month" --end "Dec 31, 2009" \
--imgformat PNG --width 500 --height 120 \
--title "Example 4" \
--vertical-label "Bytes" \
--logarithmic \
--units=si \
--units-length 5 \
DEF:disk1=sysinfo.rrd:disk_used:AVERAGE \
AREA:disk1#0000FF:"Disk"
```



Using CDEFs to manipulate data in RRDTool

Introduction

The CDEF directive provides a means for manipulating the “raw” data stored in a round-robin archive (RRA). It is typically used to apply a mathematical function to each data point referenced by one or more DEF statements which results in an array of new values — each of remains associated with the respective time of the original “raw” data point. This is an in-memory transformation only; the original data remains unchanged in the RRA.

There is often much confusion regarding the differences of the DEF, CDEF, and VDEF directives. It may help to think of the different directives in the following manner:

- A DEF directive references a set of “raw” data as it is stored in a RRA
- A CDEF directive applies a function to *each* data point it references
- A VDEF directive applies a function to an *aggregate* of data points

The CDEF Directive

The basic format for a CDEF directive is as follows:

CDEF:Label=RPN Expression

Label is the name of the CDEF. It may be referenced in other directives for inclusion in LINE, AREA charts or even other CDEF calculations. It may be from 1-19 characters long and consists of characters in the set [a-zA-Z0-9_]. Note that the label must be unique and cannot overlap with any labels assigned to other DEFs, CDEFs, or VDEFs.

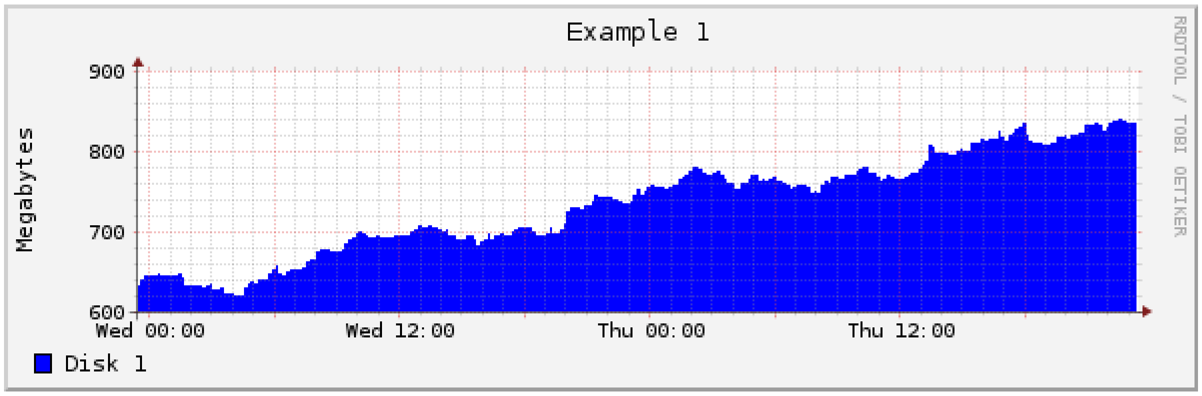
RPN Expression is the mathematical or logical expression that may be used to manipulate the raw data values as referenced by DEF or CDEF directives or even a pure mathematical function. The expression uses Reverse Polish Notation to eliminate confusion or errors that may occur with the precedence rules required of traditional infix notation. There are a number of mathematical, boolean and logical operators available for inclusion in a CDEF directive.

CDEF Examples

Example 1

This example illustrates the commonly used transformation of bytes to megabytes. Disk and memory readings are often reported in bytes, but frequently this is not the most convenient unit for visualization. In this case, the CDEF directive divides the "raw" data value as referenced by the DEF and divides it by 1048576 (1024 x 1024). Note that the AREA directive now references the CDEF label and that the vertical label has been updated to reflect the proper units.

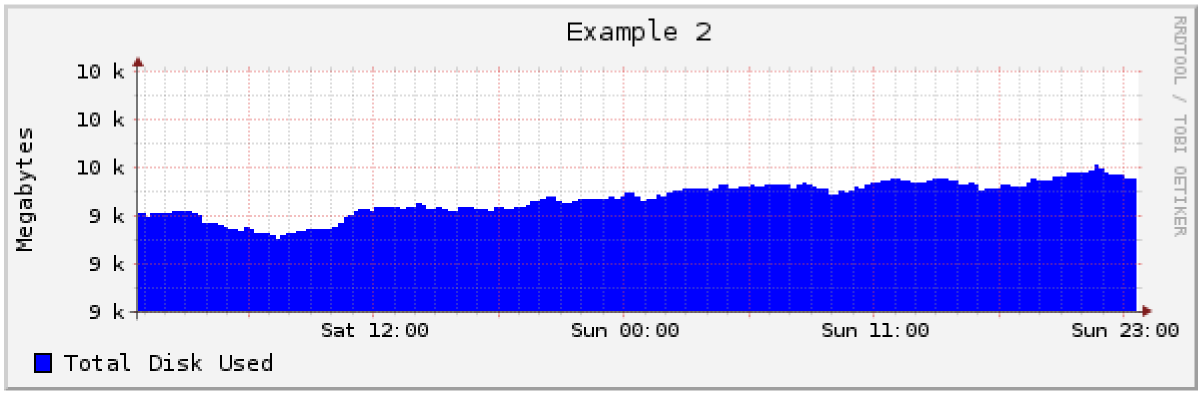
```
rrdtool graph "Example 1 CDEF.png" \
--start "end-48 hours" --end "Dec 31, 2009" \
--imgformat PNG --width 500 --height 120 \
--title "Example 1" \
--vertical-label "Megabytes" \
DEF:disk1=sysinfo.rrd:disk_used:AVERAGE \
CDEF:megadisk1=disk1,1048576,/ \
AREA:megadisk1#0000FF:"Disk 1"
```



Example 2

This example shows a slightly more complex instance of the reverse-polish math that may be referenced in a CDEF. In this case, the CDEF first sums up the the "raw" values as referenced by the two DEF statements and then converts the sum to megabytes.

