Remstats 1.00a4

Thomas Erskine

Mon Sep 10 15:18:31 EDT 2001

Contents

Chapter 1

About Remstats

Remstats is a system of programs to:

- gather data from servers and routers,
- store and maintain the data for long periods,
- produce graphs and web-pages tieing them together, and
- monitor the data for anomalous behavious and issue alerts

It's built on RRDtool (see **http://ee-staff.ethz.ch/~oetiker/webtools/rrdtool/**). There is a proto-FAQ (see the proto-FAQ section) ; feel free to contribute. There's also a to-do list (see the to-do list section) to give some idea what might be coming.

1.0.1 Where to get it

The best available version is 0.13.1. You can get it at http://remstats.sourceforge.net/release/src/remstats-0.13.1.tar.gz (see **http://remstats.sourceforge.net/release/src/remstats-0.13.1.tar.gz**). The current version is 1.00a4. (This version may not be available yet, as I will push out new documentation before the new release, to make additions/corrections available as soon as possible.) You can get other versions of remstats from the source archive (see **http://remstats.sourceforge.net/release/src**). Since almost all of it is written in perl (see **http://www.perl.org/**) scripts, there is no binary version.

1.0.2 How to get started

First, you should make sure that you have all the requirements (see the requirements section) . Then read the installation docs (see the installation docs section) . Then read the server installation docs (see the server installation docs section) .

Then check out run-remstats (see the run-remstats section) which runs almost everything else and documents how all the pieces work together.

Thank-you (see the Thank-you section) .

Chapter 2

Release Notes

2.1 Release Notes for Remstats version 1.0a3

It's always a good idea to run check-config (see the check-config section) after changing any of the config-files, but it's also a good idea to run it after doing an upgrade, especially when, as in this version, there are changes to the config-files.

Mostly small new features and bug-fixes, except:

- **Incompatible:** re-written alert-sending mechanism. Permits easily written new methods of sending alerts, by separating the alert-text generation (see alerter (see the alerter section)) from the alert-sending (see alert-email (see the alert-email section) and alert-winpopup (see the alert-winpopup section)). The new alert-destination-map (see the alert-destination-map section) config-file permits mapping an alert-destination to different addresses depending on the time-of-day, day-of-week, ... and its alias facility permits sending to a list of addresses which may use different methods of sending the alert.

- **Incompatible:** The unix-status-server (see the unix-status-server section) 's do_df now returns **bytes** not **K-bytes**. This avoids silliness in the graphs saying that you've got 20k kbytes free. It may have been correct, but it wasn't intuitive. You can multiply all the old numbers by 1000 to convert RRDs.

- **Warning:** new-config (see the new-config section) now copies the configuration files which you are likely to change, so that your changes won't be overwritten by an update to remstats. Unfortunately, as all updates (including this one) will overwrite config-base, so people upgrading from a previous version of remstats should convert the following files from symlinks to config-base into copies of those files:

  - alerts alert-destination-map general html links tools

You can do this by running the supplied convert-config-links (see the convert-config-links section) script **BEFORE INSTALLING THIS VERSION**

2.1.1 New Features

- the new datapage-status (see the datapage-status section) script generates datapages for each host, which will show the current values of all rrd variables. There is a new tool in the default tools config-file (see the tools config-file section) which will show this page, and it's been added to the defaults generated by the new-xxx-hosts (see the new-xxx-hosts section) programs.
- Host templates (see the Host templates section) . So that you can configure similar hosts like:

  ```
  %
  desc whatever
  template their-template
  ```

  Can also be used to make changing some things for many hosts easier. E.G., you could have a template, say `default-nt-status-server` which contained:

  ```
  %
  nt-status-server some-host
  ```

  and configure all hosts which use c<some-host> like:

  ```
  %
  template default-nt-status-server
  ```

- New availability-report (see the availability-report section) and availability-report.cgi (see the availability-report.cgi section) and config-file availability (see the availability section) for reporting on "availability".
- New nt-status-server (see the nt-status-server section) and nt-status-collector (see the nt-status-collector section) and RRDs for them (ntactivity, ntmemory, ntpaging, ntnetwork and ntlogicaldisk-*).
- New cleanup (see the cleanup section) program to remove stale files.
- New new-snmp-hosts (see the new-snmp-hosts section) now adds other rrds than snmpif-*
- New new-unix-hosts (see the new-unix-hosts section) program to add hosts which are running the unix-status-server (see the unix-status-server section) with the apropriate rrds.

- ought to work with perl 5.6 now. I'm not using perl 5.6 on the main collector yet, but it seems to install correctly on a test system.
- run-remstats (see the run-remstats section) now checks all configuration sub-directories to figure out if anything has changed, so you ought to be able to just edit files and the changes will get caught on the next run.
- remstats internal instrumentation allows monitoring remstats collectors, for now. More later. Look at the pseudo-host _remstats_.
- removed old Overall Index, since I never looked at it, and wrote a new RRD Index, which I was always wanting.
- new nt-discover (see the nt-discover section) program to discover and add NT systems **Note**: this adds the new discovery config-file (see the discovery config-file section) which must be locally configured. I won't even attempt to guess at values here.
- The old `alertflag` entry in the html config-file (see the html config-file section) has been replaced by three new entries: `alertflagcritical`, `alertflagerror` and `alertflagwarn`, allowing e.g. different colors for the different levels of alert.
- datapage.cgi (see the datapage.cgi section) now does variable substitution properly for HTML macros. See the example datapage upss.page under /home/remstats/etc/config/datapages.
- datapage.cgi (see the datapage.cgi section) and dataimage.cgi (see the dataimage.cgi section) have two new commands: `alertstatus` and `alertvalue` to fetch alert statuses and values. To be used on forthcoming status pages.

Chapter 3

Release Notes

3.1 Release Notes for Remstats version 0.13.1

I fixed a minor buglet in 0.13.0 which was noticed shortly after release. I was annoyed enough with it that I made 0.13.1.

Chapter 4

Release Notes

4.1 Release Notes for Remstats version 0.13.0 (AKA 0.12.2)

There are lots of little improvements, which are detailed in the Change History (see the Change History section) , which I'm not going into here. The main incompatible changes are:

- The configuration structure ($main::config) now has the graphs stored under $main::config{RRD}{$wildrrd}{GRAPH} instead of $main::config{GRAPH}, so there won't be problems with having the same graph-name defined under two different rrds. This will only affect you if you've been writing your own code for remstats, like a new page-maker. I thought that the bug was annoying enough and difficult to figure out when it was triggered that the incompatibility was worth the change.
- After typing $main::config{CUSTOMGRAPH} instead of $main::config{CUSTOM} one too many times, I renamed $main::config{CUSTOM} to $main::config{CUSTOMGRAPH} which is what it should have been all along. Again, this should only affect you if you've been writing your own remstats code, like a new page-maker.
- Changed default location for datapages to /home/remstats/etc/config/datapages, so that all the configuration, including the datapages are together.
- Removed the general config-file directive `pagesas` since all the generated pages are CGIs now. Check-config will abort if you still have it. Just delete that line in the general config-file.
- `use strict` in all the scripts (unless I missed some) in preparation for perl 5.6, which doesn't like `use vars`. Shouldn't bother you unless you've been writing remstats code, in which case, you probably know what to do.
- To deal with alert templates (see below), you'll need to manually fix your config-dirs. For each one, you need to:

```
%
su remstats
cd your-config-dir
cp /home/remstats/etc/config-base/alert-template-map .
mkdir alert-templates
cp /home/remstats/etc/config-base/alert-templates/* alert-templates
```

- remoteping-collector has been modified to return the server-name instead of a number to differentiate the data from different servers. There's also a new remoteping-* wildcard RRD to make it more usefull.

4.1.1 Customgraphs on host-index pages

The new `customgraph graphname` directive for host config-files permits you to add a customgraph to a host-index page. (Thanks Marek.)

4.1.2 graph.cgi - remstats graphs anywhere

Like it says, using the new graph.cgi (see the graph.cgi section) , you can put remstats graphs on any page you want.

4.1.3 Views

You can now define your own pages with page-layout of your choice using views (see the views section) . (Thanks to Marek and Thorsten and Matt and anyone else I've forgotten.) Don't forget to add view-writer (see the view-writer section) to the list of pagemakers if you've changed the default.

4.1.4 ping-* rrd

You can now ping different interfaces on a host separately (Thanks Steve)

4.1.5 fileage section for unix-status-server

This allows you to fetch the last-modification-time for specified files. It was written to allow remstats to monitor lock-files to check for stale locks. There is no included rrd using it as lock-files are all over the place.

4.1.6 port-collector can collect data from results

The port-collector (see the port-collector section) has always been able to send a string to remote services to that it could tell if they were working correctly. Now it can pull values for RRDs and status-files from the results as well. I've included a sample rrd (weathernetwork) and script (weathernetwork) to collect current weather data for Ottawa. Look at the updated docs for scripts config-files (see the scripts config-files section) .

4.1.7 new script - snmpif-description-updater

The snmpif-description-updater (see the snmpif-description-updater section) will keep the descriptions on snmpif-* RRDs up-to-date with whatever you've set as ifAlias for that interface. (Thanks Steve Francis.)

4.1.8 Alert Templates

This feature allows you to customize the alert messages by addressee or by RRD. Look at the docs in alert-template-map (see the alert-template-map section) and alert-templates (see the alert-templates section) .

4.1.9 Autoconf-like configure

You can now do:

```
%
./configure
make
make install
```

for the beginning of of the install (see the install section) .

Chapter 5

Release Notes

5.1 Release Notes for Remstats version 0.12.1

Ideally, this document will only have to tell you about the great new features of remstats in this version.

Not this time.

In addition, due to various stuff (read the Change History (see the Change History section)), this covers changes since version 0.11.1.

5.1.1 Configuration File Replaced by Configuration Directory

The old "one huge configuration file" has been replaced by a directory of files and sub-directories. (See the new configuration docs (see the new configuration docs section) for details.) This means that most programs don't need to read and parse everything, including stuff that they're not going to use. It also makes it easier to find things, as you can go directly to the file that has what you want, e.g. details on a particular host. It also made possible the newly revamped replacements for `make-ping-hosts`, `make-port-hosts` and `make-snmp-hosts`, which will insert their additions directly into the appropriate configuration files.

There is a new script, split-config (see the split-config section) , which will take your old config-file and a new name and generate a new config-dir from it.

On a related note, I broke the groups (see the groups section) line out of the `general` config-file into its own file. It's easier to see what you've got. `split-config` will do this for you. Also the (undocumented) [html] section will absorb large portions of the [general] section which really belong to wep-page generation.

If you've made your own collector, you'd better look at the new skeleton-collector for the required changes. Just change `read_config` to `read_config_dir`, with extra args. There's also documentation on how to write your own collector (see the collector section) .

5.1.2 do-remstats replaced by run-remstats

The old `do-remstats` shell-script and all the kludgy shell-scripts that went with it and the `watchdog` and `lockfile` scripts have all gone away. The replacement run-remstats (see the run-remstats section) does everything they did and does it correctly. It's also configurable, so you don't need to modify the scripts to change which collectors you want to run, e.g.

A new feature of `run-remstats` configurability is that you can have it run the `ping-collector` before everything else and not bother trying hosts that didn't answer it. You can also choose which collectors (see the collectors section) , monitors (see the monitors section) and pagemakers (see the pagemakers section) to run.

5.1.3 CGI scripts and non-default config-dirs

At the moment, the supplied CGI scripts don't deal with non-default config-dirs. I do consider this to be a problem, but I need to get this release out to deal with other serious installation problems.

You can work-around this by editing the installed CGI scripts and putting in the correct definition for $config_dir, near the top.

5.1.4 plugin-collector is gone

It was an inefficient, difficult-to-configure, kludge and isn't needed anymore with the new run-remstats.

5.1.5 pre-release testing automated

You won't see it, but I hope you'll all notice the improvement in release quality.

5.2 Known Bugs for version 1.00a

- Alerts in general - The alerts shown by the alerts.cgi (see the alerts.cgi section) page never get expired. If a hub is down, then you'll get alerts for everything behind it. Ought to only get the alert for the hub.

5.3 Known Bugs for version 0.12.2

- Neither new-snmp-hosts, nor snmp-collector use get_ifname, with the consequence that neither copes well with "oddly" named interfaces, say with spaces in them. Fixed in 0.12.3.

5.4 Known Bugs for version 0.12.1

- CGI scripts don't work with non-default config-dirs. I consider this a bug, but I need to get this release out now to deal with serious installation problems with previous releases. For now, do the following for each config-dir:

```
%
\% make install-cgis CONFIGDIR=/wherever/you/put/it
```

- run-remstats only checks the config-dir for change, not the subdirectories. For now, just `touch config-dir` whenever you make a change.
- customgraphs are completely broken. Urgh. Upgrade. [FIXED in 0.12.2]
- You can't have two graphs of the same name, even in different rrd definitions. This is just flat-out wrong and will be fixed. Unfortunately, the fix will mean even longer file-names, so I hope nobody has some old system with the 14-character limit. [FIXED in 0.12.2]
- graphs with descriptions can't have quotes in the description. [FIXED in 0.12.2]

5.5 To-Do List for Remstats

5.6 High Priority

134 20010829 [**LOW**] - make header_bar (in htmlstuff) do the link making, if available and fix whatever uses it not to.

133 20010829 [**LOW**] - add an option to make nt-discover update old hosts with a standard set of RRDs, even if the hosts are already known

132 20010824 [**HIGH**] - BUG: get rid of the spikes in uptime from the unix-status-server

131 20010824 [**MED**] - make status pages for each host, group and for all hosts using the new alertstatus and possibly alertvalue.

130 20010823 [**HIGH**] - add an <RRD::EXEC ...> tag to rrgcgi. To by used in host index pages (see 129).

128 20010629 [**MED,HOLD**] - custom, configuration-supplied info per rrd which is simply available wherever it makes sense, e.g. in alerts.

- first make sure someone has a use for it.

127 20010622 [**MED**] - graph data together with historical data. This will probably mean either populating another rrd with historical averages, temporarily or permanently, or modifying rrdtool. The former is certainly simpler to do, given my knowledge of the internals of rrdtool. However, it needs to have another rrd for each period? Need to keep the same data over some longer period, a multiple of the period of interest, as well as the averages, from period to period.

122 20010330 [**HIGH**] - rrd prog-* which tells if a particular named process is running, using the ps section of the unix-status-collector.

121 20010202 [**MED,HOLD**] - how about an discovery program, to find and identify hosts and then run the appropriate new-xxx-hosts scripts to add them?

DONE 20010608 - nt-discover to find and add NT boxen

115) 20001229 [**HIGH**] - need docs on errors. Specifically, when run-remstats kills a collector for taking too long. And where to find the output of the killed collector.

112) 20001212 [**LOW,HOLD**] - web-based remstats configurator. Needs to consider security, at least from the point of view that you don't want to lose your configuration. The most important part is hosts. A lot of the rest doesn't have to be changed, or only once.

111) 20001212 [**LOW,HOLD**] consider grafting on (at least links to) some kind of system configuration interface. For configuring the mmonitored entities, not remstats.

110) 20001212 [**LOW,HOLD**] consider problem-fixing interface. It'd be nice to try to fix things if there is a known way to do so. A simple kludge would be to add another method to the alert-destination-map which deals with problems that it knows about, possibly invoking plugins for specific alerts.

109) 20001212 [**MED**] nt-log-collector, with modules for event-logs and ntmail logs.

70) 20000407 [**HIGH**] CGI scripts need to have a way to deal with alternate config-files, and graph-writer needs to tell them if they can't work it out themselves. Otherwise, people need to be told to do multiple installs of the CGI scripts, which might be the best way.

```
%
make install-cgis CONFIGDIR=config-xxx
```

Not that painfull, but wastefull and makes upgrade messier.

- I don't like the multiple-install method, but any other method needs a way of getting configuration information into the CGI scripts. Any method which passes info in via the URL or form fields is out: too unsafe. The only other method I can think of is to read a configuration file in the same directory as the CGI script. This ought to be safe from modification, or your web-site is waiting to be mutilated. The other part to consider is whether any part of the info in the CGI config-file is sensitive. I.E. do we have to protect it in some way.

- Configuration file in the same directory won't work either, you'd still have to install the cgi's multiple times. I'm starting to think that multiple installations may be the only safe thing to do.

99) 20000619 [**HIGH**] make unix-status-collector send the directories that we want df for and make unix-status-server do "df /dir1 /dir2" to get them, and pull them off one line at at time. This is to deal with things like disconnected NFS-mounted directories hanging df when we do just a bare "df".

86) 20000419 [**HIGH**] trends analysis

87) 20000419 [**HIGH**] alerts based on trends analysis and historical data, like one-week average and standard-deviation, ... (for Steve)

106) 20000922 [**MEDIUM**] make a file-collector. Similar to the log-collector, only for small, local files. Slurp the file into memory, match patterns and pull out values. The data line in an rrd definition would be like:

```
%
source file
data VARNAME GAUGE:600:0:U FUNCTION PATTERN(WITH)PARENS
```

in fact, this would share so much code with the log-collector that it might be worth combining the two. This allows collection from things like Linux's /proc.

98) 20000619 [**MEDIUM**] add group index files and store hosts under group directories. For easier application of access-controls. (for Florian)

2) ???????? [**MEDIUM-INPROGRESS**] make rrd munger, like copyrrd was supposed to be use dump/restore and process the xml form (rrddump-munger) what functions do we need? Make one script for each function.

- add a DS (less important, as we can just make a new rrd)
- remove a DS (less important, as we can ignore it)
- add an archive
- extend an archive
- change CF of an archive
- remove an archive
- filter data within an archive
  - change NaN to number/max/min
  - change ># to NaN/max/min
  - change <# to NaN/max/min

5.7 Lower Priority

102) 20000912 [**LOW**] add see-also to host config, which will materialize links in the host header. Config line like:

```
%
seealso host:xyzzy http://www.somewhere ftp://ftphost
```

the special "host:" pseudo-URL gets changed to a link to the remstats page for that host.

103) 20000915 [**MEDIUM**] make-path doesn't work with non fqdn hosts Make it read the configuration, so it can look up the IP number in the host config and use that if it's defined. Otherwise, default to gethostbyaddr.

107) 20000922 [**MEDIUM**] extra status header lines for hosts, from specified STATUS files creaded by the various collectors. Add lines to host definition like:

```
%
extrastatus "STATUS DESCRIPTION" STATUS-FILE-NAME
```

60) 20000328 [**MEDIUM**] replace route-collector with something which scales. SNMPwalking bgp4PathAttrBest doesn't scale to large Internet routers with 400 peers, taking over an hour to complete. (see also 61)

- look at a script to follow the output of zebra. That's a lot of overhead though. Easy if zebra is solid.

- How difficult can it be to make a native BGP listener? I'm not clear on the protocol, but it doesn't look too bad.

45) 20000121 [**MEDIUM**] make snmp-collector send only one packet per host

- test and make sure that we do get back whatever succeeded. I vaguely remember that it didn't work. [Later: at least under UCD snmp under linux, if an item isn't implemented in the MIB, you get back NOTHING. Specifically, look for the non-unicast packet counters as well as something else; you get nothing back. This isn't good.]

- have to re-write snmp-collector completely, which isn't that bad an idea. This means a two-pass structure. On pass one, we construct the complete query and then send it. On pass two, we examine all the results and format them.

9) ???????? [**MEDIUM-TESTING**] make alerts take connectivity dependence into account

- add "via" line to host section to deal with hubs and switches [DONE]

- I think it's done. See what happens next outage.

42) 20000114 [**MEDIUM**] snmp-collector mod to allow summary data collected from a walk and then filtered as a single data-point. E.G. specify a rrd "oid" like:

```
%
walk    count ifOperStatus = 1
```

would produce a count of the number of interfaces on that device that were active (i.e. had a live device plugged into them). Or a similar one would let you count BGP routes, or arp addresses, ...

- Unfortunately, from experience with the snmp-route-collector, this is going to be slow for anything with a large number of items.

43) 20000114 [**MEDIUM**] parallelizing the collectors, at least on a group basis, preferably host or group.

- collectors must accept `-G` and `-H` flags to request processing of the specified group or host, respectively. Run-remstats needs to fork extra processes according to a config-file line, "parallel group" or "parallel host".

- 20010831 TEE - implemented -H flags for all collectors except for the remoteping-collector, which I'm not using anyway right now.

51) 20000216 [**LOW**] need a way to specify URL for port-http. The root page doesn't always exist.

37) 19991216 [**LOW**] traceroute sometimes shows incorrect routing, which confuses the topology-monitor, causing false positives

50) 20000215 [**LOW**] make inventory script. Runs uname (for hardware and software), `ifconfig -a`, `netstat -nr`, `hostname` and any others I can think of to collect configuration info. Then figures out the versions of important software, e.g. run `perl -v`, `gcc -v ...` Make a subdir to put it in and make a tool definition to get it onto the host pages.

- looks like the beginning of a discovery script.

62) 20000329 [**LOW**] make different markers for different levels of alert on quick-index.

69) 20000406 [**LOW**] is there any use for write_environment in check-config?

97) 20000616 [**LOW**] make port-collector or check-config complain about having a script with ok/warn/error/critical patterns but no send string. The port-collector will ignore patterns unless there is a send string.

5.8 On Hold

Usually waiting for next major release, or trapped by something else. (in priority order)

40) 20000104 [**MEDIUM-HOLD**] consider some form of access-control for servers

- hash-based "password"

- ssl tunneling ought to work for everything except SNMP

- what does this buy? With the various servers run under tcp_wrappers an attacker must either gain access to the remstats collector machine or spoof a tcp session from them. If you've been "owned" you've got bigger problems. If the attacker spoofs a session with a remstats server, tcp-wrappers will insist that it must come from one of the allowed hosts, so that's where the stolen output will go. This is only usefull to the attacker if they have access to the remstats collector machine or if they can sniff the traffic between the collector and the server. The only data loss possible is with the log-server which keeps state. (Ignoring DOS attacks which are always a problem.)

- unless someone needs this, it's on hold

6) ???????? [**LOW-NEEDS:2-HOLD**] increase CA3 resolution

- need rrd munger (2)

10) ???????? [**LOW-INPROGRESS-HOLD**] make graph of connectivity

13) ???????? [**LOW-INPROGRESS-HOLD**] snmp trap listener to update status files

- needs filter to be usefull [DONE]

- I haven't seen any useful traps so this is on hold.

14) ???????? [**LOW-NEEDS:2-HOLD**] make rrd structural changes in config file get applied to the rrds.

- some taken care of with snmpif-setspeed, but need a more general solution

- look at new XML output of rrddump

39) ???????? [**LOW-HOLD**] make RRD dumper, to put data out in a form that can be loaded into a database

- I don't need it, per se, but it might be easier than writing the availability report generator.

52) 20000215 [**LOW**] make a makegraph.cgi, or whatever, that will let you make a somewhat custom graph on the fly. makegraph.cgi by itself will list all the hosts and let you choose one. makegraph.cgi?host=xxx will list all the RRDs for this host and let you choose ?one?. makegraph.cgi?host=xxx&rrd=yyy will list the various DSs for this RRD and let you choose the ones you want. Then you get to define any CDEFs needed and then LINEn/AREA/STACK for each DEF or CDEF desired. And size, title, legends...

- On hold since graph.cgi (see the graph.cgi section) will let you get at any existing graph you want. If I find a use or need for this, I'll re-activate it.

92) 20000518 [**HIGH**] collect traffic info from cflowd (artsportms). Make it flexible enough that it can let you choose which ports you want (one per rrd?). Make a loader to load historical data.

- [DONE 20000524] artsportms-loader done

- I no longer have access to devices with this feature

I've also kept the stuff that used to be here, but has already been done (see the done section) .

Chapter 6

FAQ

6.1 FAQ for remstats

This is only a proto-FAQ, with stuff I couldn't figure out where to document.

1. **snmp-collector complains that I don't have SNMP_Session installed, but I don't want SNMP. Do I have to collect and install it?** No you don't. Modify the `collectors` line in the general (see the general section) config-file to exclude snmp.

2. **I modified the RRD definition, adding a new RRA, but remstats is ignoring it. How do I do this?** Sorry. At the moment, remstats won't propagate any changes to the RRD structure after creation. Some changes, like extending RRAs and changing min/max for DSs can be done manually with rrdtool. If you do use rrdtool manually, I recommend that you modify the rrd definition as well, to keep them in sync. At some point in the future, I'd like to try to do this kind of update, and it's more likely to succeed if remstats' rrd definition matches what's in the actual RRD.

6.2 Documentation Conventions

The only documentation conventions the reader has to know about are:

- things inside [square brackets] are optional
- parenthesized lists with the items separated by vertical bars, (like — this — one) require that you choose one and only one of the alternatives.

Everything else ought to be explicit. If it isn't, or if you don't understand it, please bring it to the author's attention, stating which part you don't understand. There's not a lot of point in my writing documentation which no-one else can understand. I'd rather do it right.

Chapter 7

Prerequisites

7.1 What you need to install remstats

1. You'll need perl (see **http://www.perl.org/**), at least version 5.005_03.
   If you don't already have it you can get it from CPAN (see **ftp://ftp.crc.ca/pub/packages/lang/perl/CPAN/src/stable.tar.gz**) (the Comprehensive Perl Archive Network).

2. You'll need a `C` compiler that works. :-) gcc or egcs (see **ftp://ftp.crc.ca/pub/packages/egcs/**) will do fine and you can find them easily in many different places.

3. Make sure you have the following perl modules installed (most of which you can find at CPAN (see **http://www.cpan.org/modules/by-module**)). The versions are the versions I'm using, but more recent versions should work too, unless there have been radical changes. They should be installed in the listed order to avoid dependency problems:

   - RRDs 1.0.16 (see **http:src/rrdtool-1.0.29.tar.gz**) - the key piece. It comes with RRDtool and does the database and graphs. Originally from http://ee-staff.ethz.ch/~oetiker/webtools/rrdtool (see **http://ee-staff.ethz.ch/~oetiker/webtools/rrdtool**).
   - Socket - should be standard if you've got the required perl
   - IO::Socket - should also be standard in the requires version of perl
   - Time::HiRes 01.19 (see **http:src/Time-HiRes-01.20.tar.gz**) - used by the port-collector to determine response time. You can comment out the `"use Time::HiRes"` line and the program will still work, but the response-time resolution will be one second instead of one millisecond. Originally from CPAN (see **ftp://ftp.crc.ca/pub/packages/lang/perl/CPAN/modules/by-module/Time**).
   - SNMP_Session 0.69 (see **http:src/SNMP_Session-0.77.tar.gz**) - used by the snmp-collector. If you don't need SNMP, you can leave it out, but you'll have to change the config file. (Modify the `collectors`

line to leave out `snmp`.) Originally from ftp://ftp.switch.ch/software/sources/network/snmp/perl/ (see **ftp://ftp.switch.ch/software/sources/network/snmp/perl/**).

- GD (see **http:src/GD-1.30.pm.tar.gz**) - used only by dataimage.cgi (see the dataimage.cgi section) to create images on the fly. Originally from http://stein.cshl.org/WWW/software/GD/GD.html (see **http://stein.cshl.org/WWW/software/GD/GD.html**).

4. You'll also need the following programs for the `unix-status-server`. (You can change the locations at the top of it.) You almost certainly have most of these and can ignore any that you don't tell the `unix-status-collector` to query. For details, look in the unix-status-server (see the unix-status-server section) docs:

```
%
uname, vmstat, df, uptime, netstat, ps, ftpcount, qmail-qstat and
qmail-qread.
```

Chapter 8

Installation

8.1 How to install remstats

READ THE RELEASE NOTES (see the RELEASE NOTES section) FIRST. This page is generic and does **NOT** include version-specific instructions.

I know that this is not simple. I do plan to make it simpler, but it'll **never** be "`./configure; make; make install`" because I don't know what you want to monitor. The two C programs (multiping (see the multiping section) and traceroute (see the traceroute section)) now use autoconf, and the main configure script works (from the outside) simlarly to an autoconf-generated configure. I haven't seen a need to convert it to autoconf yet. It's mostly perl scripts and if you have the right version of perl properly installed, it shouldn't need anything special. The `unix-status-server` is a slight exception to this, but the only configuration needed so far is done dynamically and is only the location of the various required utilities.

1. Unpack the distribution tarball:

   ```
   %
   gunzip -dc remstats.tar.gz | tar xf -
   ```

2. create the remstats user and group, if you haven't already, (by default `remstats` and `remstats` respectively.) (See also the remstats user (see the the remstats user section) .)

3. Build and install the software. If you're upgrading, you might want to take a copy of fixup.config from the old version:

   ```
   %
   sh configure
   ```

   If you want to override the defaults, then run

```
%
sh configure --help
```

for a list of what can be overridden.

[Check fixup.config to make sure it is properly setup.]

```
%
make all
make install
su -c 'make install-suid'
```

Note: this step also customizes the programs and documentation with your choice of directories, owner, ... so this documentation should refer to your setup after you've done the install.

The `make install-suid` simply makes traceroute and multiping suid root. They won't work most places unles run as root, one way or another. Since I don't like to run all of remstats as root, this was the best compromise I could come up with.

4. fix the config-base for site-specific things. Edit the following files in /home/remstats/etc/config-base, looking the the string "FIXME", without the "quotes".

```
%
alerts general html scripts/http-proxy
```

I'll try to keep this list up to date, but you can make sure by doing:

```
%
grep -l FIXME /home/remstats/etc/config-base/* /home/remstats/etc/config-base/*/*
```

5. Make a config-dir (see the config-dir section) to describe what you want to monitor. You can do this by hand, or using the configuration building tools. To use the tools, you'll have to make a few files listing various kinds of hosts:

```
%
cd /home/remstats/etc
/home/remstats/bin/new-config config
/home/remstats/bin/new-ping-hosts groupname1 group1-hosts-file
/home/remstats/bin/new-ping-hosts groupname2 group2-hosts-file
...
/home/remstats/bin/new-port-hosts groupname3 port-hosts-file
/home/remstats/bin/new-snmp-hosts groupname4 SNMP-community-string snmp-hosts-file
```

After you've installed the unix-status-server (see the unix-status-server section) on some hosts, you can also use:

```
%
/home/remstats/bin/new-unix-hosts groupname5 unix-hosts-file
```

If you have any Windows NT hosts that you want to monitor, after you have installed the nt-status-server (see the nt-status-server section) , you can run nt-discover (see the nt-discover section) to find and add the NT hosts for a given NT domain.

If you're going to use the log-collector, you'll have to build the rrd entries for each by hand. There doesn't seem to be much standard in where log-files go, let alone what's in them.

6. Arrange for cron to run run-remstats (see the run-remstats section) at an appropriate interval. For a five-minute interval, something like the following will do:

```
%
0,5,10,15,20,25,30,35,40,45,50,55 * * * * /home/remstats/bin/run-remstats
```

This checks the configuration, collects the new data, updates the rrds, runs the monitors to compute statuses and updates the web-pages. Note: it does **not** re-write the web-pages for every iteration; it only does so when configuration files have changed, as the web-pages will show new data by themselves.

optional Arrange for cron to run `do-traceroutes` at an appropriate interval. You could run it in the wee hours of each morning like:

```
%
5 3 * * * /home/remstats/bin/do-traceroutes
```

This information isn't currently used, but I'm planning to make use of it.

optional Arrange for cron to run snmpif-description-updater (see the snmpif-description-updater section) periodically, if you have any snmpif-* RRDs, which you're likely to change the descriptions on. Say every day, like:

```
%
0 3 * * * /home/remstats/bin/snmpif-description-updater
```

7. Arrange for cron to run cleanup (see the cleanup section) every now and then to remove old un-needed files, like:

```
%
0 3 * * * /home/remstats/bin/cleanup
```

This removes no-longer-needed files, like old host graphs, traceroute results, log-files, ...

8. You'll need to set up your web-server (see the web-server section) to allow CGI scripts in the remstats html tree and make sure that you're not allowing everyone in.

9. Make a symlink in the html directory from whichever index you prefer to index.cgi.

10. You'll want to look at the server installation docs (see the server installation docs section) if you're going to be running any of the remote servers (log-server (see the log-server section) , nt-status-server (see the nt-status-server section) , remoteping-server (see the remoteping-server section) , and unix-status-server (see the unix-status-server section)).

Enjoy your pretty pictures and I hope that you find them usefull.

8.2 The remstats user

You **must** choose a userid to run the remstats processes under. By default, it will be the user `remstats`, but you'll have to create it manually, as I'm not going to risk damaging someone's /etc/passwd file. Many operating-systems have a script called `useradd` or `adduser` or some variant on that.

NOTE: Don't run the remstats programs except as the remstats users. Many of the programs write extra files you won't know about unless you read the source, and when you do run them as the remstats user, it won't be able to modify the files that were created by the other user. This will probably cause the program to die, with a meaningful error message I hope, and you'll have to modify the owner by hand, as root. If you need to do this, go back to the source directory and do:

```
%
\% su -c 'make install-owner'
```

The remstats user must be able to write files within the remstats directory trees rooted at /home/remstats, /home/remstats/data and /home/remstats/html. The collection/update processes will also create files under /home/remstats/tmp and /home/remstats/data. The pagemakers write files under /home/remstats/html. It's simplest to have all the remstats files and directories(except multiping (see the multiping section) and traceroute (see the traceroute section)) owned by the remstats user.

You must also ensure that the CGI scripts (and almost every web-page remstats creates is a CGI script) get run by the remstats user. The CGI scripts read files under /home/remstats/data and /home/remstats/datapage. (See also the web-server installation (see the the web-server installation section) .

8.3 Magic Cookies

There are various places in the configuration file where you can have text substituted for you. It's a (very-limited) macro facility. Currently, it only works in graphs, scripts and tools. The cookies are always UPPERCASE and the name is surrounded by "##", so that a request for "color1" would look like "##COLOR1##", without the quotes.

Here they are:

8.3.1 Colours:

You're not required to use these, but you'll really regret not doing so when you decide to change colors later.

- COLOR1, COLOR2, ... COLOR6 and also COLOR1a, ... - generic colours for graphs
- PROBLEMCOLOR - an alarming colour
- TOTALCOLOR - the full amount of something
- USEDCOLOR - how much is used of something

8.3.2 Other stuff:

- DB - the full path and file-name of the current rrd
- DATADIR - where the data files live
- GRAPH - the name of the current graph
- GRAPHTIME - the name of the current time-span
- HOST - the name of this host
- IP - the IP number of the host
- IPORHOST - the IP number of the host, if it's defined in the host's config-file, or else its name.
- HTMLURL - where the html and graphs live in web-space
- RRD - the name of the current rrd, without the .rrd extension or file-name fixing
- SHORTHOST - the name of the host before the first dot, unless it is a generic name like www or ftp or mail
- THUMBHEIGHT, THUMBWIDTH - how large to ask for the thumbnail to be. Not how large the resulting gif is.

- WEBMASTER - who is responsible for this web presense
- WILDPART - the instance part of a wild rrd. If if-* is being used by if-le0, then WILDPART will be le0.

8.4 private.pl - configuration-supplied functions

There are several places where you can have functions you choose invoked by remstats:

- updater (see the updater section) permits functions to be applied to incoming data via the rrd definition (see the rrd definition section) .
- datapage.cgi (see the datapage.cgi section) and dataimage.cgi (see the dataimage.cgi section) permit functions to be applied on any `eval` line.

Rather than me continually adding functions or you having to hand-modify your copy of remstats whenever I make new releases, I've supplied an almost empty perl file called `private.pl` which will be installed in the /home/remstats/lib if you don't have one, but will never be modified by me after that.

8.5 Server Installation

One of the interesting things about remstats (I think), is the remote servers (see the remote servers section) . To install them, you'll need to do a few things on each host which will run the servers:

- Add entries for the servers in `/etc/services`, like this:

  ```
  %
  unix-status 1957/tcp \# remstats unix-status server
  log-server 1958/tcp \# remstats log server
  ```

  You can run them on different ports, but these are the defaults and you'd have to change run-remstats (see the run-remstats section) to add the appropriate switches.