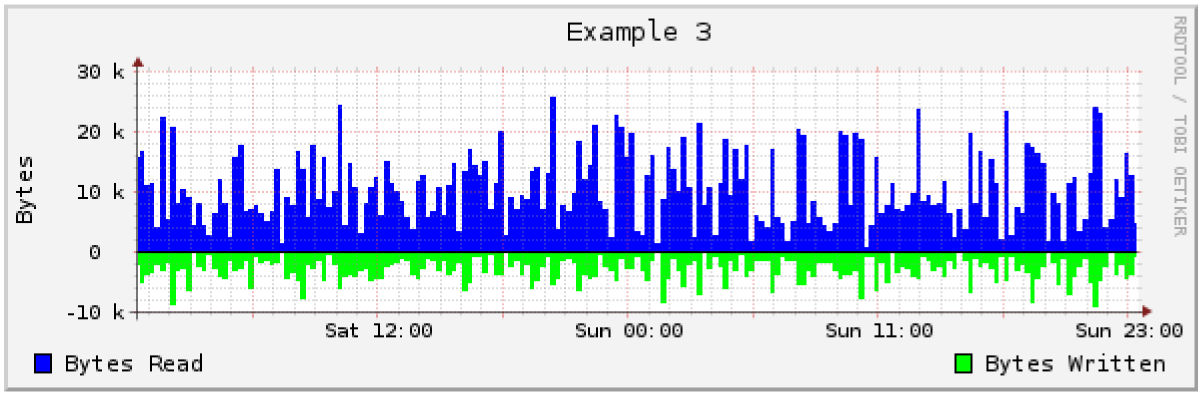
```
rrdtool graph "Example 2 CDEF.png" \
--start "end-48 hours" --end "Nov 1, 2009" \
--imgformat PNG --width 500 --height 120 \
--title "Example 2" \
--vertical-label "Megabytes" \
DEF:disk1=sysinfo.rrd:disk_used:AVERAGE \
DEF:disk2=sysinfo.rrd:disk2_used:AVERAGE \
CDEF:megadisk=disk1,disk2,+,1048576,/ \
AREA:megadisk#0000FF:"Total Disk Used"
```



Example 3

This example illustrates a common technique for differentiating several types of related measurements. It is a frequent graphing style for disk IO (reads vs. writes) as well as network IO (octets in vs. octets out). It is achieved by simply negating the values of one of the operations and graphing the result. In this example, a horizontal rule (HRULE) at the 0 point has also been added in order to highlight the baseline. Note that the HRULE is specified as the last element to be drawn, which ensures that it will overlay the other graphed elements.

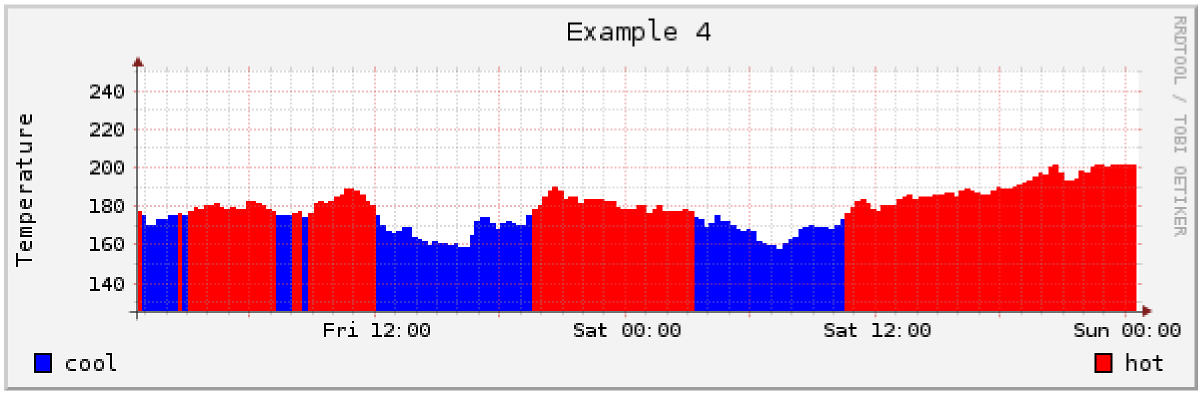
```
rrdtool graph "Example 3 CDEF.png" \
--start "end-48 hours" --end "Nov 1, 2009" \
--imgformat PNG --width 500 --height 120 \
--title "Example 3" \
--vertical-label "Bytes" \
DEF:read=sysinfo.rrd:bytes_read:AVERAGE \
DEF:write=sysinfo.rrd:bytes_written:AVERAGE \
CDEF:negwrite=0,write,- \
AREA:read#0000FF:"Bytes Read" \
AREA:negwrite#00FF00:"Bytes Written" \
HRULE:0#000000
```



Example 4

This is a more complex example which illustrates the use of the IF operation as well as a more advanced graphing style useful to call out anomalous behaviors. In this case, the temperature values (referenced by the DEF “temp”) are first assessed by the LE (less than or equal to) and GT (greater than) operations. The values assigned in these CDEFs (iscool, ishot) will then be used in the CDEFs with the IF operations. The IF operations evaluate iscool/ishot and if it is “true” (i.e. not zero), then the value for temp is returned. Otherwise, the special constant “unknown” is returned. These values for the cool/hot CDEFs are then graphed and result in a clear demarcation where the system is over-heated.