Optional Unless you're going to run the servers as root (unnecessary), you'll need to create the user that the servers will run as. The only reasons I can think of for running the servers as root are if you need to run them on a low-numbered port (<=1024), or if you need to read a log-file which isn't readable by the remstats user, or if you want to run multiping non-suid. On Linux and Solaris, you can do:

```
%
groupadd remstats
useradd -g remstats -d /home/remstats remstats
```

- Modify `/etc/inetd.conf` to get the servers invoked, like this:

  ```
  %
  unix-status stream tcp nowait remstats /home/remstats/unix-status-server unix-status-server
  log-server stream tcp nowait remstats /home/remstats/log-server log-server logfile1 logfile2
  ```

  Or if you're using tcp_wrappers (see **ftp://ftp.porcupine.org/pub/security/tcp_wrappers_7.6.BLURB**), which you should be:

  ```
  %
  unix-status stream tcp nowait remstats /path/to/tcpd /home/remstats/unix-status-server
  log-server stream tcp nowait remstats /path/to/tcpd /home/remstats/log-server logfile1 logfile2
  ```

  And remember to update `/etc/hosts.allow` to allow your remstats host access.

- Tell inetd to re-read it's config-file:

  ```
  %
  kill -HUP pid-of-inetd
  ```

- copy the remstats servers to the machines which will run them

  ```
  %
  rcp unix-status-server log-server remoteping-server multiping host:/home/remstats
  ```

8.6 The nt-status-server

This one is a bit different to install. I've only done it under the ActiveState Perl (see **http://www.activestate.com/Products/ActivePerl/index.html**) under Windows NT 4.0. Installing the ActiveState Perl is straightforward; if you got here, you'll have no trouble with that. Installing it as a service is not as simple as I intended. You'll have to get the SRVANY and INSTSRV programs from the NT Resource Kit, and follow their instructions. The program that SRVANY will be running is, of course, perl (usually `C:\Perl\bin\perl.exe`), and the argument string something like:

```
%
c:$\backslash$wherever$\backslash$you$\backslash$put$\backslash$nt-status-server -s -t10.111.12.13
```

You'll have to replace `c:\wherever\you\put` with the path to nt-status-server (see the nt-status-server section) , and `10.111.12.13` with the IP number of the host running the nt-status-collector (see the nt-status-collector section) .

8.7 Getting your web-server ready for remstats

8.7.1 Choosing userid for remstats

Almost all the remstats web-pages are generated by some kind of CGI script. Many of them will read additional files not available under the html directory tree. In order to provide access to these files, you need to make sure that the scripts get run as the remstats user. The simplest way to do this is to run a separate instance of the web-server software as the remstats user. You may have other methods of accomplishing this, depending on the web-server you're using. (See also remstats user (see the remstats user section) .)

8.7.2 Running CGI scripts under the remstats tree

You also may need to tell your web-server that `xxx.cgi` means that this file is a CGI script and needs to be run, instead of just displayed. With the apache (see **http://httpd.apache.org**) web-server, you could add the following lines to the `httpd.conf` file:

```
%
$<$Directory /home/remstats/html$>$
Options FollowSymlinks ExecCGI
AddHandler cgi-script .cgi
$<$/Directory$>$
```

8.7.3 Restricting access to CGI scripts

There are a few things you should do before telling others about remstats. Remstats comes with a few CGI scripts which you probably don't want to make publicly available and two that you certainly don't. `ping.cgi`, `traceroute.cgi` and `whois.cgi` should probably be restricted to your own organization, unless you don't mind letting anyone on the Internet run pings, traceroutes and whois queries from your domain. Rectricted to your domain, you only have to worry about your own people.

However, `alert.cgi` and `log-event.cgi` are a different kettle of fish. They will permit anyone who can run it to quench alerts and log comments about them. You will probably want to be a bit more restrictive about who you let run this.

Using the apache (see **http://httpd.apache.org/**) web-server, you can restrict the use of these CGIs using a `.htaccess` file something like this:

```
%
\# Note that this example uses the private network 192.168.0.0.
\# Stuff to make Apache expire the files to get them refreshed
ExpiresActive on
\# images every 5 minutes, when the data gets updated
ExpiresByType image/gif M300
```

```
ExpiresByType image/png M300
\# html every day
ExpiresByType text/html M300

%
\# What to allow
Options ExecCGI FollowSymlinks Indexes

%
$<$Files "\^(whois.cgi|traceroute.cgi|ping.cgi)\$"$>$
order deny,allow
deny from all
allow from 192.168. 127.0.0.1
$<$/Files$>$

%
$<$Files "\^(alert.cgi|log-event.cgi)\$"$>$
order deny,allow
deny from all
allow from 192.168.20.1 192.168.23.3
$<$/Files$>$

%
\# How they're allowed in
order deny,allow
allow from all
```

I won't claim the IP#-based access-control is completely safe, but it's easy and keeps out casual browsers. If you **really** need to keep this information safe, use a secure web-server, say apache with mod_ssl. If that's not good enough, you ought to consider whether this stuff really belongs on a network at all.

Chapter 9

The Configuration Directory

The run-time configuration of remstats is done through a directory-tree of files. The current tree structure is:

```
%
    configdir
        +--- alerts (see the alerts section)
        +--- alert-destination-map (see the alert-destination-map section)
        +--- alert-template-map (see the alert-template-map section)
        +--- alert-templates (see the alert-templates section)
        |       +--- template1
        |       +--- template2
        |       ...
        +--- archives (see the archives section)
        +--- availability (see the availability section)
        +--- colors (see the colors section)
        +--- customgraphs (see the customgraphs section)
        |       +--- graph1
        |       +--- graph2
        |       ...
        +--- datapages (see the datapages section)
        |       +--- datapage1.page
        |       +--- datapage2.page
        |       ...
        +--- general (see the general section)
        +--- groups (see the groups section)
        +--- hosts (see the hosts section)
        |       +--- host1
        |       +--- host2
        |       ...
```

```
+--- host-templates (see the host-templates section)
|       +--- host-template1
|       +--- host-template2
|       ...
+--- html (see the html section)
+--- links (see the links section)
+--- oids (see the oids section)
+--- remotepings (see the remotepings section)
+--- rrds (see the rrds section)
|       +--- rrd1
|       +--- rrd2
|       ...
+--- scripts (see the scripts section)
|       +--- script1
|       +--- script2
|       ...
+--- times (see the times section)
+--- tools (see the tools section)
+--- views (see the views section)
|       +--- view1
|       +--- view1
|       ...
+--- view-templates (see the view-templates section)
        +--- template1
        +--- template2
        ...
```

(You can look at the base configuration directory if you want, but you should also read through this so you know the significance of what you see.

Almost all the configuration files allow both blank lines and comment-lines. A comment-line **begins** with a # and the whole line is ignored by remstats. Inline comments are **not** permitted as the '#' is used in some places for other purposes. The only files which don't permit comments are the `view-templates`, which are html, and the last part of `datapages`, which are also html.

`alert-templates`, `customgraphs`, `datapages`, `scripts`, `rrds`, `hosts`, `host-templates`, `views` and `view-templates` are sub-directories with the files within describing one of that kind of entity. E.G. a file in the `hosts` sub-directory is named for the host and contains that host's configuration.

NOTE: within the sub-directories, files with names beginning with 'IGNORE-' or ending with '~', will be ignored.

There are also a few tools (see the tools section) to help you make and update your config-file, although not all parts of it.

9.1 Configuration - Alerts

The alerts config-file is used by the alert-monitor (see the alert-monitor section) to decide who to send alerts for problems. The rrds (see the rrds section) and hosts (see the hosts section) config-files decide `when` an alert needs to be raised, and these lines tell `who` gets the alert.

Each line is in seven parts, most of which are patterns, e.g.:

```
%
warn * MISC UPTIME 0 0 uptime-alerts
error   silverlock.dgim.crc.ca  * * 600 900 test-alerts
error   news.crc.ca * * 600 900 news-alerts
critical * * * 600 900 critical-alerts
```

The first "word" is the status, as decided by the alert-monitor (see the alert-monitor section) . The second word is a regex to match against the hostname. The third is a regex to match against the rrdname. The fourth is a regex for the variablename. The fifth is the minimum time for the alert condition to be present before an alert can be triggered. The sixth is the interval after sending an alert before another will be sent. The seventh is an `alert-destination` as specified in the alert-destination-map (see the alert-destination-map section) .

Note: The seventh used to be an alert program and there was an eighth which was an address, of a form appropriate to the alert program. This has been rolled into the `alert-destination-map` to make it more flexible.

If the current condition matches the status, host, rrd, and variable, then alert-monitor has to look at the times. If this is a new condition (i.e. it was in OK status previously), then an alert won't be triggered until after the minimum time has passed. This avoids transient problems being reported. If you want these to be reported, then set it to zero. If this is an old alert, then an alert won't be triggered until the interval time has passed since the previous alert. If the interval is 0 (zero) then there will only be one alert at the start-time.

9.2 Configuration - alert-destination-map

This config-file specifies the mapping from an abstract alert destination, specified in the alerts config-file (see the alerts config-file section) , and the address(es) to send it to. The alerter (see the alerter section) attempts to match the abstract alert destination against each of the `map` lines and sends alerts to any which match.

There are three kinds of lines in this config-file: `map`, `alias`, and `method`. `Map` lines map from an abstract alert destination, listed in the alerts config-file, to a less abstract alias, listed here. The `alias` lines allow a crude list capability and also permit the use of different methods to deliver the alert. `Method` lines tell what program to run with what arguments in order to deliver to that type of address.

9.2.1 Map Lines

A map line looks like:

```
%
map DEST TIME DOW DOM MON ALIAS
```

Where:

- DEST is an abstract alert destination, listed in the alerts config-file
- TIME is a time-of-day specification, comma-separated time-ranges or '*' meaning all times. A range looks like HHMM-HHMM.
- DOW is a day-of-the-week spec, a comma-separated list of weekdays, in numeric form (0=sunday, 1=monday, ...) or '*' for all weekdays.
- DOM is a day-of-the-month spec. It's a comma-separated list of day-ranges, where a range is a day or DD-DD, or '*' ofr all days.
- MON is a month spec. It's a comma-separated list of month-ranges, in numeric form, like MM or MM-MM or '*' for all months.
- ALIAS is the alias that this DESTination maps to during the specified time-period. It's defined in an alias line.

This permits different DESTinations to be sent to different people at different times, depending on who's on duty.

9.2.2 Alias Lines

An alias line looks like:

```
%
alias ALIAS METHOD:ADDR ...
```

Where:

- ALIAS is the alias being defined
- METHOD is an alert-delivery method (see methods below)
- ADDR is an address which is valid for that method

This indirection permits delivery of the same alert via multiple methods, in case one or more of the methods isn't available, as well as to different people.

9.2.3 Method Lines

A method line looks like:

```
%
method METHOD COMMAND-LINE
```

Where:

- METHOD is the method being defined
- COMMAND-LINE is the program to run with any arguments it requires. It will be passed the alert message on stdin and the address to send it to at the end of the COMMAND-LINE.

9.2.4 An Example

We have three guys on different shifts who manage network operations (Tom, Dick and Harry) during the week. On the week-end Frank is on call. We also want to email a copy to an email address which collects all the alerts. We want to send the alerts to whoever is working at the time. Say the abstract destination specified in the alerts config-file (see the alerts config-file section) is `alerts`. We might use lines like those below:

```
%
map alerts 0600-1359 1,2,3,4,5 * * tom
map alerts 1400-2159 1,2,3,4,5 * * dick
map alerts 0000-0559,2200-2359 1,2,3,4,5 * * harry
map alerts * 0,6 * * frank
```

```
%
alias tom email:tom@our.com email:alert-history@our.com winpopup:console
alias dick email:dick@our.com email:alert-history@our.com winpopup:console
alias harry email:harry@our.com email:alert-history@our.com winpopup:console
alias frank page:555-1234 email:alert-history@our.com winpopup:console
```

```
%
method email /home/remstats/bin/alert-email
method winpopup /home/remstats/bin/alert-winpopup
method page /home/remstats/binalert-page
```

Note: the hypothetical `page` method isn't provided with remstats. There are lots of different programs to send pages. Look at alerter (see the alerter section) if you want to add your own methods; it's easy.

9.3 Configuration - alert-template-map

The `alert-template-map` tells which alert-template (see the alert-template section) to use for which addressee or RRD. The lines look like:

```
%
address regex template
```

or

```
%
rrd rrd template
```

or

```
%
rrd rrd:variable template
```

Addresses are checked first. This is to allow special mapping for devices like pagers which can't display a lot of information. If none of the special addresses match, then RRDs are checked, first with variables then without. An RRD can be an RRD instance, like `port-ftp`, or the wild RRD, e.g. `port-*`.

The `template` is the name of the template file in the `alert-templates` config-dir.

9.4 Configuration - alert-templates

The `alert-templates` directory contains the alert message templates. They are just text files with cookies (see the cookies section) in them. The cookies available are slightly different than the standard list, but they work the same way. You put ##COOKIENAME## wherever you want to see the value of the 'cookiename' variable. The ones available for alerts are:

- HOST - the host for the alert
- REALRRD - the RRD instance for the alert
- FIXEDRRD - the RRD instance with the character-set translated a bit for file-names and message-id's
- VAR - the variable name
- STATUS - the alert status (OK, WARN, ERROR, CRITICAL)
- VALUE - the value of the variable that caused this alert
- RELATION and THRESHOLD - the alert is triggered when the VALUE is no longer in RELATION to the THRESHOLD value.

- START - when the alert was first noticed
- DURATION - how long the alert has been in this STATUS
- HOSTDESC - the description line for this host
- RRDDESC - the description for this instance of the RRD
- NOW - the current time as a unix timestamp
- NOWTEXT - the current time for email headers
- ALERTHOST - the hostname of the host sending the alert
- TOWHO - the addressee for this alert

There are three special files in the `alert-templates` directory, which **must** exist:

- DEFAULT - which contains the default template to be used when no other matches the alert-template-map (see the alert-template-map section) .
- HEADERS - which supplies the headers for each message, with the same substitutions as the rest of the template files. Make very sure that the HEADERS file ends with or contains an empty line or your message will be interpreted as part of the headers and will undoubtedly look wrong. The alert-email (see the alert-email section) script does not check this.
- FOOTER - supplies a standard ending for each message.

9.5 Configuration - Archives

The archives file names various data-retention periods. RRDtool—http://ee-staff.ethz.ch/~oetiker/webtools/rrdtool calls them RRAs. Each line is in two pieces: an archive name and an RRA specification, exactly as documented in the rrdcreate manpage. Unfortunately, modifying this after the rrd has been created isn't one of the things that RRDtool does. I've got an rrd munger on my L<to-do list—todo (see the RRDtool—http://ee-staff.ethz.ch/~oetiker/webtools/rrdtool calls them RRAs. Each line is in two pieces: an archive name and an RRA specification, exactly as documented in the rrdcreate manpage. Unfortunately, modifying this after the rrd has been created isn't one of the things that RRDtool does. I've got an rrd munger on my L<to-do list—todo section) , but it's still not done.

For example:

```
%
day-avg         AVERAGE:0.1:1:600
week-avg        AVERAGE:0.1:7:300
month-avg       AVERAGE:0.1:30:300
3month-avg      AVERAGE:0.1:90:300
year-avg        AVERAGE:0.1:365:300
```

```
%
day-min         MIN:0.1:1:600
week-min        MIN:0.1:7:300
month-min       MIN:0.1:30:300
3month-min      MIN:0.1:90:300
year-min        MIN:0.1:365:300
```

9.6 Configuration - Availability

There are two types of availability definitions: for an RRD or for an RRD on a particular host. The RRD may also be a wildcard RRD, like "df-*" or an instance of an RRD, like "df-/home". The definitions look like:

```
%
rrd RRDNAME VARNAME CF RELATION THRESHOLD
and
host HOSTNAME RRDNAME VARNAME CF RELATION THRESHOLD
```

The CF is one of LAST, MIN, MAX or AVERAGE, with rrdtool's usual meaning. The RELATION can be any one of: <, <=, >, >=, or =. The THRESHOLD is the number to which the value of VARNAME must be in the correct RELATION. (Clear as mud.)

As an example, take the following definition:

```
%
rrd ping rcvd MINIMUM $>$ 0
```

This means that the variable 'rcvd' in the ping rrd must be greater than zero for it to be considered "available". All time intervals where it isn't, or for which no data is available, are considered "unavailable".

There are also two other record types: colors and thresholds. A colors record looks like:

```
%
colors COLOR1 ...
```

A thresholds line looks like:

```
%
thresholds NUMBER ...
```

and must have the same number of values as the colors line. Only one of each. Here's an example to make the use clear (I hope):

```
%
colors avail1 avail2 avail3 avail4
thresholds 99 98 95 90
```

The `colors` line above requires that the colors 'avail1', ... be defined in the colors (see the colors section) config-file. The `thresholds` line above specifies that if an availability is 99% or above, it should be colored 'avail1' color, 98% to 99%, use 'avail2' color, etc.

9.7 Configuration - Colors

The `colors` config-file just gives names to colors so you don't have to remember or decipher hex numbers. Each line is: a colour name and a six-digit hex number giving the RGB values. The color "down" is special and is used in the ping index to color the background of hosts which are down.

For example:

```
%
totalcolor      00a000
usedcolor       a00000
downcolor       a07777
```

9.8 Configuration - Customgraphs

The `customgraphs` directory contains files which define graphs which aren't associated with a specific host. They are very like a graph defined on an rrd (see the rrd section) . They do have a times line preceeding the graph definition, to set the time-periods for that graph. They are linked in under the `Custom Index`.

Note that the graph definition itself must be indented, like rrd graphs.