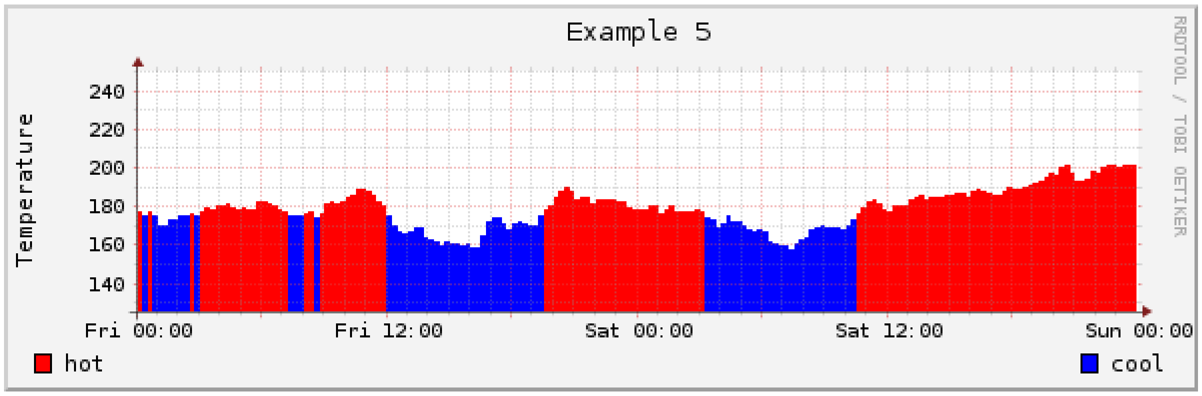
```
rrdtool graph "Example 4 CDEF.png" \
--start "end-48 hours" --end "Nov 1, 2009" \
--imgformat PNG --width 500 --height 120 \
--title "Example 4" \
--vertical-label "Temperature" \
DEF:temp=sysinfo.rrd:temperature:AVERAGE \
CDEF:iscool=temp,175,LE \
CDEF:ishot=temp,175,GT \
CDEF:cool=iscool,temp,UNKN,IF \
CDEF:hot=ishot,temp,UNKN,IF \
AREA:cool#0000FF:"cool" \
AREA:hot#FF0000:"hot"
```



Example 5

This example illustrates the use of the LIMIT operation to achieve a similar effect for highlighting anomalous conditions. In this case, the entire data set is initially graphed using the “hot” color and then the “cool” data set is overlayed on top of the appropriate sections.

```
rrdtool graph "Example 5 CDEF.png" \
--start "end-48 hours" --end "12am Nov 1, 2009" \
--imgformat PNG --width 500 --height 120 \
--title "Example 5" \
--vertical-label "Temperature" \
DEF:temp=sysinfo.rrd:temperature:AVERAGE \
CDEF:cool=temp,0,175,LIMIT \
AREA:temp#FF0000:"hot" \
AREA:cool#0000FF:"cool"
```



Example 6

This example shows the use of the MIN operation to provide a "layer cake effect" in the graph. This is a popular graphing technique that can be used either to signify a state change above a given threshold or simply to provide a color gradient in the graph for extra polish. This particular example again relies on overlaying the "cool" graph to mask out the relevant sections of the data.

```
rrdtool graph "Example 6 CDEF.png" \
--start "end-48 hours" --end "12am Nov 1, 2009" \
--imgformat PNG --width 500 --height 120 \
--title "Example 6" \
--vertical-label "Temperature" \
DEF:temp=sysinfo.rrd:temperature:AVERAGE \
CDEF:cool=temp,175,MIN \
AREA:temp#FF0000:"hot" \
AREA:cool#0000FF:"cool"
```

Using VDEFs for set calculations in RRDTool

Introduction

The VDEF directive provides a mechanism for applying mathematical operations on *sets* of data. Unlike the CDEF directive, the result of a VDEF is a single value. The VDEF can reference either a DEF or CDEF data set and can perform a number of operations including calculating averages, standard deviations, least squares lines, and more. These values can then be further referenced in CDEFs for graphing or printed out.

There is often much confusion regarding the differences of the DEF, CDEF, and VDEF directives. It may help to think of the different directives in the following manner:

- A DEF directive references a set of "raw" data as it is stored in a RRA
- A CDEF directive applies a function to *each* data point it references
- A VDEF directive applies a function to an *aggregate* of data points

The VDEF Directive

The basic format for a VDEF directive is as follows:

VDEF:Label=RPN Expression

Label is the name of the VDEF. It may be referenced in other directives including HRULEs or CDEF calculations. It may be from 1-19 characters long and consists of characters in the set [a-zA-Z0-9_]. Note that the label must be unique and cannot overlap with any labels assigned to other DEFs, CDEFs, or VDEFs.

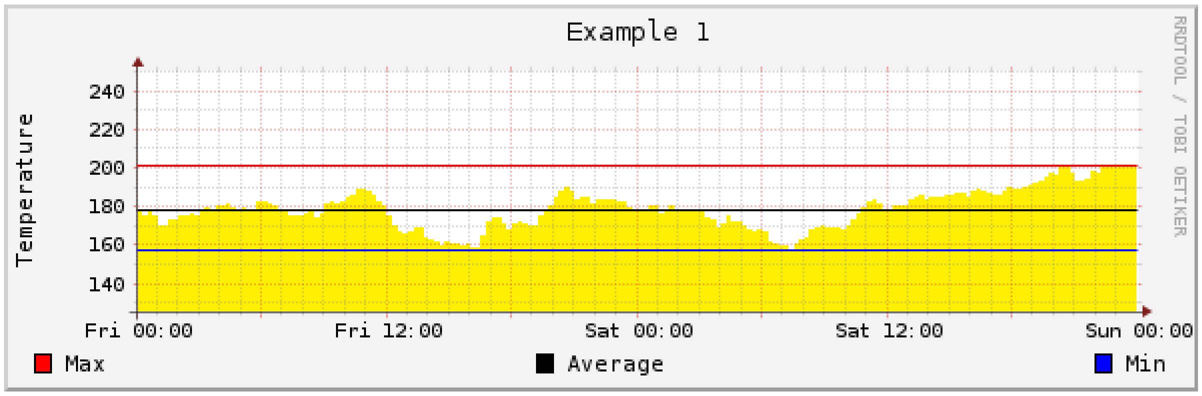
RPN Expression is the mathematical or logical expression that is applied to a data set. The expression uses Reverse Polish Notation to eliminate confusion or errors that may occur with the precedence rules required of traditional infix notation. There are a number of mathematical, boolean and logical operators available for inclusion in a VDEF directive.

VDEF Examples

Example 1

This is a simple example which illustrates the use of VDEFs to reveal information about the set of data. In this case, the MINIMUM, MAXIMUM, and AVERAGE operations are used to determine the respective values for the data set. The resulting value for each operation is then used as the value for a HRULE providing a clear illustration in the resulting graph. Note that the VDEF values are based *only on the data referenced*. In this case, the data is bounded by the start and end time specified and so would likely differ for a different window into the original data set.

```
rrdtool graph "Example 1 VDEF.png" \
--start "end-48 hours" --end "12am Nov 1, 2009" \
--imgformat PNG --width 500 --height 120 \
--title "Example 1" \
--vertical-label "Temperature" \
DEF:temp=sysinfo.rrd:temperature:AVERAGE \
VDEF:min=temp,MINIMUM \
VDEF:max=temp,MAXIMUM \
VDEF:avg=temp,AVERAGE \
AREA:temp#FFEF00 \
HRULE:max#FF0000:"Max" \
HRULE:avg#000000:"Average" \
HRULE:min#0000FF:"Min"
```



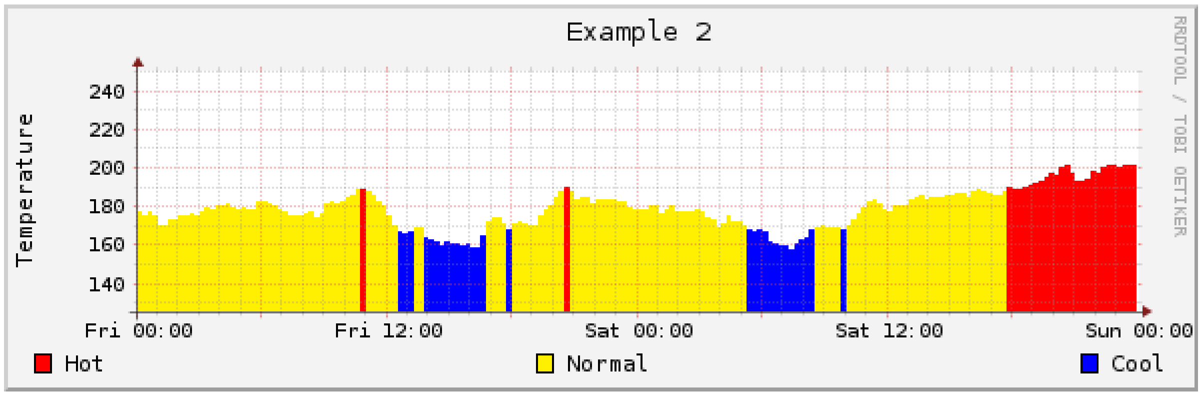
Example 2

This example uses the average and standard deviation VDEF operations to identify temperature ranges that exceed the average temperature by more than one standard deviation. The VDEF operations determine the appropriate values, which are then used in the CDEF operations to generate new data sets. The original data set is graphed in the “hot” color with the “normal” and “cool” graphs overlayed on top of the appropriate sections.