As an example:

`times day yesterday week 3month year`

```
%
        --upper-limit 100 --lower-limit 0 --rigid
        --vertical-label 'CPU \%'
        --title 'CPU Usage (\#\#GRAPHTIME\#\#)'
        DEF:silverlockuser=\#\#DATADIR\#\#/silverlock.dgim.crc.ca/cpu.rrd:user:AVERAGE
        DEF:silverlocksys=\#\#DATADIR\#\#/silverlock.dgim.crc.ca/cpu.rrd:system:AVERAGE
        DEF:loisuser=\#\#DATADIR\#\#/lois.dgim.crc.ca/cpu.rrd:user:AVERAGE
        DEF:loissys=\#\#DATADIR\#\#/lois.dgim.crc.ca/cpu.rrd:system:AVERAGE
        CDEF:silverlock=silverlockuser,silverlocksys,+
        CDEF:lois=loisuser,loissys,+
        'LINE2:silverlock\#\#\#COLOR1\#\#:silverlock'
        'LINE2:lois\#\#\#COLOR2\#\#:lois'
```

9.9 Configuration - General

This is the miscellaneous config-file, but there are some critical pieces here:

- `datadir` (REQUIRED) - The data for a given host is stored under `datadir/hostname`. There are also other status files stored in this directory.
- `staletime` (UNUSED) - How long before we count a status as stale. (seconds)
- `minuptime` - How long a host must be up before it stops being flagged as recently up. (seconds)
- `keepalerts` (UNUSED) - How long to keep records of alerts after the condition no longer exists. (seconds)
- `uptimealert` - If this is set, the alert-monitor (see the alert-monitor section) will cause a warning level condition on the fake rrd MISC for the fake variable UPTIME, for any host whose uptime is less than this value. Whether this will trigger an alert depends on the alerts (see the alerts section) file.
- `pinger` - If defined, this names the ping-collector (see the ping-collector section) to be used before all the other collectors (see the collectors section) . (Unless you write your own, you put `ping-collector` here.) If you don't include this line, then you'll want to make sure the the `ping-collector` is listed in the `collectors` line (below).
- `collectors` - This line tells run-remstats (see the run-remstats section) which collectors to run. The default list is all of them, so you can gain some benefit by pareing this line down to those you are using, but remember it if you add new rrds (see the rrds section) that need other collectors later. **Note:** you list the names of the collectors without the '`-collector`' on the end. E.G. the `ping-collector` would be included as just '`ping`'.
- `monitors` - This line tells run-remstats (see the run-remstats section) which monitors (see the monitors section) to run. The default is all of them.
- `pagemakers` - This tells run-remstats (see the run-remstats section) which pagemakers (see the pagemakers section) to run at the end, if the config-dir has changed. The default is all of them.
- `max-port-patterns` - This tells the port-collector (see the port-collector section) how many parenthesised patterns there can be, at most, in `valuepattern`s or `infopattern`s. The default is 10.
- `watchdogtimer` - This sets the limit that run-remstats (see the run-remstats section) will apply to each of the programs that it runs, so that, e.g., a hanging collector will not hang the whole remstats cycle.
- `keeplogs` - This tells how long cleanup (see the cleanup section) will permit old files to hang around, in seconds.

9.10 Configuration - Groups

If any name has spaces, it must be 'quoted' or "quoted". Any groups not listed here *will not be linked into the HTML index pages generated by html-writer*, but pages will still be created for them. If there is a file in the `htmldir` with the same name as a group-name, but with the spaces replaced by '_' and all the letters lower-case, followed by '.html', then the group-names in html pages will be linked to those pages. E.G. for a group named `"Local Routers"`, if there is a file called `local_routers.html`, the name of the group will be made into a link to that file.

9.11 Configuration - Hosts

The `hosts` files are what the whole configuration has been working toward. Here we tell which hosts we're interested in and what we want to monitor. Here's a sample host file called `clark.dgim.crc.ca`:

```
%
desc    DNS and Web
ip      142.92.39.18
aliases ns1.crc.ca
via 142.92.32.10
group   Servers
contact Thomas Erskine $<$thomas.erskine@crc.ca$>$
tools   ping traceroute telnet http clark-special:special
rrd     ping
rrd     cpu
noalert cpu user
community xyzzy
rrd     load
nograph load users
rrd     if-le0
alert if-le0 ierr $<$ 1000 5000 10000
alert if-le0 in WARN
rrd     df-/var
rrd     df-/tmp
rrd     port-http critical
rrd     port-ssh
rrd     port-whois
noavailability port-whois status
noavailability port-whois response
rrd     port-domain critical
```

The name of the file (`clark.dgim.crc.ca]`) is the host that you're interested in. The name should be a fully-qualified-domain-name, but anything which perl's getaddrbyname can resolve should work.

The `ip` line saves the IP number from having to be looked up and could be used to deal with hosts which aren't in the DNS. If you want the IP number to be looked up each time, you can leave this line out.

The `desc` line gives this host a description graph-writer (see the graph-writer section) will put on pages about this host.

The `alias` line tells remstats about other names for this host. This is mainly for the `ping-collector` to allow it to tell for sure when it has got a response from this host.

The `via` line is used by the topology-monitor (see the topology-monitor section) to specify networking gear (like hubs and switches) which are in the path to the host, but won't show up in a traceroute.

The `group` line is required and tells which group this host belongs to. Remember, you defined all the groups back in the general (see the general section) file?

The `contact` line tells who to contact for this host. If a line in the alerts config-file (see the alerts config-file section) refers to a recipient called `CONTACT`, the value of the host's contact line will be substituted.

The `tools` line tells which tools (defined in the tools config-file (see the tools config-file section)) you want to appear for this host. E.G. if a host doesn't have a web-server, there's no point in providing a link to connect to it. To accomodate host-specific tools, a toolname can be given as `real-tool-name:display-name`. This means that the tool will be defined in the `tools` config-file as `real-tool-name`, but will be displayed as `display-name`.

The rrd (see the rrd section) lines tell which rrds to collect for this host. If the rrd was defined as a wildcard, it will have the instance specified here. In the example there are three wildcard lines, referring to `if-le0`, `df-/var` and `df-/mail`. The first is looking at the data for network interface hme0 and the others are getting data on the /var and /mail file-systems, respectively.

The first `alert` line is setting the alert threshold for `if-le0` to 50. If this host file was from the same configuration as the previous `rrd` sample, the alert here would override the one in the `rrd` file. There is also a `noalert` line, which cancels an alert set in the rrd without setting a replacement alert. The alert line for a host must specify the rrd as well, but is otherwise the same as an alert on an rrd.

The second `alert` line is specifying the status (`WARN`) for missing data for the `in` variable.

There can also be descriptions for rrds. If you append to an `rrd` line something like `desc='xyzzy'`, then you'll see that description on pages dealing with it. I added this for labelling network interfaces, but you can use if for anything you want.

The `community` specifies the SNMP community string to use for this host to fectch SNMP data. If the host config-file doesn't specify any RRDs collected by the snmp-collector, you don't need to specify a community.

If this host uses any rrds collected by the snmp-collector, it can also specify a port to use like:

```
%
snmpport 3401
```

If the RRD itself specifies a port, then the RRD-specified port will be used instead, for that RRD.

If you don't want a particular graph for this host, you can include a `nograph` line. It looks like:

```
%
nograph rrdname graphname
```

There can also be a `statusfile` line, looking like:

```
%
statusfile NNN
```

with `NNN` replaced by the name of a status file from that hosts's data directory. This permits the main index pages to show the status of an un-pingable host as the status of something else, like the reachability of it's web-server (STATUS-port-http).

The `noavailability` lines tell the availability-report (see the availability-report section) program not to report on certain rrd/variable combinations. In this case, we don't want to see availability stats on the whois server. Maybe it's too embarassing?

9.12 Configuration - Host Templates

These config-files simply hold the same kind of lines as a host config-file (see the host config-file section) . By adding a line like:

```
%
template some-host-template
```

to a host config-file, you achieve the same effect as adding all the lines contained within the template file. If you have many hosts which are similar, this can be a usefull way of keeping the configuration consistent.

It can also be used to parameterize things. As an example, if you are using the nt-status-server (see the nt-status-server section) and are only running it on a single host which is providing information on various other NT hosts, you might make a template, say `default-nt-status-server` like:

```
%
nt-status-server my.nt.status.server
```

and replace the `nt-status-server` lines in those hosts with:

```
%
template default-nt-status-server
```

Then if you want to change which machine is running the `nt-status-server`, you'd just have to change the template.

9.13 Configuration - HTML

The html file defines stuff related to web-page generation. There are several different kinds of information.

9.13.1 Locations

These things define where things are, like URLS. They are:

- htmldir (REQUIRED) - The html stuff for a given host is stored under htmldir/hostname.
- htmlurl (REQUIRED) - How to refer to the htmldir in a URL.
- viewdir - Where to store the views, in case you don't want them under the htmldir.
- viewurl - How to refer to the viewdir in a URL.
- webmaster (REQUIRED) - Who's in charge of these web-pages, an email address to get stuffed into mailto URLs.
- logourl - Where is the logo for this site
- homeurl - ehere is home for this site
- topurl - where top goes for this site
- rrdcgi - where to find the rrdcgi program, I like to link it and rrdtool into /usr/local/bin, for ease of use.
- motdfile - where to find the Message-Of-The-Day file. This is used to add in announcements at the top of the index pages, except the host index.

9.13.2 "How-To's"

- thumbnail - How big the graph portion of a thumbnail image is to be (WIDTHxHEIGHT)
- metadata - Where to store CERN-style meta-data, to set expiry times for the gifs. (METADIR METASUFFIX)
- background - what should the background look like. It's mostly obsolete, because you can get the most of the same effects by editing the default.css style file instead.
- htmlrefresh - How often to cause the web pages to refresh themselves. (seconds)
- upstatus, upunstablestatus, downunstablestatus, downstatus, okstatus, warnstatus, errorstatus, criticalstatus - HTML to display for various statuses. The defaults use <span style="xxx"> tags.

- `viewindices` - Should view-writer (see the view-writer section) write the index links at the top of view pages? (yes or no)
- `showinterfaces` - Should graph-writer (see the graph-writer section) show interfaces on a host page? (yes or no)
- `keepimages` - How long cleanup (see the cleanup section) will permit old images to hang around, in seconds.
- `default-tools` - What tools to show for a host which doesn't specify any.

9.13.3 Markers

This group supplies html to wrap various things in the generated web-pages.

- indexprefix, indexsuffix - for the items on the `Indices` line of the header
- groupprefix, groupsuffix - for the group names on the various indices
- hostprefix, hostsuffix - for the host names on the various indices
- toolprefix, toolsuffix - for the tool names on the toolbar
- linkprefix, linksuffix - for the links in the footer
- outofrangeprefix, outofrangesuffix - for the current value on the availability pages when it has gone outside the specified bounds. (See availability-report (see the availability-report section) .)

9.13.4 Labels

If you translate the labels, the web-pages should be translated. It doesn't include error-messages or debugging messages.

The currently available ones, with their defaults, are:

```
%
alertreport Alert Report
comment Comment
contact Contact
customindex Custom Index
description Description
groupindex Group Index
hardware Hardware
hostindex Host Index
indices Indices
ipnumber IP \#
lastupdateon This page last updated on
links Links
logreport Log Report
```

```
operatingsystem Operating System
overallindex Overall Index
pingindex Ping Index
quickindex Quick Index
status Status
tools Tools
uptime Uptime
viewindex View Index
```

And also the:

- `uptimeflag` - shows on some index pages when a host has been up for less than `mintime` (defined in the general (see the general section) file.)
- `alertflagwarn`, `alertflagerror` and `alertflagcritical` - give HTML to be inserted in the quick index for hosts which have alerts active.

9.14 Configuration - Links

The links config-file supplies links that you want to put with the standard links at the bottom of the web pages. They're in two pieces: the text to be shown and the URL to link to.

An example:

```
%
SourceWorks http://www.sourceworks.com/
```

9.15 Configuration - OIDs

[These are for SNMP and you can ignore this config-file if you're not interested.]

The SNMP implementation in the snmp-collector is primitive and only knows about OIDs (Object IDs) by their number. Since I'm not interested in bringing in a full MIB compiler to deal with the MIBs, this section lets you specify names for the OID numbers you're interested in using later. The lines look like:

```
%
CiscoCpuLoad    1.3.6.1.4.1.9.2.1.58.0
```

for a non-hypothetical example, if you happen to have Cisco routers. If you have the ucd-snmp package, their snmptranslate program comes in handy for pulling out the appropriate numbers without the bother of tracking through the wretched MIBs.

9.16 Configuration - remotepings

The `remotepings` file simply lists all the machines which are running the remoteping-server. Or at least all the machines that you want to query.

9.17 Configuration - RRDs

These files are the most complicated. Here's an example, again taken from the `if-` rrd supplied with remstats.

```
%
source          unix-status
step            300
data            in=interface\_packets\_in:* COUNTER:600:0:U
data            ierr=interface\_errors\_in:* COUNTER:600:0:U
data            out=interface\_packets\_out:* COUNTER:600:0:U
data            oerr=interface\_errors\_out:* COUNTER:600:0:U
data            coll=interface\_collisions:* COUNTER:600:0:U
alert           in $<$ 100
alert           out $<$ 100
alert in nodata WARN
archives        day-avg week-avg month-avg 3month-avg year-avg
times           day yesterday week 3month year
graph           if-* desc='Interface data for \#\#RRD\#\#'
--title 'Interface \#\#RRD\#\# \#\#GRAPHTIME\#\#'
--lower-limit 0
--vertical-label 'packets'
DEF:in=\#\#DB\#\#:in:AVERAGE
DEF:out=\#\#DB\#\#:out:AVERAGE
DEF:ierr=\#\#DB\#\#:ierr:AVERAGE
DEF:oerr=\#\#DB\#\#:oerr:AVERAGE
'LINE1:in\#\#\#COLOR1\#\#:Input Packets'
'LINE1:out\#\#\#COLOR2\#\#:Output Packets'
'LINE1:ierr\#\#\#COLOR3\#\#:Input Errors'
'LINE1:oerr\#\#\#COLOR4\#\#:Output Errors'
```

This example shows most things that can be done, except multiple graphs on the same rrd, which is as simple as adding another graph line and its definition.

First, the rrd name is special, in this case. Any rrd file which ends in a '-' is assumed to be for a wildcard rrd, in this case `if-*`. This avoids problems with file-systems which are overly fussy about which characters can be in file-names.

This rrd definition will match any rrd beginning with 'if-' specified in a host config-file. Wildcard rrds are necessary when a given host may have more than one of whatever the rrd is referring to, in this case network interfaces. The network interface name will replace the '*' in the rrd line in the host config-file. It will also be available in the `##WILDPART##` magic cookie (see the magic cookie section) .

The `source unix-status` means that this RRD gets its data from the unix-status-collector (see the unix-status-collector section) .

The `step` line sets the step value for the rrd. This is the expected frequency of data updates. (See the manpage for rrdcreate.) N.B. Setting this is required, but changing some RRDs won't change how often the collectors run. If you have significant numbers which require different update periods, you've got a choice. If it's not very "expensive" to do those queries every time, then just ignore any complaints from run-remstats about updates failing. Otherwise it gets messy. You've got to set up three separate config-dirs. One for one time period, and one for the other running out of cron at appropriate time-periods only running collectors, and a separate one to run the monitors and pagemakers.

(FIXME - the writing stinks)

The `data` lines define various DS elements for this RRD. [See the manpage for rrdcreate.] The first part is the DS name, with an extension. The collectors produce long names and may have instance-names added to the variable name, in this case to tell which interface this data is for. So the first part looks like `dsname=variable:instance`. The `dsname` is used for the RRD DS name and the `variable:instance` part is used to tell updater which collector information applies to this DS. The rest of the line is straight from rrdcreate's description of DS.

It's also possible to invoke configuration-supplied private functions (see the private functions section) on the incoming raw data. The `data` line would look like:

```
%
data xyzzy=\&function(variablename) ...
```

It's your responisbility to make sure that `function` is available and that it works.

The `alert` lines are setting the thresholds for alerts, in this case for the variables `in` and `out`. They must specify, in order: the variable-name, the relation (<, =, >, delta< and delta>) and a space-separated list of thresholds. Since these ones only provide one number each, they can only have OK or WARN statuses. If the variables `in` or `out` have values less than (<) 100, they are considered to be OK. Otherwise they're elevated to WARN status. What will happen when they go into WARN status depends on the alerts (see the alerts section) file. These alerts will apply to any host which uses this rrd, unless it overrides it.

The last alert specifies that missing data for the variable `in` will be considered to be status `WARN`, for purposes of generating alerts. The full description of the alerts is kept in te docs for alert-monitor (see the alert-monitor section) as it is the program which implements them.

The `archives` line tells how to keep the data for this rrd, using the names defined in the archives (see the archives section) file.

There can be multiple `graph` lines describing as many graphs from the data in this rrd as you want. The graph-name must be wildcarded if the rrd is. A `graph` line is followed by its definition which must be indented. The definition is straight from rrdgraph with the magic cookie (see the magic cookie section) substitution. If you want a description , you can add:

```
%
```

```
desc='whatever you want'
```

or

```
%
desc="whatever you want"
```

to the `graph` line. This is used to set the alt text on the web-page.

9.17.1 Collector-specific Stuff

An rrd collected by the port-collector (see the port-collector section) may specify that this particular service is critical, by simply including the word "critical" at the end of line. This will cause the status to be elevated to CRITICAL status if the status ever reaches ERROR level.

An rrd collected by the log-collector (see the log-collector section) will have extra stuff on each data line after the DS information. The extra stuff will be the function and pattern needed by log-collector to pass to the log-server (see the log-server section) to get that variable's data.

An RRD collected by the snmp-collector (see the snmp-collector section) needs to specify which OIDs to fetch. They are specified by name in the RRD with a line like:

```
%
oid APCUpsAdvInputLineVoltage
```

which refers to a name defined earlier in the oids (see the oids section) file.

An RRD collected by the snmp-collector may also specify an SNMP port to use with a line like:

```
%
port 3401
```

9.18 Configuration - Scripts

The `script XXX` files are describing how to query a given port for its status and are used by the port-collector (see the port-collector section) . They look like:

```
%
send GET / HTTP/1$\backslash$.0$\backslash$n$\backslash$n
timeout 5
port 80
infopattern \^Server:$\backslash$s+(.*)\$
valuepattern \^Content-length:$\backslash$s*($\backslash$d+)
ok \^HTTP/$\backslash$d$\backslash$.$\backslash$d 200
warn \^HTTP/$\backslash$d$\backslash$.$\backslash$d [45]$\backslash$d$\backslash$d
```

This example is taken from the supplied `config-base` and queries an HTTP server for its root page. First, it sends the "send" text, which in this case is a minimal HTTP request, and waits no more than 5 seconds. After the port is closed from the remote end, or the timeout expires, any text which was returned is examined by the various tests. In this case, if the web-server sends back a line beginning something like "HTTP/1.1 200", the port will be marked as "OK". Similarly, there are "warn", "error" and "critical" statuses possible.

The `port` is optional and `getservbyname` will be called on the script name, if port isn't specified. This also lets you have multiple scripts for the same port, using different names for the script.