The CDEF calculations may be difficult to parse for a novice to Reverse Polish syntax. It may help to break it down as follows:

1. cool=temp,avg,stdev,-,LE,temp,UNKN,IF
2. cool=temp,(avg – stdev),LE,temp,UNKN,IF
3. cool=(temp <= (avg – stdev)),temp,UNKN,IF
4. cool=if (temp <= (avg – stdev)) then temp else UNKN

```
rrdtool graph "Example 2 VDEF.png" \
--start "end-48 hours" --end "12am Nov 1, 2009" \
--imgformat PNG --width 500 --height 120 \
--title "Example 2" \
--vertical-label "Temperature" \
DEF:temp=sysinfo.rrd:temperature:AVERAGE \
VDEF:avg=temp,AVERAGE \
VDEF:stdev=temp,STDEV \
CDEF:cool=temp,avg,stdev,-,LE,temp,UNKN,IF \
CDEF:medium=temp,avg,stdev,+,LE,temp,UNKN,IF \
AREA:temp#FF0000:"Hot" \
AREA:medium#FFEF00:"Normal" \
AREA:cool#0000FF:"Cool"
```

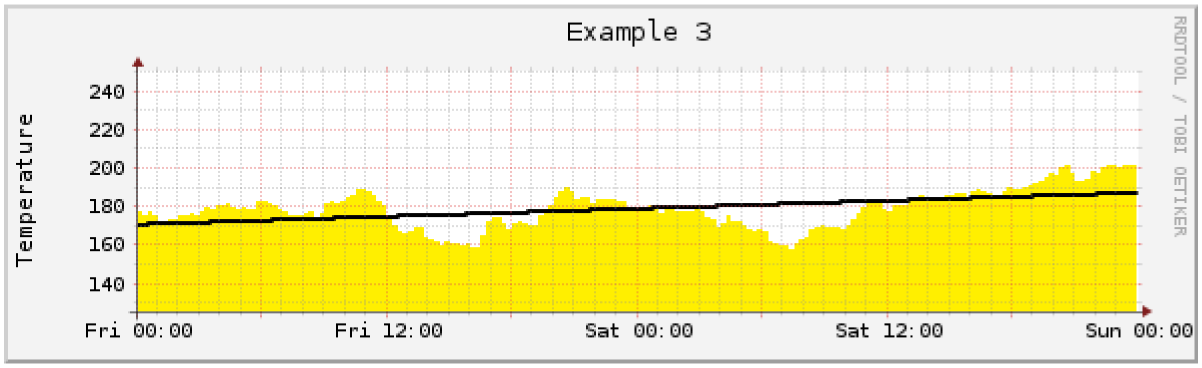


Example 3

This example illustrates how to use the least squares line VDEF operations to draw a trendline. The LSLSLOPE operation can determine the slope of line and the LSLINT can provide the y-axis intercept value. A trendline can then be generated using the classic "y=mx+b" formula (where m is the slope and b is the intercept).

The CDEF operation implements this formula using several “tricks” of rrdtool: a workaround for the CDEF requirement to reference a DEF or CDEF, and the use of the COUNT operation to increment a value for each data point in the graph set. In the example, the CDEF reference requirement is satisfied by the “temp,POP” elements, which effectively puts a value from the temp DEF on the stack and then pops it back off, discarding it.

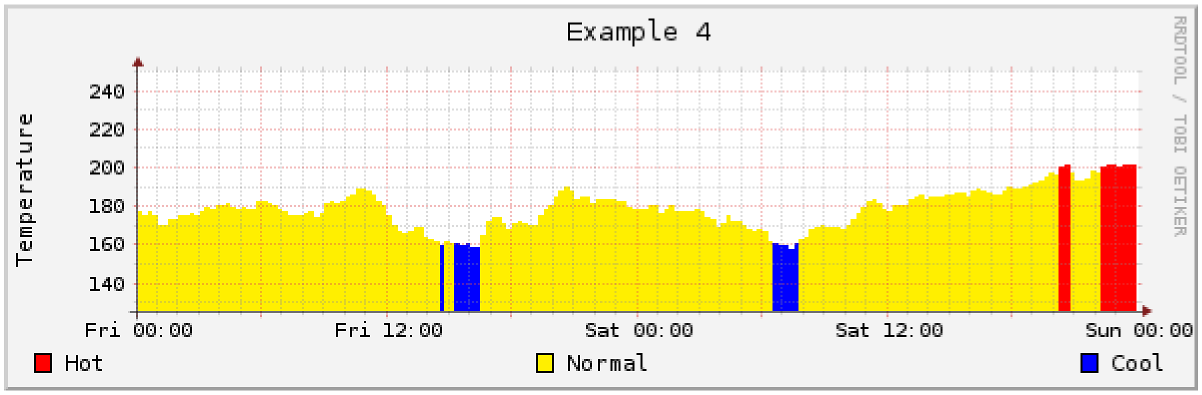
```
rrdtool graph "Example 3 VDEF.png" \
--start "end-48 hours" --end "12am Nov 1, 2009" \
--imgformat PNG --width 500 --height 120 \
--title "Example 3" \
--vertical-label "Temperature" \
DEF:temp=sysinfo.rrd:temperature:AVERAGE \
VDEF:slope=temp,LSLSLOPE \
VDEF:intercept=temp,LSLINT \
CDEF:trendline=temp,POP,COUNT,slope,*,intercept,+ \
AREA:temp#FFEF00 \
LINE2:trendline#000000
```



Example 4

This examples uses the PERCENTNAN operation in order to identify the 5% coolest and 5% hottest temperatures in the data set. These values are then applied as part of the CDEF calculations to generate the appropriate data sets for graphing. Note that it is frequently best to use the PERCENTNAN operation instead of the PERCENT operation as the PERCENTNAN variant handles any gaps in the data in a more graceful manner.

```
rrdtool graph "Example 4 VDEF.png" \
--start "end-48 hours" --end "12am Nov 1, 2009" \
--imgformat PNG --width 500 --height 120 \
--title "Example 4" \
--vertical-label "Temperature" \
DEF:temp=sysinfo.rrd:temperature:AVERAGE \
VDEF:cool5=temp,5,PERCENTNAN \
VDEF:hot95=temp,95,PERCENTNAN \
CDEF:cool=temp,cool5,LE,temp,UNKN,IF \
CDEF:medium=temp,hot95,LE,temp,UNKN,IF \
AREA:temp#FF0000:"Hot" \
AREA:medium#FFEF00:"Normal" \
AREA:cool#0000FF:"Cool"
```



Advanced Color Graphing Techniques using RRDTool

Introduction

This section covers some of the more advanced uses of color in a RRDTool graph. These techniques can not only help in providing an additional bit of polish to an otherwise ordinary graph, but can also be useful in clarifying interpretation and providing additional information.

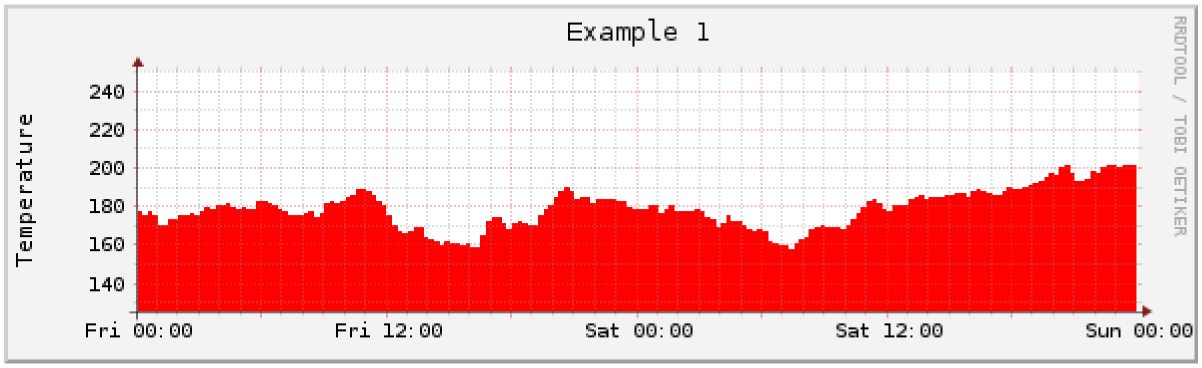
Despite the availability of the techniques outlined below, it is still essential that proper color conventions and patterns be observed. For example, red typically denotes “hot” in the context of temperature or “severe” in a notification/aberration detection. Using red to denote a cool temperature on a temperature graph or a “normal” operating condition is very likely to lead to viewer confusion. For more information, please be sure to consult a good reference on color theory.

Examples

Example 1

This is a simple example of a “stock” graph. It uses no special color treatments but it is still able to clearly convey the necessary information. In this case, it simply presents a graph of the system temperature.

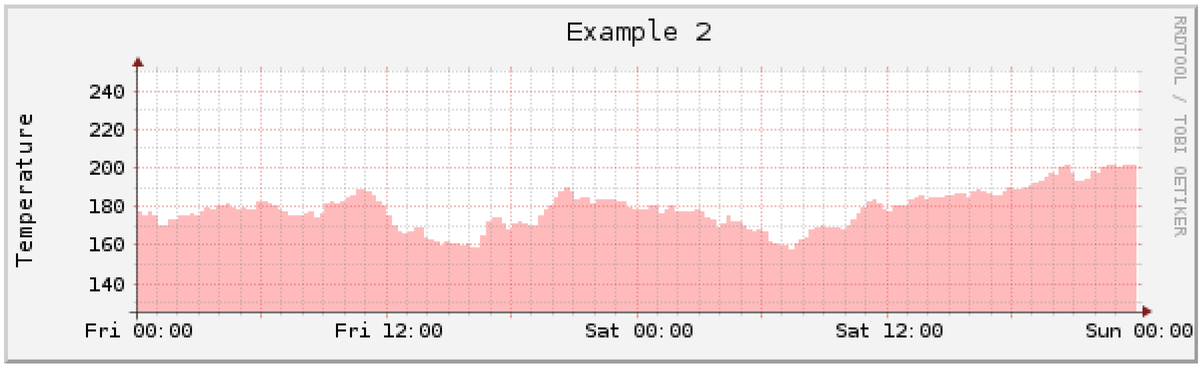
```
rrdtool graph "Example 1 Colors.png" \
--start "end-48 hours" --end "12am Nov 1, 2009" \
--imgformat PNG --width 500 --height 120 \
--title "Example 1" \
--vertical-label "Temperature" \
DEF:temp=sysinfo.rrd:temperature:AVERAGE \
AREA:temp#FF0000
```



Example 2

In this example, the use of an “alpha channel” in the color specification is introduced. The easiest way to think of an alpha channel is as a transparency measure. A alpha channel of “FF” would be completely opaque, while an alpha channel of “00″ would be completely transparent. The alpha channel is specified in hexadecimal form at the end of the RGB color value. In the example below, the color value of “FF000044″ has specified an alpha channel value of “44″. If no alpha channel is specified, then a default value of “FF” (fully opaque) is used.