The `infopattern` is optional, and supplies a pattern which will be matched against each line in the result. If there is a match, files will be created in the data directory for that host called `INFOn-rrdname`, where `n` will be in the range 1..9 and `rrdname` will be the name of this rrd, converted to a file-name. The files will contain matches for parenthesised items in the regular expression. E.G. in the example above, a file will be created called `INFO1-http` which will contain whatever the web-server said its type and version was.

Similarly, the `valuepattern` is also optional, but the matches will be returned as collected items called `value1` through `value9`. In the example, this would cause the collector to return a line like:

```
%
hostname timestamp value1 1022
```

An RRD definition could use this by including a line like:

```
%
data pagesize=value1 GAUGE:600:0:10000
```

For a working example, look at the RRD definition for weathernetwork.

9.19 Configuration - Times

The `times` file specifies time intervals for which graphs will be made. I suppose it should be renamed graphtimes or something, but I've got other things to do. Each line is in three pieces: a time name, a start time and and end time. The times are relative to the current time and so will always be non-positive.

The currently defined times are:

```
%
thumb           -60*60*2 0
day             -60*60*24 0
week            -60*60*24*7 0
month           -60*60*24*30 0
3month          -60*60*24*30*3 0
year            -60*60*24*365 0
yesterday       -60*60*24*2 -60*60*24
lastweek        -60*60*24*7*2 -60*60*24*7
```

9.20 Note:

The times `thumb` and `day` are special. The graph-writer (see the graph-writer section) expects them to exist and to have certain meanings. The `thumb` time is a short interval which is used to make the ping thumbnail graphs for the ping index. The `day` time is the default time interval. The higher-level pages will use the `day` graph as a link to the other time intervals.

9.21 Configuration - Tools

The `tools` file is only used by graph-writer (see the graph-writer section) to create toolbars. Each line is in two pieces: a tool-name and a URL to link to for this tool.

The URL can have magic cookies (see the magic cookies section) in it to substitute in things like hostname. Currently, the only cookies which will get substituted here are `HOST`, `IP` and `HTMLURL`. If you think of other usefull ones, please tell me (see the tell me section) .

9.22 Views - your own selection of graphs on one page

The views config-dir contains files, one per view, describing a collection of things that you want to see on one page. There are three kinds:

- **simple** - you specify which graphs or customgraphs you want, using lines like:

  ```
  %
  graph        hostname rrdname graphname
  customgraph  customgraphname
  ```

  You can have as many lines as you want, and can mix graphs and customgraphs. The order you list them in is the order they will appear in the resultant page.

- **template** - you specify a view template to use to generate the page. See the docs on view template (see the view template section) for explanations.

- **datapage** - you specify a datapage to use to generate the page. See the docs on datapage.cgi (see the datapage.cgi section) for explanations.

You can also specify a `description` line, like:

```
%
desc This is what I'm taking about
```

Where the pages are generated is dependant on the `viewdir` and `viewurl` directives in the html config-file (see the html config-file section) . The view pages may have the usual indices on them, if the html config-file (see the html config-file section) includes:

```
%
viewindices yes
```

but by default leave them off.

9.23 View Templates

View templates give you complete control over page layout in a view. They are complete HTML pages with embedded magic cookies, which are substituted for during view generation (by view-writer (see the view-writer section)). The resulting page will be an `rrdcgi` CGI script. The magic cookies are:

- <VIEW::GRAPH hostname rrdname graphname [graphtime]> - This inserts a graph definition for `rrdcgi`. The graphtime is from the times config-file (see the times config-file section) , and is optional.
- <VIEW::CUSTOMGRAPH customgraphname [graphtime]> - This inserts a customgraph definition for `rrdcgi`. The graphtime is from the times config-file (see the times config-file section) , and is optional.
- <VIEW::INCLUDE filename> - This inserts the contents of the named file when the view-page is displayed. (Using the `rrdcgi` cookie <RRD::INCLUDE filename>.)
- <VIEW::HEADER title here> - Inserts a standard remstats header.
- <VIEW::FOOTER> - Inserts a standard remstats footer.
- <VIEW::STATUS host status-file> - inserts the contents of the named status-file when the view-page is displayed. (Using the `rrdcgi` cookie <RRD::INCLUDE filename>.)

You can also include `rrdcgi` magic cookies.

Chapter 10

Configuration Tools

These tools are intended to help you build the hosts part of your configuration file. They take a file (or files) of hostnames and emit host config-files for them. There are currently no config generators for the log-collector, the remoteping-collector or the unix-status-collector.

- split-config (see the split-config section) - converts old config-files to new config-dirs (see the config-dirs section)
- new-config (see the new-config section) - makes a new config-dir populated by symlinks to config-base
- new-ping-hosts (see the new-ping-hosts section) - adds a hosts with a ping rrd
- new-port-hosts (see the new-port-hosts section) - adds hosts which are collected by the port-collector (see the port-collector section)
- new-snmp-hosts (see the new-snmp-hosts section) - adds hosts which are collected by the snmp-collector (see the snmp-collector section)
- new-unix-hosts (see the new-unix-hosts section) - adds hosts which are running the unix-status-server (see the unix-status-server section)
- nt-discover (see the nt-discover section) - finds and adds Windows NT hosts
- snmp-showif (see the snmp-showif section) - shows interfaces from SNMP
- snmp-get (see the snmp-get section) - for testing if you can get a particular OID

10.1 split-config - convert a config-file to a config-dir

10.1.1 Usage:

```
%
split-config version 1.4
usage: ../split-config [options] configfile configdir
where options are:
-d enable debugging output
-h show this help
```

10.1.2 Description:

This is the conversion program to convert an old-style config-file (everything in one file) into the new-style config-dir (a directory containing files and subdirectories. For the layout of the new config-dir, see the <docs—configuration>.

It also deals with the other configuration changes which came with version 0.12.1:

- the [html (see the html section)] group is now documented and `split-config` will create it in the likely case that it's missing.
- All the web-page creation stuff that was in the [general (see the general section)] section has been moved to the html config-file (see the html config-file section) .
- The [groups] section has been separated out into the groups (see the groups section) file.

10.2 new-config - make a new config-dir

10.2.1 Usage:

```
%
new-config version 1.9
usage: new-config [options] new-config-dir
where options are:
    -d ddd  set debug level to 'ddd'
    -f fff  use 'fff' as config-base [/home/remstats/etc/config-base]
    -h      show this text
```

10.2.2 Description:

`new-config` makes a new config-dir and populates it with symlinks to `/home/remstats/etc/config-base`. It makes the `scripts` and `rrds` subdirectories as symlinks too. It makes the `customgraphs` and `hosts` subdirectories as real directories. If you disagree with which ones it chooses as symlinks, look at the top of the program at @links and @subdirs.

You are likely to be modifying the following files, so they are copied instead of being links:

```
%
alerts alert-destination-map general html links tools
```

10.3 new-ping-hosts - add ping RRD to host definition

10.3.1 Usage:

```
%
new-ping-hosts version 1.9
usage: new-ping-hosts [options] group [hostfile ...]
where options are:
    -d nnn  enable debugging output at level 'nnn'
    -f fff  use 'fff' as config-dir [/home/remstats/etc/config]
    -h      show this help
```

10.3.2 Description:

You supply a list of files containing hostnames, one per line. If there is no hostfile supplied, then it will read from stdin. If there is no host config-file for that host, `make-ping-hosts` will write entries like:

```
%
\# hosts/www.example.com
desc gggg host
group gggg
ip 123.456.789.123
tools ping traceroute
rrd ping
```

for that host. If that host already exists, it will simply add:

```
%
rrd   ping
```

to the end of the hosts file. It doesn't check if the host already has a ping rrd.

10.4 new-port-hosts - add RRDs for services

10.4.1 Usage:

```
%
new-port-hosts version 1.9
usage: new-port-hosts [options] group [hostfile ...]
where options are:
    -d      enable debugging output
    -f fff  use 'fff' for config-dir [/home/remstats/etc/config]
    -h      show this help
```

10.4.2 Description:

You supply a file or files of hostnames, one per line, or let it read from stdin.

You can use this to add RRDs to a host describing the various services running on a server, or at least the ones that the port-collector (see the port-collector section) knows how to talk to. It's actually a very limited port-scanner, and will attempt to connect to each of the services. For each one that answers, `new-port-hosts` will write an entry for the corresponding rrd.

If the host has no file, `new-port-hosts` will add one with appropriate header info and a ping rrd. Otherwise, it will just add the port-based rrd's to the end of the host file.

10.5 new-snmp-hosts - add RRDs collected by snmp-collector

10.5.1 Usage:

```
%
new-snmp-hosts version 1.12
usage: new-snmp-hosts [options] group community-string [hostfile ...]
where options are:
    -d      enable debugging output
    -f fff  use 'fff' for config-dir [/home/remstats/etc/config]
    -h      show this help
```

10.5.2 Description:

if you don't supply a file of hostnames, then it will read from stdin.

You can use `new-snmp-hosts` to add RRDs collected by the snmp-collector. It works by looking for a single OID for each RRD. If the OID exists on a given host (i.e. it returns data), then the RRD which uses that data is added to the host. There is currently no way to configure it except for modifying the code. Complain if you'd actually use such a thing.

It's up to you to discard any you don't want.

If there is no hosts file, it will create one with default header info. If there is one, it will just append the rrds.

10.6 new-unix-hosts - add rrds collected by the unix-status-collector

10.6.1 Usage:

```
%
new-unix-hosts version 1.6
usage: new-unix-hosts [options] group [hostfile ...]
where options are:
    -d      enable debugging output
    -f fff  use 'fff' for config-dir [/home/remstats/etc/config]
    -h      show this help
    -p ppp  use port 'ppp' [1957]
    -t ttt  use 'ttt' for timeout [10]
```

10.6.2 Description:

If you don't supply a file of hostnames, then it will read from stdin.

This will add all the remstats distributed rrds collected by the unix-status-collector (see the unix-status-collector section) , except for those using the `PS` section of the collector.

It's up to you to discard any you don't want.

If there is no existing host file for a given host, it will create one with default header info. If there is one, it will just append the new rrds.

10.7 nt-discover - find and add new NT hosts

10.7.1 Usage:

```
%
nt-discover version 1.4
usage: ../nt-discover [options]
where options are:
    -d nnn  enable debugging output at level 'nnn'
    -f fff  use 'fff' for the config-dir [/home/remstats/etc/config]
    -h      show this help
    -s      update status files even if host data-dir found
    -t ttt  use 'ttt' as timeout (in seconds) [10]
```

10.7.2 Description:

Note: `Nt-discover` is not called `new-nt-hosts` because it is a different kind of program. Instead of you providing a list of hosts to add, it finds them itself.

Using information supplied in the discovery config-file (see the discovery config-file section) , `nt-discover` will contact a host running the nt-status-server (see the nt-status-server section) , and run three separate queries:

- `NET-VIEW` will give a list of hosts to check
- `USRSTAT` will give a list of NT users (currently unused, but may be interesting) -item `SRVINFO` (for each of the hosts found in the first step) will give some more details on each host and write new host config-files for each new one. It will not update an existing host config-file.

10.8 snmp-showif - display interfaces from SNMP

10.8.1 Usage:

```
%
snmp-showif version 1.5
usage: ../snmp-showif [options] host community
where options are:
-d enable debugging output
-h show this help
```

10.8.2 Description:

This utility can be used to get a list of interfaces on an SNMP queryable host. You can use it to figure out which interfaces you want to add, and what names remstats uses for them.

It will show ifIndex, ifDescr, ifSpeed, ifType, ifName, ifInOctets, ifOutOctets, ifOperStatus and ifAlias (if it exists).

10.9 snmpif-description-updater

10.9.1 Usage:

```
%
snmpif-description-updater version 1.5
usage: snmpif-description-updater [options]
where options are:
-d ddd  set debugging level to 'ddd'
-f fff  use 'fff' for config-dir [/home/remstats/etc/config]
-h      show this help message
```

10.9.2 Description:

This script updates host config-files, for those which contain `snmpif-*` RRDs, by fetching the ifAlias OID via SNMP and re-writing the configuration file if any of the descriptions have changed. I'd suggest running it every now and then, say once a day, unless you're making very frequent changes to the descriptions.

If somebody has gear which uses an OID other than ifAlias to store the description, then I'll have to consider making this more general, but it'll do for now.

Chapter 11

Servers

11.1 Servers

There are a four servers currently:

- unix-status-server (see the unix-status-server section) is queried by the unix-status-collector (see the unix-status-collector section) for various information it obtains by running various commonly-available programs: `df`, `vmstat`, `uptime`, `netstat`, `uname`, `ps` ...
- log-server (see the log-server section) (queried by the log-collector (see the log-collector section)) reads the unread portion of the specified log-file and returns the requested statistics.
- remoteping-server (see the remoteping-server section) is contacted by the remoteping-collector (see the remoteping-collector section) which supplies a list of hosts. The server runs multiping (see the multiping section) against them and returns results for each, similar to the results obtained from the ping-collector (see the ping-collector section) .
- nt-status-server (see the nt-status-server section) - provides access to information from Windows NT workstations and servers.

11.2 log-server - providing remote access to log information

11.2.1 Usage:

```
%
log-server version 1.8
usage: ../log-server [options] logfile ...
where options are:
```

```
-d nnn enable debugging output at level 'nnn'
-p ppp set the prefix for context-files to 'ppp' [log-server-]
-h show this help
```

The log-server must be supplied with at least one log-file to serve.

11.2.2 Description:

The log-server is queried by the log-collector (see the log-collector section) using a "protocol" described in the log-collector documentation. It will provide information from any of the log-files on it's command-line, but no others. It is recommended that you use the tcp_wrappers or some other form of access-control to limit access to this server. The information may or may not be sensitive, according to which log-files you are serving, but letting anyone query it will mean that you will lose some data, unless you're sure that they will only query it in test mode.

The log-server will store context for each log-file that is served, by default in `/var/tmp/log-server-XXX`, where `XXX` is replaced by a munged version of the log-file name. If you want this stored somewhere else, use the `-p` switch or change the program.

11.2.3 Notes:

Don't forget to list all the log-files that you want to serve on the command-line. If there are too many for your inetd, make a tiny shell script with the `log-server` invocation and run that from inetd.

For details on installation, you'd better look at the server installation docs (see the server installation docs section) .

11.3 nt-status-server - allow remote gathering of Windows NT data

11.3.1 Usage:

```
%
usage: nt-status-server [options]
where options are:
    -a      show all available performance counters
    -d ddd  enable debugging at level 'ddd' [\$main::debug]
    -h      show this help message
    -i [ppp sss] install this as an NT service, using 'ppp' as
    as perl and 'sss' as this script.  Defaults to
perl=C:$\backslash$Perl$\backslash$bin$\backslash$perl.exe
script=wherever-it's-invoked-from
    -p ppp  run server on port 'ppp' [1957]
```

```
    -s      run stand-alone, i.e. not as a service
    -u      un-install this service
N.B.: Just running this script will cause it to run as a service,
and when it stops, it will properly stop as a service.
```

11.3.2 Description:

The nt-status-server allows the nt-status-collector (see the nt-status-collector section) to get data from a remote machine running some flavour of NT and possibly Windows 2000. It runs `SRVINFO` from the NT Resource Kit, to find the version of NT and examines the NT performance counters for other information.

For details on installation, look at the server installation docs (see the server installation docs section) .

11.3.3 Protocol:

The nt-status-collector (see the nt-status-collector section) connects to the `nt-status-server` and sends a series of commands, ending with 'GO'. Then the server sends back the data it obtained and closes the connection. The commands are often the names of programs to run (in UPPERCASE) and the ones known currently are:

- SRVINFO - runs SRVINFO and returns the version of NT
- PERFCOUNTERS - examines the NT performance counters and returns information about memory, disk, processes, ...
- PULIST - runs PULIST (from the NT ResKit) and shows counts for all the running processes.
- MSDRPT - runs WINMSDP to show (currently) memory total and free.
- USRSTAT - runs USRSTAT (from the NT ResKit) and shows when the various users in the specified NT domain last logged in, and which domain-controller authorized them.
- NET-VIEW - runs "NET VIEW" to list the computers currently visible.
- TIME - compares local and remote times

If you want to see what it returns, you can simply start it up and telnet to it.

11.3.4 Installation

You'll have to use SRVANY to run it as a service until I figure out why the service code doesn't work. Note that I've had to run the service under the local system account to get it to be able to access most interesting info.

11.3.5 Bugs

- It is intended that it will eventually install itself as an NT service, and most of the code is there, but it doesn't currently work. Patches gratefully accepted. For now you have to invoke it with the `-s` switch to have it run stand-alone or use SRVANY to provide the NT service stuff, (which also requires the `-s` flag.
- Not only is is currently single-threaded (i.e. won't accept more than one connection at a time), but if a second connection comes in the server won't answer any more requests and will have to be re-started. I've added code to nt-status-collector (see the nt-status-collector section) so that it won't run if there is another instance running already. This won't help if you're using telnet to test the nt-status-server, so be prepared to restart nt-status-server if it gets wedged because of this.

11.4 remoteping-server - allow remote collection of ping data

11.4.1 Usage:

```
%
remoteping-server version 1.5
usage: ../remoteping-server [options]
where options are:
-d nnn enable debugging output at level 'nnn'
-h show this help
```

11.4.2 Description:

The remoteping-server allows the remoteping-collector (see the remoteping-collector section) to obtain ping data remotely. Like the ping-collector (see the ping-collector section) , it uses multiping (see the multiping section) to get ping data, but it can be queried from a remote site.

I'm looking for volunteers; please look at this note (see the this note section) .

11.5 unix-status-server - allow remote gathering of unix data

11.5.1 Usage:

```
%
unix-status-server version 1.26
usage: ../unix-status-server [options]
```

```
where options are:
-d nnn enable debugging output at level 'nnn'
-h show this help
-r include remotely-mounted file-systems
-t tst do tests 'tst, a comma-separated list of:
vmstat, df, uptime, netstat, uname, ps, proc,
ftpcount, netstat-tcpstates, fileage and qmailq
```

11.5.2 Description:

The unix-status-server allows the unix-status-collector (see the unix-status-collector section) to get data from a remote machine running some flavour of unix. It runs several different programs on request (uname, vmstat, df, uptime, netstat, ps, ftpcount, qmail-qstat, and qmail-qread).

11.5.3 Protocol:

The unix-status-collector (see the unix-status-collector section) connects to the `unix-status-server` and sends a series of commands, ending with 'GO'. Then the server sends back the data it obtained by running the requested programs and closes the connection. The commands are usually the names of programs to run (in UPPERCASE) and the ones known currently are:

- UNAME runs uname and returns: machine, os_name, os_release, os_version
- VMSTAT runs vmstat and returns variables relating to memory usage depending on the operating system
- DF runs df and for each file-system returns: dfsize:FSNAME, dfused:FSNAME, dfpercent:FSNAME inodessize:FSNAME, inodesused:FSNAME, inodespercent:FSNAME
- UPTIME runs uptime and returns: uptime (a timestamp in seconds), users, load1, load5, load15
- NETSTAT runs netstat and, for each interface, returns: interface_packets_in:IFNAME, interface_errors_in:IFNAME, interface_packets_out:IFNAME, interface_errors_out:IFNAME, interface_collisions:IFNAME
- PS runs ps and returns various numbers pulled out of the output (see below)
- FTPCOUNT runs ftpcount (from wuftpd) to find out which groups the ftp-server's users fall into
- QMAILQ runs qmail-qstat and qmail-qread and returns: qmail_qsize, qmail_qbacklog, qmail_qlocal, qmail_qsite, qmail_qremote
- FILEAGE returns timestamps for the ages of specified files (see below)

- TIME return the difference in time-stamps between the host running the unix-status-server and the querying host. It must be given the querying host's timestamp following the `TIME` directive. The two variables returned are time and timediff.

If you want to see what it returns, you can simply invoke the `unix-status-server` as a local script and type commands at it.

11.5.4 Programs:

- `/usr/local/bin/uname` or `/usr/bin/uname` It's cosmetic, for web-page header info, but sometimes it's really usefull too.
- `/usr/bin/vmstat` or `/usr/ucb/vmstat` for scanrate, interrupts, context-switches and cpu-time, and of course free memory and swap
- `/usr/local/bin/df` or `/usr/xpg4/bin/df` or `/bin/df` the gnu df. I need the -P flag (for Posix, but it makes df put its info on one line), and the -i flag for inode info.
- `/usr/local/bin/uptime` or `/usr/bin/uptime` or `/usr/ucb/uptime` the gnu uptime has a format I know how to parse. Others sometimes invent new ways to be cute about the display that I don't always recognise.
- `/usr/bin/netstat` or `/usr/ucb/netstat` to get network interface info
- `/usr/bin/ps` or `/bin/ps` for counting the number of running instances of a named process.
- `/usr/local/bin/ftpcount` (part of wu-ftpd distribution) shows the number of ftp clients of wu-ftpd from each access group.
- `/var/qmail/bin/qmail-qstat` and `/var/qmail/bin/qmail-qread` If you have qmail (see **http://www.qmail.org/**), these will let you get information about the queue size, which you can't find from the logs. Otherwise, ignore them.

11.5.5 PS Usage:

With extended commands, of which `PS` is the first, you also specify what you want to look for with extra commands, in addition to the `PS` command. A command looks like:

```
%
varname PS func pattern
```

The `varname` is used to create a variable-name for the returned data. The name will be `ps:varname`. `Func` is one of `count`, `sum`, `last`, `average`, `min`, `max`.

`Pattern` is a perl-style regular-expression, the simplest form of which is just a string.

For an example, if we wanted to know how many web-servers were running over time, we might use (very sloppily):

```
%
webservers PS count httpd
```

[You probably want a better regular expression.]

11.5.6 FILEAGE Usage:

For the `FILEAGE` command, you have to specify an extended command that looks like:

```
%
varname FILEAGE /path/to/file
```

This will produce a timestamp for the last-modification time of `/path/to/file`.

11.5.7 Notes:

With older versions of `vmstat` (ones that mash fields together), it will give up on `vmstat` and not return memory and CPU info. It also requires a version of `df` that will accept the `-P` and `-i` flags. The `-P` flag forces the output for a file-system to stay on one line (easier for me to parse) and the `-i` returns info about inodes. If the `-i` flag is missing, you won't get any inode data. You also won't get any inode data if the file-system doesn't have inodes. (Duh :-).

For details on installation, look at the server installation docs (see the server installation docs section) .

Chapter 12

Collectors

12.1 Collectors Data Format

All the collectors produce data on stdout in the same standard form:

```
%
host timestamp variable value
```

If the `variable` is for something like network interfaces, where the host can have several of them, the data will look like:

```
%
host timestamp variable:instance value
```

Having all the collectors using a standard form permits a single updating program, updater (see the updater section) , to process the data from them all, and also means that I can write a new collector without needing to change the updater.

Chapter 13

Collectors

13.1 Remstats supplied collectors

- log-collector (see the log-collector section) - gets info from remote log-files
- ping-collector (see the ping-collector section) - pings hosts
- port-collector (see the port-collector section) - checks on remote services
- remoteping-collector (see the remoteping-collector section) - pings hosts from somewhere else
- unix-status-collector (see the unix-status-collector section) - gets info from unix hosts
- snmp-collector (see the snmp-collector section) - gets info via SNMP
- snmp-route-collector (see the snmp-route-collector section) - counts routes available from BGP peers

The usual invocation of a collector (via run-remstats (see the run-remstats section)) is:

```
%
xxx-collector | updater xxx
```

Chapter 14

Collectors

14.1 How to write your own collector

There are a few requirements:

1. it must write its results to stdout in standard form (see the top of this file.)

2. it must be placed in the same directory with the rest of the collectors, and must be called XXX-collector, replacing "XXX" with whatever it's collecting.

3. it must take (or at least ignore), the same arguments that the other collectors do, specificly, the `-f`, `-u` and `-F` flags, and the `-G` and `-H` flags, if I ever get around to implementing them (see todo (see the todo section)).

4. you must add it to the list of collectors (the `collectors` line in the general config-file (see the general config-file section)).

5. you must define rrd(s) specifying "source XXX" to use the data from this collector.

6. you must add "rrd YYY" to the appropriate host config-files (see the host config-files section) .

There is a supplied `skeleton-collector.pl` file supplied with the distribution, which does everything except collect data. You should be able to plug your code into its `collect_host` routine and have a collector, if you don't mind writing in perl.

14.2 log-collector - get stats from remote log-files

14.2.1 Usage:

```
%
log-collector version 1.12
usage: ../log-collector [options]
where options are:
   -d nnn  enable debugging output at level 'nnn'
   -f fff  use config-dir 'fff'[/home/remstats/etc/config]
   -F      force collection, even if it's not time
   -h      show this help
   -H HHH  only do hosts from 'HHH', a comma-separated list
   -p ppp  contact log-server on port 'ppp' [1958]
   -t ttt  timeout each port attempt after 'ttt' seconds [10]
   -u      ignore uphosts file
```

14.2.2 Description:

The log-collector gets data from remote log-server (see the log-server section) 's. This way the whole log-file doesn't have to be transfered. The "protocol", if it deserves that name, is very simple. The collector sends a request, which looks like (you can type it in via telnet):

```
%
LOGFILE /wherever/the/logfile.is
varname function pattern
...
GO
```

The directives are all in **UPPERCASE**. They are `LOGFILE`, `GO`, `DEBUG` and `TEST`. The `LOGFILE` directive tells the `log-server` which file to read. The `GO` directive starts the request. `DEBUG` causes some extra remote debugging output, and `TEST` makes the `log-server` operate in test mode. In test mode it doesn't update the last-read position for that log-file, so you won't lose any data when testing.

The other lines are telling the `log-server` what data to collect. The first "word" is the variable name to be returned. The next is the function to be applied (from `count`, `sum`, `average`, `min`, `max`, `first` and `last`). The rest of the line is a perl-style regex. Except for the count function, the regex must contain a (parenthesized) number, to which the function will be applied.

For example, the line:

```
%
rootlogins count ROOT LOGIN
```

would return data for a variable called rootlogins. The value would be the count of the records in the specified logfile which had the string 'ROOT LOGIN' in them.

The pattern can be much more complicated, for example (from the httpdlog rrd):

```
%
bytes sum $\backslash$sHTTP/$\backslash$d$\backslash$.$\backslash$d"$\backslash$s+2$\backslash$d$\backslash$d$\backslash$s+($\backslash$d+)
```

This looks through a standard web-server log-file and extracts the bytes transferred and adds them up to produce the total number of bytes transferred in that sample period.

14.2.3 How to make RRDs that use the log-collector

It's easiest to explain by example. Look at the beginning of the rrd `httpdlog`, copied here:

```
%
source  log
step    300
data    requests        GAUGE:600:0:U   count (GET|POST)
data    success         GAUGE:600:0:U   count $\backslash$sHTTP/$\backslash$d$\backslash$.$\backslash$d"$\backslash$s+2$\backslash$d$\backslash$d
data    bytes           GAUGE:600:0:U   sum $\backslash$sHTTP/$\backslash$d$\backslash$.$\backslash$d"$\backslash$s+2$\backslash$d$\backslash$d$\backslash$s+($\backslash$d+)
```

To form the requests to be sent to the log-server, the log-collector takes the DS name, e.g. `success`, and the last part of the line after all the DS definition `count \sHTTP/\d\.\d"\s+2\d\d`, combines the two and sends:

```
%
success count $\backslash$sHTTP/$\backslash$d$\backslash$.$\backslash$d"$\backslash$s+2$\backslash$d$\backslash$d
```

Note that the pattern can include magic cookies (see the magic cookies section) as of remstats version 0.12.2.

14.3 nt-status-collector - stats from Windows NT hosts

14.3.1 Usage:

```
%
nt-status-collector version 1.25
usage: ../nt-status-collector [options]
where options are:
    -d nnn  enable debugging output at level 'nnn'
    -f fff  use 'fff' for config-dir [/home/remstats/etc/config]
```

```
    -F      force collection, even if it's not time
    -h      show this help
    -H HHH  only try hosts from 'HHH', a comma-separated list
    -p ppp  connect to server on port 'ppp' [1957]
    -t ttt  set timeout to 'ttt' [25]
    -u      ignore uphosts file
```

14.3.2 Description:

The `nt-status-collector` gets its data from the nt-status-server (see the nt-status-server section) . It sends a query consisting mostly of the sections of the server that it wants to run. It can also request that processes with specific names be counted and that information returned too. A query for all the sections might look like:

```
%
SRVINFO
PERFCOUNTERS
PULIST
MSDRPT
USRSTAT ntdomain
NET-VIEW
GO
```

Note that `PULIST`, `MSDRPT`, `USRSTAT` and `NET-VIEW` aren't currently used by anything, but they may be usefull for something. Also, USRSTAT wants an NT domain-name with the query.

14.4 ping-collector - get reachability of hosts

14.4.1 Usage:

```
%
ping-collector version 1.16
usage: ../ping-collector [options]
where options are:
    -d nnn  enable debugging output at level 'nnn'
    -f fff  use 'fff' for config-dir [/home/remstats/etc/config]
    -F      force collection even if it's not time
    -h      show this help
    -H HHH  only try hosts from 'HHH', a comma-separated list
    -u      for compatibility with run-remstats; ignored
```

14.4.2 Description:

ping-collector uses multiping (see the multiping section) to get numbers on how reachable the hosts are. Each host is sent 10 pings (ICMP echo-request) and the number of responses and the min, max and average RTT (Return Trip Time) is logged, giving the variables ping-sent, ping-rcvd, pingrtt-min, pingrtt-avg and pingrtt-max.

14.5 multiping

14.5.1 Usage:

```
%
usage: ../multiping/multiping [-Rdfnqrtv] [-c count] [-i wait] [-l preload]
            [-p pattern] [-s packetsize] host

       Options:
       R: ICMP record route
       d: set SO_DEBUG socket option
       f: 'flood' mode
       n: force addresses to be displayed in numeric format
       r: set SO_DONTROUTE socket option
       t: show results in tabular form
       v: verbose mode for ICMP stuff
```

14.5.2 Description:

Multiping runs pings in parallel which permits you to ping lots of hosts quickly. I wrote a program using Net::Ping to try to how bad a simple-minded approach was. With .025s between pings in my programs and 1s in multiping, my program took 17 minutes and multiping took 37 seconds. Maybe someone will not be able to get multiping going and will re-write it in perl. Not me. I'd be happy to include it if some-one wants to send me a copy.

14.6 port-collector - get service status

14.6.1 Usage:

```
%
port-collector version 1.15
usage: ../port-collector [options]
where options are:
    -d nnn  enable debugging output at level 'nnn'
    -f fff  use 'fff' for config-dir [/home/remstats/etc/config]
    -F      force collection even if it's not time
    -h      show this help
    -H HHH  only try hosts from 'HHH', a comma-separated list
    -t ttt  set default timeout to 'ttt' [5]
    -u      ignore uphosts file
```

14.6.2 Description:

The `port-collector` gets data about services running on specified TCP ports. It will attempt to connect to the specified port on the host, and optionally send a string to it. It will then examine the response, if any, for certain patterns. This permits it to query almost any text-based protocol. For information on how to set up a new service, look in the scripts (see the scripts section) directory in the configuration directory.

The rrd specification can also override the port specified by the script, by appending a colon and the port-number. E.G. for a web-server on port 8000, you could specify the rrd like:

```
%
rrd port-http:8000
```

According to whether it can connect to the service and what response it gets back, it sets a status to one of OK, WARN, ERROR, CRITICAL. These are arbitrary levels, except that OK means normal, and their meanings are determined by the configuration file. The `port-collector` will also log how long it took the service to respond. These numbers are not intended for benchmarking, but only for determining the health of the service.

14.6.3 Returned Data

The variables returned by the port-collector are:

```
%
port-PORTNAME - containing the status of the port, calculated by
the script for this port
port-PORTNAME-response - containing the response-time for the query
```

14.6.4 Other data from the port-collector

The main RRD for the port-collector is `port-*`, but it is possible to define other RRDs. If you want to collect information from the results elicited by the send string, you can provide either or both of the `infopattern` or `valuepattern` in the script associated with the RRD. The script must be named the same as the rrd. Look in the scripts configfile docs (see the scripts configfile docs section) for details on scripts.

A matching `valuepattern` will cause the port-collector to return variables named `RRDNAME:value#` with "`#`" replaced by a single digit, corresponding to the number of the parenthesized part of the pattern that was matched.

A matching `infopattern` will cause the port-collector to create status files for this host called INFO1-RRDNAME. None of the existing pagemakers (see the pagemakers section) use these status files, but the view-writer (see the view-writer section) could do so if a view template (see the view template section) refered to them.

14.7 remoteping-collector - reachability from other sites

14.7.1 Usage:

```
%
remoteping-collector version 1.10
usage: ../remoteping-collector [options]
where options are:
    -d nnn  enable debugging output at level 'nnn'
    -f fff  use 'fff' for config-dir [/home/remstats/etc/config]
    -h      show this help
    -p ppp  use port 'ppp' instead of the default [1959]
    -t ttt  use 'ttt' for timeout [60]
    -u      for run-remstats compatibility
```

14.7.2 Description:

The `remoteping-collector` is intended to gather ping statistics (see ping-collector (see the ping-collector section)) from remote sites. It works by contacting remoteping-server (see the remoteping-server section) s running on other machines. This way it will be possible to monitor the same list of hosts from multiple points on the network and determine several things:

- general health of parts of the network of interest to the co-operating parties, and
- if certain parts of the network are performing better or worse than others.

14.7.3 Note:

Note the use of the future tense in the previous paragraph. I'm looking for volunteers to run the remoteping-server (see the remoteping-server section) and let me have access to it. In return, I'll let you look at the stats that this process gathers. I'm planning to monitor ISPs and other sites across Canada, as well as commonly accessed sites around the world, to determine how we're doing in network performance, here in Canada. If you want to volunteer, please hit the mailto URL below and ignore the bounce from my spam protection (see **http://silverlock.dgim.crc.ca/~terskine/qmail/tms.html**).

14.8 snmp-collector - get data via SNMP

14.8.1 Usage:

```
%
snmp-collector version 1.15
```

```
usage: snmp-collector [options]
where options are:
    -c ccc  use 'ccc' for the read community string; overrides host
    -d nnn  enable debugging output at level 'nnn'
    -f fff  use 'fff' for config-dir [/home/remstats/etc/config]
    -F      force collection even if it's not time
    -h      show this help
    -H HHH  only try hosts from 'HHH', a comma-separated list
    -u      ignore uphosts file
```

14.8.2 Description:

The snmp-collector collects data available via SNMP. There are some things that are hardcoded in, but it's mostly configurable. It will attempt to query for the following, if available:

- **sysDescr** - tells what kind of a device this is
- **sysUptime** - how long it has been up

and use them in the host index page.

The `snmpif-*` rrd is wired into the snmp-collector (for now) and causes it to fetch the following for each interface:

- **ifType** - interface type
- **ifOperStatus** - operational status
- **ifSpeed** - interface speed
- **ifInErrors** - input errors
- **ifOutErrors** - output errors
- **ifInOctets** - input octets (aka bytes)
- **ifOutOctets** - output octets (aka bytes)
- **ifInUcastPkts** - input unicast packets
- **ifOutUcastPkts** - output unicast packets
- **ifInNUcastPkts** - input non-unicast (broadcast and multicast) packets
- **ifOutNUcastPkts** - output non-unicast (broadcast and multicast) packets

The sysDescr and sysUptime are saved for the host display and the ifType and ifSpeed are combined to give a hardware description for the interface. Crude, but portable.

For other SNMP data, you'll need to look at the [oids] (see the [oids] section) file in the configuration directory. The rrd will need to contain `oid` lines specifying names assigned in the oids section (see the oids section section) . If the host doesn't have one, the rrd will also need to specify a `community`. Here's an example:

```
%
[rrd snmpmem]
source          snmp
step            300
data            freemem=ciscofreemem GAUGE:600:0:U
data            totalmem=ciscototalmem GAUGE:600:0:U
archives        day-avg week-avg month-avg year-avg
times           day yesterday week month year
oid             CiscoFreeMem
oid             CiscoTotalMem
```

This rrd definition will fetch the amount of free memory and total memory available on a Cisco router. Since it's querying a Cisco-specific MIB, it's not usefull on other gear.