```
rrdtool graph "Example 2 Colors.png" \
--start "end-48 hours" --end "12am Nov 1, 2009" \
--imgformat PNG --width 500 --height 120 \
--title "Example 2" \
--vertical-label "Temperature" \
DEF:temp=sysinfo.rrd:temperature:AVERAGE \
AREA:temp#FF000044
```



Example 3

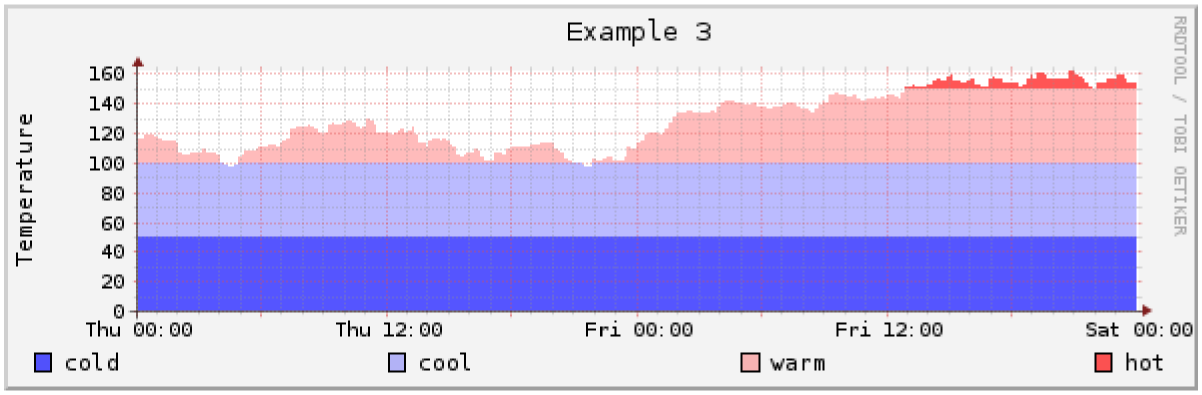
This example illustrates the use of the “layer cake” effect. This technique can help to provide additional context to a graph, as the colored layers can help clearly delineate when a system is operating within tolerances or not. Thus, the health of the system can be determined at a glance and is not dependent on the viewer being intimate with the operational thresholds. This example breaks down the temperature readings into four layers (cold, cool, warm, and hot) of 50 degrees each, but it would be a trivial extension to increase/decrease the number of layers.

The CDEF for the middle layers can be somewhat intimidating for those who are not experts in Reverse Polish Notation. In this example, each layer relies on being stacked and so the appropriate calculation is determining the portion of the temperature (if any) that makes up the layer. The following breakdown may help make it more palatable:

```
cool=temp,50,GT,temp,100,GT,50,temp,50,-,IF,UNKN,IF
if (temp > 50) then
      if (temp > 100) then
            cool = 50
      else
            cool = temp - 50
else
      cool = UNKN
```

As each layer is a maximum of 50 degrees, the trick is to determine how much (if any) of a layer falls within the designation. If the actual temperature exceeds that of the layer, then simply use the maximum value (50). If the temperature falls within the layer, then the value should be the temperature *less the total of any previous bands*. If the temperature is less than the minimal temperature for this layer, then simply return the “unknown” value to prevent any graphing.

```
rrdtool graph "Example 3 Colors.png" \
--start "end-48 hours" --end "12am Dec 5, 2009" \
--imgformat PNG --width 500 --height 120 \
--title "Example 3" \
--vertical-label "Temperature" \
--lower-limit 0 --rigid \
DEF:temp=sysinfo.rrd:temperature:AVERAGE \
CDEF:cold=temp,50,LE,temp,50,IF \
CDEF:cool=temp,50,GT,temp,100,GT,50,temp,50,-,IF,UNKN,IF \
CDEF:warm=temp,100,GT,temp,150,GT,50,temp,100,-,IF,UNKN,IF \
CDEF:hot=temp,150,GT,temp,150,-,UNKN,IF \
AREA:cold#0000FFAA:cold:STACK \
AREA:cool#0000FF44:cool:STACK \
AREA:warm#FF000044:warm:STACK \
AREA:hot#FF0000AA:hot:STACK
```

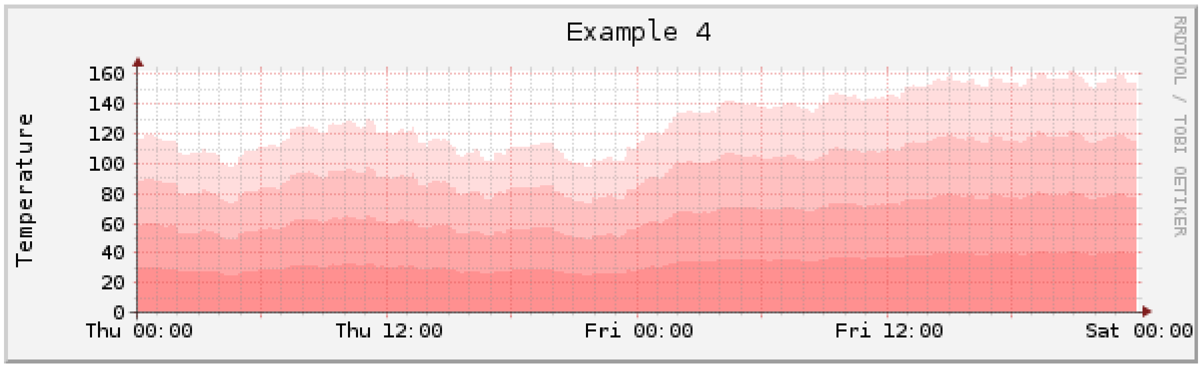


Example 4

There are several techniques for “feathering” the colors in a graph as shown in this example. The technique illustrated in this example is suitable for a representing a single color palette with the gradient lightest at the top and darkest at the bottom. It is achieved by simply overlaying the graph with the same color selection at selected proportions and relying on the alpha channel to “build up” as the layers overlap. Care should be made when using this technique not too make the top layers so translucent they become difficult to discern.

This example simply maps the set of data values into sets for 1/4, 1/2 and 3/4 values and then overlays the original value graph. Additional looks can also be achieved through the use of alternative data transformation maps/ratios.

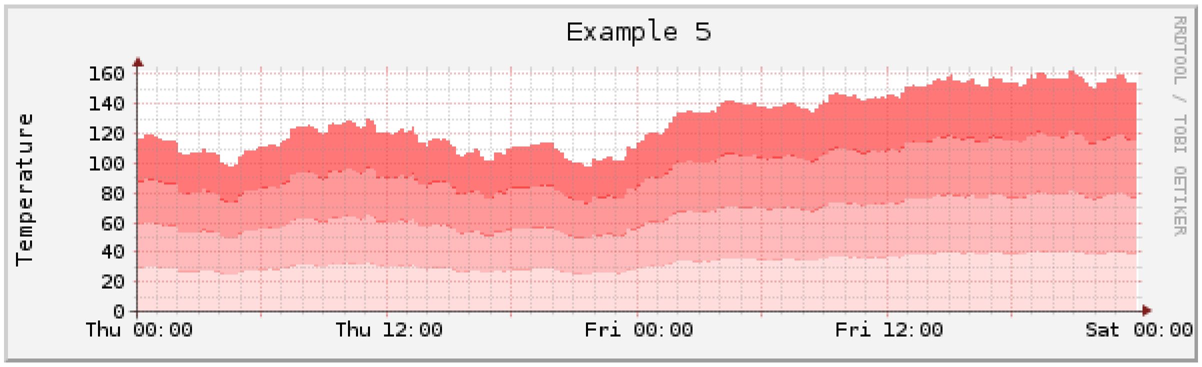
```
rrdtool graph "Example 4 Colors.png" \
--start "end-48 hours" --end "12am Dec 5, 2009" \
--imgformat PNG --width 500 --height 120 \
--title "Example 4" \
--vertical-label "Temperature" \
--lower-limit 0 --rigid \
DEF:temp=sysinfo.rrd:temperature:AVERAGE \
CDEF:tier1=temp,4,/ \
CDEF:tier2=temp,2,/ \
CDEF:tier3=temp,4,/,3,* \
AREA:temp#FF000022: \
AREA:tier3#FF000022: \
AREA:tier2#FF000022: \
AREA:tier1#FF000022:
```



Example 5

This example illustrates another method of “feathering” the colors in the graph. In this case, the color gradient is lightest at the bottom and darkest at the top. In order to achieve this, the value is simply divided up and then each layer is stacked on top of the other while steadily increasing the alpha channel value.

```
rrdtool graph "Example 5 Colors.png" \
--start "end-48 hours" --end "12am Dec 5, 2009" \
--imgformat PNG --width 500 --height 120 \
--title "Example 5" \
--vertical-label "Temperature" \
--lower-limit 0 --rigid \
DEF:temp=sysinfo.rrd:temperature:AVERAGE \
CDEF:tier=temp,4,/ \
AREA:tier#FF000022::STACK \
AREA:tier#FF000044::STACK \
AREA:tier#FF000066::STACK \
AREA:tier#FF000088::STACK
```

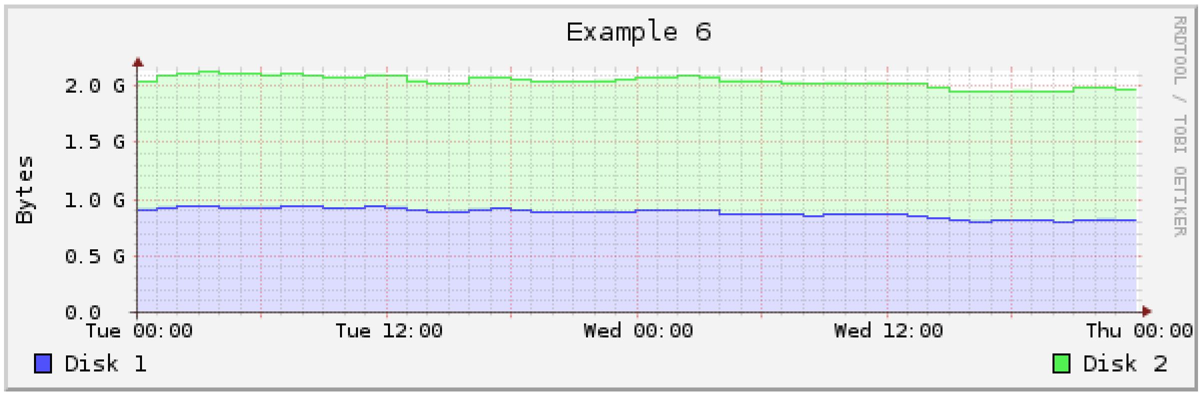


Example 6

This example illustrates the use of highlights to clearly delineate the borders between stacked area graphs. It allows the use of a softer color palette without having to resort to a clashing color scheme to define the borders.

The highlight lines should be specified after *all* the area graphs have been declared. Each highlight should be specified in the same order as its corresponding area graph in order to ensure the proper color is “on top” should the data sets have any overlap. It is typically easiest to maintain the same color scheme by using the same RGB value as the area graph but specifying high alpha channel value.

```
rrdtool graph "Example 6 Colors.png" \
--start "end-48 hours" --end "12am Jan 15, 2009" \
--imgformat PNG --width 500 --height 120 \
--title "Example 6" \
--vertical-label "Bytes" \
--lower-limit 0 --rigid \
DEF:disk1=sysinfo.rrd:disk_used:AVERAGE \
DEF:disk2=sysinfo.rrd:disk2_used:AVERAGE \
AREA:disk1#0000FF22:: \
AREA:disk2#00F00022::STACK \
LINE1:disk1#0000FFAA:"Disk 1" \
LINE1:disk2#00F000AA:"Disk 2":STACK
```


Example 6
Bytes
2.0 G
1.5 G
1.0 G
0.5 G
Tue 00:00
Tue 12:00
Wed 00:00
Wed 12:00
Thu 00:00
🟦 Disk 1
🟩 Disk 2
RRDTOOL / TOBI OETIKER