14.9 snmp-route-collector

14.9.1 Usage:

```
%
snmp-route-collector version 1.12
usage: snmp-route-collector [options]
where options are:
    -c ccc  use 'ccc' for the read community string; overrides host
    -d nnn  enable debugging output at level 'nnn'
    -f fff  use 'fff' for config-dir [/home/remstats/etc/config]
    -F      force collection even if it's not time
    -h      show this help
    -H HHH  only try hosts from 'HHH', a comma-separated list
    -u      ignore uphosts file
```

14.9.2 Description:

This is a specialized form of SNMP collector which walks a part of the BGP4 MIB, specificly `bgp4PathAttrBest`, to count routes available from a given BGP peer host. It notes both the total number of routes and how many of them are the best route to that destination.

Unfortunately, it doesn't scale. On routers with large numbers of peers it can take a **long** time to troll through all the routes. This needs to be replaced.

14.10 unix-status-collector - stats from unix hosts

14.10.1 Usage:

```
%
unix-status-collector version 1.16
usage: unix-status-collector [options]
where options are:
    -d nnn  enable debugging output at level 'nnn'
    -f fff  use 'fff' for config-dir [/home/remstats/etc/config]
    -F      force collection even it it's not time yet
    -h      show this help
    -H HHH  only try hosts from 'HHH', a comma-separated list
    -p ppp  connect to server on port 'ppp' [1957]
    -t ttt  set timeout to 'ttt' [10]
    -u      ignore uphosts file
```

14.10.2 Description:

The unix-status-collector gets its data from the unix-status-server (see the unix-status-server section) . For a detailed explanation of what the various directives mean, see its documentation, as it's the implementor. It sends a query consisting mostly of the sections of the server that it wants to run. It can also request that processes with specific names be counted and that information returned too. A query for all the sections might look like:

```
%
UNAME
UPTIME
TIME 986485967
VMSTAT
DF
NETSTAT
QMAILQ
PS
webservers ps count httpd
FILEAGE
test fileage /var/spool/locks/lockfile
PROC
swaptot proc /proc/meminfo \^SwapTotal:$\backslash$s+($\backslash$d+)
GO
```

The `webservers ps count httpd` line requests that the ps section count the number of processes called `httpd` and return that as a variable called `webservers`.

The `test fileage /var/spool/locks/lockfile` line requests the last modification time of the file `/var/spool/locks/lockfile`, which is returned in seconds.

The `swaptot ...` line looks for the total swap size in `/proc/meminfo`.

The best way to see what it will produce is to run it manually.

Chapter 15

updater

15.1 updater - add new data to RRDs

15.1.1 Usage:

```
%
updater version 1.9
usage: updater [options] collector
where options are:
-d nnn enable debugging output at level 'nnn'
-f fff use 'fff' for config-dir [/home/remstats/etc/config]
-h show this help
```

15.1.2 Description:

It reads collector output in standard form from stdin and updates the appropriate RRDs. It wants to know which collector the information came from to avoid looking for information that won't be there.

Chapter 16

Monitors

16.1 Monitors

Currently, there are three monitors:

- ping-monitor (see the ping-monitor section) - determines reachability of hosts
- alert-monitor (see the alert-monitor section) - figures out status of various values specified in the rrds (see the rrds section) and hosts (see the hosts section) config-files
- topology-monitor (see the topology-monitor section) - to analyze changing routes to your monitored hosts

16.2 alert-monitor - a status evaluator and alert trigger

16.2.1 Usage:

```
%
alert-monitor version 1.20
usage: ../alert-monitor [options]
where options are:
-d nnn  enable debugging output at level 'nnn'
-f fff  use config-dir 'fff'[/home/remstats/etc/config]
-h      show this help
-s sss  search 'sss' data samples for values [5]
-u      generate alerts for hosts unreachable through a down host
```

16.2.2 Description:

The `alert-monitor` compares the current value of variables specified in the alerts file (see the alerts file section) in the configuration directory with threshold values and sets the status of those variables accordingly. It saves the current status of variables in `/home/remstats/data/ALERTS`.

What value corresponds to what status level is set in the rrd definition (see the rrd definition section) or sometimes the host definition (see the host definition section) . This way an rrd definition will specify generally reasonable levels, but they can be overridden for hosts where they aren't reasonable.

For an rrd definition, an alert line looks like:

```
%
alert varname relation oklevel [warnlevel [errorlevel]]
```

or

```
%
alert varname nodata status
```

[The latter says that missing data for variable `varname` will cause its status to be level `status`.]

For a host-specified alert level, the line looks like:

```
%
alert rrdname varname relation oklevel [warnlevel [errorlevel]]
```

or

```
%
alert rrdname varname nodata status
```

and the interpretation is the same, except that you're having to say which rrd this alert refers to.

The available relations are:

```
%
$<$ (value is less than threshold)
$>$ (value is greater than threshold)
= (value is equal to threshold)
|$<$ (absolute value of value is less than threshold)
|$>$ (absolute value of value is greater than threshold)
delta$<$ (difference between last two values is less than threshold)
delta$>$ (difference between last two values is greater than threshold)
$>$daystddev (value is outside threshold * the past day's standard-deviation)
$>$weekstddev (value is outside threshold * the past day's standard-deviation)
$>$monthstddev (value is outside threshold * the past day's standard-deviation)
```

16.2.3 Example

To make things more concrete for the first (normal) case, here's a real example, from the `load` rrd supplied in `config-base`:

```
%
alert load5 $<$ 3 7 10
```

This means that if the `load5` variable is less than 3, the status is set to OK. If it's less than 7, it's WARN, less than 10 it's ERROR and more than that, it's CRITICAL.

Since the first match is taken, it's possible to leave out the upper levels if you don't want them to ocurr. For example if you only wanted `load5` to ever go to WARN level, never above, you could use:

```
%
alert load5 $<$ 3
```

and then the only possible status levels are OK and WARN.

The possible `relation`s are: <, =, >, —<, —>, delta<, delta>. The first three should be obvious. The next two allow comparisons to the absolute value of the variable's current value. The last two allow comparisons to the change in value.

16.2.4 Causing alerts

Depending on the lines in the alerts file (see the alerts file section) , the status may also trigger alerts. A matching line in the alerts config-file (see the alerts config-file section) will cause `alert-monitor` to run the alerter (see the alerter section) for each of the specified recipients. It will also be passed, in order:

- **recipient** - the recipient; for alert-email (see the alert-email section) it will be an email address
- **hostname** - the name of the host that the alert applies to
- **ip** - the IP number for that host, in case it's not in DNS
- **rrdname** - the name of the RRD
- **wildpart** - the wild part of a wildcard RRD. E.G, for an RRD of `port-ftp` (using the wildcard RRD `port-*`) the wildpart would be `ftp`.
- **variable** - the name of the variable
- **status** - the current status, as decided by alert-monitor
- **old_status** - the previous status
- **value** - the current value of the variable

- **relation** - the relation used to compare the variable to the threshold, mostly for creating informative messages
- **threshold** - the threshold value that was exceeded
- **start** - timestamp of when the alert started
- **duration** - number of seconds that the alert has been active
- **host-description** - the description field from the host config-file
- **rrd-description** - the description tag on this rrd (desc="xxx")
- **webmaster** - the email address of the remstats person
- **template** - the name of the template file to generate the message from.

16.3 alerter - construct and send alert text

16.3.1 Usage:

```
%
alerter version 1.8
usage: alerter [options] args
where options are:
    -d ddd  set debugging output to level 'ddd'
    -h      show this help
    -f fff  use 'fff' as configuration directory [/home/remstats/etc/config]
The args are documented in alert-monitor; a quick list:
    towho host ip realrrd wildpart var status old_status value relation
    threshold alertstart duration hostdesc rrddesc webmaster template
```

16.3.2 Description:

[`Alerter` may be rolled into the `alert-monitor` at some point in the future. It was easier to test as a separate program, and the performance hasn't been an issue for me.]

`Alerter` is passed its parameters (specified above) by the alert-monitor (see the alert-monitor section) . Most of them are used to fill in information in the text of the alert. The interesting ones are `towho` and `template`.

It also reads the alert-destination-map config-file (see the alert-destination-map config-file section) to decide where the alert needs to go. This will give it a list of (method, address) pairs.

For a given template-name, say `xxx`, and method, say `method`, it will look for files in /home/remstats/etc/config/alert-templates, called:

```
%
method-xxx
```

```
method-DEFAULT
xxx
DEFAULT
```

and take the first one it finds. Similarly, it will look for a header to add to the top of the template called:

```
%
method-HEADER
HEADER
```

and a footer in one of:

```
%
method-FOOTER
FOOTER
```

The three pieces will be concatenated giving the template text. Then substitutions will be done for the following ##MAGICCOOKIES##:

```
%
HOST IP REALRRD WILDPART FIXEDRRD VAR STATUS OLDSTATUS
VALUE RELATION THRESHOLD START DURATION HOSTDESC RRDDESC
NOW TEXTNOW ALERTHOST TOWHO WEBMASTER
```

This gives the alert text. From the method definition in the alert-destination-map config-file (see the alert-destination-map config-file section) `alerter` knows which program to run to send the alert text to the appropriate address, and it does it.

16.3.3 Alert-Sending Scripts

These are now easy to write, and in many cases you won't even have to write one. There are two requirements for an alert-sending script:

1. It must take an address to send to on the command-line, and
2. It must accept the text on stdin.

E.G. you could use sendmail with no wrapper.

16.4 alert-email - an alert sending script

This is a simple script intended to be run by the alerter (see the alerter section) . Like all alert-senders, it takes one argument on the command-line: the "address" to send the alert to. The text of the alert is fed to this script on stdin.

It sends the alert text to "address" via email, by invoking sendmail, though there's no reason that it couldn't be re-written to do an SMTP injection directly if some-one wanted to. With no error-checking, it could be re-written as:

```
%
\#!/bin/sh
/usr/lib/sendmail "\$1"
```

16.5 alert-winpopup - an alert sending script

This is a simple script intended to be run by the alerter (see the alerter section) . Like all alert-senders, it takes one argument on the command-line: the "address" to send the alert to. The text of the alert is fed to this script on stdin.

It sends the alert to use "address", in this case a windows NetBIOS machine name, via a Windows popup message.

16.6 ping-monitor - determine reachability status

16.6.1 Usage:

```
%
ping-monitor version 1.5
usage: ../ping-monitor [options]
where options are:
    -d nnn  enable debugging output at level 'nnn'
    -f fff  use 'fff' for config-dir [/home/remstats/etc/config]
    -h      show this help
    -s sss  examine 'sss' samples [5]
```

16.6.2 Description:

The `ping-monitor` looks at the last 5 samples (by default) of ping data and determines the status of the host. It will choose one of the following four statuses:

- **UP** - the host is up now and has always (throughout the sample period) responded to pings.
- **UPUNSTABLE** - the host is up now, but on at least one of the samples, it did not respond.
- **DOWNUNSTABLE** - the host is not responding now, but it has responded within the sample period.
- **DOWN** - the host is down now and has not responded within the sample period.

It also writes coloring information for the ping-index web-page.

16.7 topology-monitor

16.7.1 Usage:

```
%
topology-monitor version 1.6
usage: topology-monitor [options] oldfile newfile
where options are:
-d enable debugging output
-f fff use 'fff' for config-dir [/home/remstats/etc/config]
-h show this help
```

16.7.2 Description:

The `topology-monitor` runs out of do-traceroutes (see the do-traceroutes section) to find changes in network paths to the monitored hosts. After `do-traceroutes` has run traceroute for each monitored host, the `topology-monitor` compares the current network path to that host with the previous path. Currently, all that is done with this information is to log when it changes.

Chapter 17

run-remstats

17.1 run-remstats - run a complete cycle

Chapter 18

run-remstats

18.1 Usage:

```
%
run-remstats version 1.9
usage: ../run-remstats [options] [-- options-to-be-passed]
where options are:
    --debug=nnn       enable debugging output at level 'nnn'
    --config_dir=fff  use 'fff' for config-dir [/home/remstats/etc/config]
    --help            show this help
```

18.1.1 Description:

`run-remstats` is the main script for a remstats collection machine. As a simplified overview:

- check-config (see the check-config section) is run first.
- In parallel, all the collectors (see the collectors section) are run, each feeding it's own updater (see the updater section) process. Some of them query remstats servers (see the servers section) , some get their information in other ways. So you have a bunch of pipelines like:

  ```
  %
  xxx-collector | updater xxx
  ```

- When all the collectors have finished, the monitors (see the monitors section) get run in parallel to figure out what's happening.
- Afterwards, if the configuration directory has changed, run the pagemakers (see the pagemakers section) , to re-do the web-pages.
- Finally, it prints all the stderr output of all the various programs, separated by program.

For each of these programs, `run-remstats` will set a timer (see `watchdogtimer` in the general config-file (see the general config-file section)). If the timer expires and the program is still running, `run-remstats` will kill that process. This avoids the problem of a hanging collector hanging the whole remstats cycle.

It also manages a lock-file to make sure that two instances don't run concurrently. The lock-file's name is based on the name of the `run-remstats` script. (See **Running multiple copies of run-remstats** below.)

It keeps a status file in the configured temp directory (`/home/remstats/tmp` by default) which is used by monitor (see the monitor section) to show where the `run-remstats` process has gotten to.

When starting, it will also look for a file in the tmp directory called `STOP-run-remstats` (default), and if it exists, will refuse to run at all.

18.1.2 Running multiple copies of run-remstats

If you symlink `run-remstats` to `run-remstats-XXX`, then the default configuration directory for `run-remstats-XXX` will be `/home/remstats/etc/config-XXX`. Since the lock-file is named for the script which invokes it, you won't have collisions between the two instances, as long as your configuration files don't conflict. You can have multiple collector-only instances collecting data which is formatted by a single pagemaker instance, (in theory) but this will require at least three config-dirs which must be closely co-ordinated. If you want to do this for performance reasons, I do plan to address this in future.

18.1.3 Configuration:

See the general (see the general section) config-file. The lines to configure run-remstats are:

```
%
pinger, collectors, monitors, pagemakers, watchdogtimer
```

18.2 check-config

18.2.1 Usage:

```
%
check-config version 1.14
usage: check-config [options]
where options are:
-c      write SNMP communities
-C CCC  set configuration-debugging output to level 'CCC'
-d ddd  enable debugging output at level 'ddd'
-D      dump configuration (must have DEBUG enabled in fixup.config)
-e      print environment variables to stdout
-f fff  use config-dir 'fff' [/home/remstats/etc/config]
```

```
-h      show this help
-l lll list 'lll' on stdout (comma-separated list of: host, ip, rrd)
-s sss  shell type to use for -e [sh]
-t      test-mode; don't make any changes
```

18.2.2 Description:

`check-config` uses the common `read_config` routine to read the configuration file which will confirm that it can be read successfully by other programs. It also makes sure that the data directories for all hosts exist, and creates new RRDs.

run-remstats (see the run-remstats section) also uses the `-e` option to make basic configuration information avaiable to the shell.

18.3 Pagemakers

The remstats pagemakers make web-pages, and update other information used in making web-pages. They only need to be run when the configuration file has changed and run-remstats (see the run-remstats section) is smart enough to do that.

- graph-writer (see the graph-writer section) - makes web-pages with the graphs and links them together
- snmpif-setspeed (see the snmpif-setspeed section) - sets maximums on snmpif-* rrds
- datapage-interfaces (see the datapage-interfaces section) - makes data-pages for every snmpif-* rrd
- datapage-inventory (see the datapage-inventory section) - lists all monitored hosts, uptime, software and hardware
- snmpif-description-updater (see the snmpif-description-updater section) - updates the descriptions for snmpif-* rrds from the SNMP descriptions

18.4 graph-writer

18.4.1 Usage:

```
%
graph-writer version 1.15
usage: ../graph-writer [options] collector
where options are:
-d nnn enable debugging output at level 'nnn'
-D enable configuration debugging output
-f fff use 'fff' for config-dir [/home/remstats/etc/config]
```

`-h show this help`

18.4.2 Description:

This is the main remstats pagemaker (see the pagemaker section) . It makes the web-pages with the graphs and other pages to link them together and organize them. There are three kinds of page that it makes:

- **Indices** - The main three indices are the **Overall Index**, the **Ping Index** and the **Quick Index**. Each of these shows all the hosts being monitored, grouped by the group you assigned them to. The **Overall Index** shows a section for each host, with a link to all of the graphs for that host. The **Ping Index** shows a small graph of the last two hours of ping data for that host, for each host, with the graph background specially coloured for hosts which aren't reachable. The **Quick Index** shows, for each host: a link, a status indicator and optionally a link to alert.cgi (see the alert.cgi section) for alerts for that host. There is also the **Custom Index** to show links to all the customgraphs (see the customgraphs section) .

- **Host Pages** - For each host, there is a **Host Page** which shows some information about the host and all the day graphs for that host. The graphs are all links to ...

- **Graph Pages** - Each graph is also available in various timespans, depending on the times (see the times section) that you specified in the rrd (see the rrd section) definition which caused the generation of that graph.

Graph-writer is the replacement for both `grapher` and `html-writer`. Before version 0.10.0, `grapher` would make new graphs, as part of the update run, and `html-writer` would re-write the html pages. Now, `graph-writer` makes a CGI script for each web-page, using `rrdcgi` as its interpreter. `Rrdcgi` simply spits out the page as it was written with "magic cookies" replaced by <IMG SRC...> tags and makes sure that there is a recent version of the graph file. Much better than generating all the graphs every five minutes and have most of them never get looked at.

18.5 snmpif-setspeed

This is a gross hack which modifies all the `snmpif-*` rrds to set the maximum limits on all monitored interfaces. The input and output bps variables have their maximum set to the the maximum for that interface. The various packet counters have their maximums set to the maximum possible packets-per-second assuming minimum-length packets, for that interface speed.

You may not want to run this if you have better knowledge of what real maximums will be encountered for particular interfaces.

18.6 datapage-alert-writer

18.6.1 Usage:

```
%
Use of uninitialized value at ../datapage-alert-writer line 156.
datapage-alert-writer version
usage: ../datapage-alert-writer [options]
where options are:
    -d nnn  enable debugging output at level 'nnn'
    -h      show this help
```

18.6.2 Description:

This writes a datapage (see datapage.cgi (see the datapage.cgi section)) which gets linked into the header of each page as the "Alert Index". It gives a quick overview of alert statuses and values for each host. It ought to be self-explanatory.

18.7 datapage-interfaces

This makes a datapage (see the datapage section) for each host with `snmpif-*` rrds, showing all those interfaces.

18.8 datapage-inventory

This makes a single page showing all the monitored hosts. For each host, it shows:

- the uptime (from the `uptime` program or SNMP uptime)
- the hardware type (from the `uname` program), if available
- the software version (from the `uname` program or the SNMP system.sysDescr)

18.9 datapage-status

18.9.1 Usage

```
%
datapage-status version 1.4
usage: datapage-status [options] file ...
where options are:
-d enable debugging output
-e      show run-time errors in generated pages
-f fff use 'fff' for config-dir [/home/remstats/etc/config]
```

`-h show this help`

18.9.2 Description

The `datapage-status` program creates a datapage (to be interpreted by datapage.cgi (see the datapage.cgi section)) showing the current values of all variables in all RRDs for that host, in addition to the usual remstats headers.

18.10 view-writer

18.10.1 Usage:

`%`

18.10.2 Description:

The `view-writer` makes web-pages from the view definitions which are contained in the views config-dir (see the views config-dir section) . All the documentation relating to `view-writer` is there too, as it only implements what the views request.

18.11 remstats-monitor - watch remstats processes

18.11.1 Usage:

`remstats-monitor [sleeptime]`
where sleeptime is the time (in seconds) to wait between polls

18.11.2 Description:

This is primarily a development tool. It loops doing a `ps` command, weeding out everything except the remstats processes and cleaning up the results, to make it easier to read. It also shows the process-id from run-remstats (see the run-remstats section) lock-file, and the status from its status file.

It's not written portably and will probably have to be tweaked by hand if you want to run it. If you find it of interest, please let me (see the me section) know.

18.12 CGI Scripts

These are intended to be invoked via the html-writer created toolbars, to do the supplied functions to the host in question.

- alert.cgi (see the alert.cgi section) - Shows the current alert status of selected rrd variables.
- availability-report.cgi (see the availability-report.cgi section) - Shows availability of RRD variables.
- dataimage.cgi (see the dataimage.cgi section) - Generates images based on live data.
- datapage.cgi (see the datapage.cgi section) - Generates web-pages containing dynamic data.
- graph.cgi (see the graph.cgi section) - Allows non-remstats web-pages to show remstats graphs.
- log-event.cgi (see the log-event.cgi section) - log a manual event.
- ping.cgi (see the ping.cgi section) - Ping the host.
- showlog.cgi (see the showlog.cgi section) - Display selected portions of the remstats log files.
- traceroute.cgi (see the traceroute.cgi section) - find network path to a host
- whois.cgi (see the whois.cgi section) - look up information about hosts, IP#s, ...

18.13 alert.cgi - Alert Reporting and Updating

This CGI script will generate the Alert Report and also let you modify it. You can turn alerts off for a specific line (by checking the `quench` check-box). You can also attach comments to a specific alert, for example to let other people know that you're already working on it, or when a service will be available again.

18.14 availability-report.cgi

The `availability-report.cgi` calls availability-report (see the availability-report section) to produce a report of "availability" according to the definitions in the availability (see the availability section) config-file.

18.15 dataimage.cgi - create images driven by live data

18.15.1 Usage:

```
%
$<$IMG SRC="http://remstats.sourceworks.com:1954/dataimage.cgi?imagename"$>$
```

18.15.2 Description:

`Dataimage.cgi` reads an image definition from `/home/remstats/datapage/imagename.image`. Some of the commands are in common with datapage.cgi (see the datapage.cgi section) and are documented there:

```
%
oid, rrd, status, eval, debug, macro, macroend and *EOD*
```

These retrieve and manipulate data. There are also commands to create images:

```
%
image, colordef, color, linewidth, line, rectangle, circle,
fill, font, text, out, flow
```

18.16 Image Commands

18.16.1 image

The image command has two formats. The first looks like:

```
%
image WIDTH HEIGHT
```

This creates a blank image of the size specified. Sometimes you'll want a background for the image, and you can use the second form to specify a file to read for the background:

```
%
image BGFILE
```

This will create the new image the same size as the one in `BGFILE`, by reading `BGFILE` and using its contents as the background. N.B., the image must be a PNG graphic.

The `image` command also defines a few colors (see `colordef` below): `black`, `white` and `transparent`, sets the current color to `black`, fills the image with `white` and sets the `linewidth` to 1.

18.16.2 colordef

[It can also be spelled `colourdef`.]

This defines a new colour and names it. The command looks like:

```
%
colordef COLORNAME RED GREEN BLUE
```

where `RED`, `GREEN` and `BLUE` specify the level of each of those colours to be mixed to define the colour referred to in the script as `COLORNAME`.

18.16.3 color

[It can also be spelled `colour`.]

This sets the current colour, to be used by those commands that don't specify a colour. It is used as simply:

```
%
color COLORNAME
```

18.16.4 linewidth

This sets the width of lines. It isn't honoured by all other commands, unfortunately, but so far this hasn't been a problem for me. It looks like:

```
%
linewidth WIDTH
```

18.16.5 line

This just draws a line in the current `color` and `linewidth`:

```
%
line X1 Y1 X2 Y2
```

18.16.6 rectangle

This is a way to draw a rectangle, without useing `line` 4 times:

```
%
rectangle X1 Y1 X2 Y2 [filled]
```

The co-ordinates (`X1`, `Y1`) and (`X2`, `Y2`) define oposite corners of the rectangle. If the keyword `filled` is added to the end, the rectangle will be filled with the current colour as well.

18.16.7 circle

Here you get a circle:

```
%
circle X Y RADIUS [filled]
```

The circle will be centered on (`X`, `Y`) with a radius of `RADIUS`. If the keyword `filled` is added to the end, the circle will be filled with the current colour as well.

18.16.8 fill

This command permits you to fill arbitrary regions:

```
%
fill X Y [COLORNAME]
```

The `COLORNAME` is optional.

18.16.9 text

The `text` command sets text into the image, for labelling things:

```
%
text X Y TEXT
```

18.16.10 font

This changes the font for the `text` command:

```
%
font (giant|large|mediumbold|medium|small|tiny)
```

18.16.11 out

This permits the script to output additional information to an auxiliary file. I added this for doing image-maps automatically, which can be automatically loaded by a datapage.cgi (see the datapage.cgi section) web-page.

The syntax is:

```
%
out TEXT
```

18.16.12 flow

This draws a strange double-headed, bi-coloured arrow. Think of it as two half arrows, split lengthwise, one in each direction. The colour and width of each half arrow indicates the flow in that direction. I use it for indicating network traffic flow, which usually isn't the same in both directions. It looks like:

```
%
flow X1 Y1 X2 Y2 INFLOW OUTFLOW
```

The co-ordinates (`X1`, `Y1`), (`X2`, `Y2`) indicate the ends of the flow. `INFLOW` and `OUTFLOW` indicate the level in each direction, relative to (`X1`, `Y1`).

18.17 datapage.cgi - dynamic data in web-pages

18.17.1 Usage:

```
%
$<$A HREF="http://remstats.sourceworks.com:1954/datapage.cgi?pagename"$>$whatever$<$/A$>$
```

18.17.2 Data Collection

`Datapage.cgi` looks for the page definition in the file `@@DATAPAGEDIR@@/pagename.page`
The page definition is in two parts, separated by a line like:

```
%
BEGIN-PAGE
```

The first part's purpose is to define variables to be included in the second part, which is an HTML template, with magic cookies.

All lines in the first or definition part are subject to variable interpolation. Any ocurrance of `${variablename}` will be replaced by the current contents of the variable `variablename`. This will be done up to five levels, permitting expansion of ${${h}_${interface}}, providing that you've got values for the variables `h` and `interface`. N.B. variable names must be lower case.

In addition, within a macro expansion, macro-arguments will also be interpolated, before variable interpolation, for all ocurrances of `${ARGNAME}`, assuming that there is an argument for the current macro called `ARGNAME`. N.B. macro argument names must be UPPER case.

18.17.3 Common Commands

The commands permitted in the first part are:

```
%
oid, rrd, status, eval, debug, macro, macroend,
alertstatus, alertvalue and *EOD*
```

These commands are in common with the dataimage.cgi (see the dataimage.cgi section) script, but are only documented here.

- oid This fetches an SNMP value into a datapage variable. The command looks like:

  ```
  %
  oid VARNAME HOSTNAME OIDNAME
  ```

  The `VARNAME` is the name of the datapage variable (let's just call them variables from now on).

The HOSTNAME is the name of the host to query. The SNMP community-string is usually supplied in that host's config-file, but can be supplied in usual MRTG fashion by giving COMMUNITY@HOSTNAME or even COMMUNITY@HOSTNAME:PORTNUMBER instead of the HOSTNAME.

The OIDNAME must be defined in the oids config-file (see the oids config-file section) , but can be suffixed by the usual numbers. E.G. you can use ifName.4 to get the ifName for interface 4.

- rrd This fetches a value from an RRD database into a variable. It looks like:

```
%
rrd VARNAME HOSTNAME RRDNAME DSNAME CF
```

The RRDNAME is the name of the rrd, as remstats knows it, not fully qualified. I.E. it will be under the config-file defined `datadir`, and under the host's directory under that.

The DSNAME is the ds-name within that RRD file and the CF is the usual RRD consolidation-function to be applied.

- status This is so-named because it fetches remstats status files, usually written by the various collectors (see the collectors section) and monitors (see the monitors section) . It looks like:

```
%
status VARNAME HOSTNAME STATUSNAME
```

The STATUSNAME is the name of the status file, as named in the host's data directory. There is a standard mapping applied by the function `to_filename` from the `remstats.pl</t` file to munge the filename so that it won't conflict with the filesystem. Either look for the name in the data directory, use the function (see eval) or look at the code. I **am** planning on changing the mapping when I figure out the best way to do it.

- eval The eval command lets you modify the values fetched by previous `oid`, `rrd`, `status` and `eval` commands with arbitrary perl code. It looks like:

```
%
eval VARNAME PERLEXPRESSION
```

The PERLEXPRESSION is a perl expression and can be arbitrarily complex, but gets messy quickly with the `datapage.cgi` and perl both doing variable interpolation.

Note: `datapage.cgi` uses private.pl (see the private.pl section) , so you can include commonly used functions here to make your datapage creation easier.

- debug The `debug` command takes a number which is the level to set debugging to. It causes extra output which may be helpful in figuring out why your page isn't working the way you expected.
- alertstatus This lets you fetch the alert level for a given (host, rrd, dsname, cf) combination. The command looks like:

  ```
  %
  alertstatus VARNAME HOSTNAME RRDNAME DSNAME [CF]
  ```

  This will fetch the alert status and put it in the datapage variable `VARNAME`. The status will be the same set of values shown on the alerts report (see the alerts report section) for status. The `CF` parameter is optional and is rrdtool's consolidation function. It will be set to `AVERAGE` if it's not supplied.
- alertvalue This is the same as `alertstatus` except that it sets the variable to the current value of the (host, rrd, variable, cf) combination.

18.17.4 The HTML template

This is almost just HTML with a few magic cookies inserted. The difference is that the beginning must include HTTP headers. If you don't want anything fancy, just begin like:

```
%
------ cut here ------
BEGIN-PAGE
content-type: text/html

%
------ cut here ------
```

Note: the empty line after `content-type:` is **not** optional. It's necessary to end the HTTP headers.

The magic cookies are:

- <DATAPAGE::STATUS host statusfile> inserts a specified status file
- <DATAPAGE::VAR varname> interpolates the value of a datapage variable
- <DATAPAGE::HEADER title> generates a standard remstats header
- <DATAPAGE::STATUSHEADER hostname> generates the status headers for the named host
- <DATAPAGE::TOOLBAR hostname> generates the toolbar for the named host

- <DATAPAGE::FOOTER> generates a standard remstats footer
- <DATAPAGE::INCLUDE filename> include the contents of a file from the datapage directory, for imagemaps ...
- <DATAPAGE::PATHINCLUDE filename-with-path> include contents of a file specified with a complete path
- <DATAPAGE::MACRO macroname [argvalue] ...> include boilerplate HTML with substitutions
- <DATAPAGE::GRAPH host rrd graph time> generate the specified remstats graph
- <DATAPAGE::CUSTOMGRAPH graph time> generate the specified remstats customgraph
- <DATAPAGE::ERRORS> inserts the text of errors encountered in generating the page. Without this one, you won't see any errors. That way you include the errors and debugging output (see next item), which you're creating/debugging the datapage and afterwards turn them off. The errors and debugging output may include information you don't want to reveal to outsiders. Also, collecting all the error output together avoids spoiling the formatting of the page.
- <DATAPAGE::DEBUG> inserts debugging output. Without it, you won't see any debugging output.

18.18 graph.cgi - exporting remstats graphs

The purpose of `graph.cgi` is to allow remstats graphs to appear on external (not part of remstats) web-pages. It's **not** efficient, with the graphs being generated whenever the page is reloaded, but it is portable. All you do is to create an <IMG SRC...> tag with the appropriate values, like:

```
%
$<$IMG SRC="http://remstats.sourceworks.com:1954/graph.cgi?host=aaa\&rrd=bbb\&graph=ccc\&graphtime=ddd"$>$
```

and replace `aaa` with the name of the host, as remstats knows it, `bbb` with the name of the RRD, `ccc` with the name of the graph within that RRD, and `ddd` with the name of the timespan, from the times config-file (see the times config-file section) . If the RRD is a wildcard RRD, e.g. `snmpif-*`, then you must use the specific instance, e.g. `snmpif-eth0`.

That's all there is to it.

18.19 log-event.cgi - log events from a web-page

This shows up on the showlog.cgi (see the showlog.cgi section) as a link to permit you to manually enter events into the log. Don't feel obliged to enter all the fields. The data isn't checked for meaning, just for syntax. In other words, a host-name must look like a host-name, but it doesn't have to be a real host-name.

This cgi-script ought to be protected. See the web-server installation (see the web-server installation section) docs.

18.20 ping.cgi

The `ping.cgi` script allows you to ping a host. It's intended to be called off a host's toolbar, but that's not required. Simply provide the hostname or IP number and it'll ping it.

As an example, you could ping ftp.uu.net with a URL like:

```
%
http://remstats.sourceworks.com:1954/ping.cgi?host=ftp.uu.net (see \textbf{http://remstats.sourceworks.com:1954/ping.cgi?host=ftp.uu.net})
```

18.21 showlog.cgi

Any alert is also logged to the remstats log files (one file per day). Other information is also logged, for example, the topology-monitor (see the topology-monitor section) logs network topology changes. The `showlog.cgi` script allows you to display selected portions of the log files, by time-period, by host, ...

18.22 traceroute.cgi

This script uses traceroute (see the traceroute section) to find the path from the remstats host to some other specified host. It's intended to be called off a host's toolbar, but that's not required. For example, you could trace the path from here (trevelyan.sourceworks.com) to ftp.uu.net with a URL like:

```
%
http://remstats.sourceworks.com:1954/traceroute.cgi?host=ftp.uu.net (see the http://remstats.sourceworks.com:1954/traceroute.cgi?host=ftp.uu.net section)
```

There are other options that you can specify:

- `no_names` - just shows IP numbers instead of looking up the domain-names
- `ASNs` - look up the Autonomous System Numbers (ASNs) for the IP number of each hop. It can be useful for figuring out which networks you are traversing.
- `owners` - look up the "owner" via SOA records
- `fast` - continues on to the next hop as soon as the current one answers

18.23 whois.cgi

This script talks to the ARIN whois database (by default) to look up network names, IP numbers and AS numbers. It's usually linked into the results of traceroute.cgi (see the traceroute.cgi section) so that you can look up what your traceroute results actually mean.

Try `traceroute.cgi` if you want to see how it works.

Chapter 19

do-traceroutes

19.1 do-traceroutes - find the path to each host

This runs traceroute against each host being monitored. After they've all finished, it runs the topology-monitor (see the topology-monitor section) .

I'm planning to make graphical representations of how you are connected to the hosts you're monitoring, but that's not working yet.

19.2 traceroute

19.2.1 Usage:

```
%
2.9.2tee0.0: Usage: traceroute [-adnruvAMOQ] [-w wait] [-S start_ttl] [-m max_ttl] [-p port#] [-q nqueries] [-g gateway] [-t tos] [-s src_addr] [-g router] host [data size]
      -a: Abort after 10 consecutive drops
      -d: Socket level debugging
      -g: Use this gateway as an intermediate hop (uses LSRR)
      -S: Set start TTL (default 1)
      -m: Set maximum TTL (default 30)
      -n: Report IP addresses only (not hostnames)
      -p: Use an alternate UDP port
      -q: Set the number of queries at each TTL (default 3)
      -r: Set Dont Route option
      -s: Set your source address
      -t: Set the IP TOS field (default 0)
      -u: Use microsecond timestamps
      -v: Verbose
      -w: Set timeout for replies (default 5 sec)
      -A: Report AS# at each hop (from GRR)
      -M: Do RFC1191 path MTU discovery
      -O: Report owner at each hop (from DNS)
```

```
-P: Parallel probing
-Q: Report delay statistics at each hop (min/avg+-stddev/max) (ms)
-T: Terminator (line end terminator)
-U: Go to next hop on any success
```

19.2.2 Description:

Hmm. I think that describes its use pretty well. What does it do? Oh. Well it sends UDP packets with the time-to-live set to 1, then 2 then 3 and so on. This causes the routers that these packets are sent through to complain after the requisite number of hops. I.E. the first router complains about the first packets, with TTL set to one, the second about the packets with TTL set to two etc. Traceroute catches the complaints and times how long it took. This not only shows you how your packets are getting to the destination, but sometimes, where the congestion is as well. There's a lots better explanation in the source, so if you want more, UTSL (see **http://www.tuxedo.org/~esr/jargon/html/entry/UTSL.html**).

This version of traceroute is used in traceroute.cgi (see the traceroute.cgi section) , which isn't required, just handy on occasion, and in do-traceroute (see the do-traceroute section) , which you don't need unless you're curious about your routing and how it's changing over time. The only extra options that do-traceroute uses are the `-A` option to look up the ASN (Autonomous System Number) and the `-O` option to look up the DNS owner.

Chapter 20

Miscellany

20.1 Miscellaneous Scripts

These are scripts that don't really fit in anywhere.

- availability-report (see the availability-report section) shows availability of RRD variables
- genindex (see the genindex section) makes an index
- genmenu (see the genmenu section) makes the vertical menu-bars used in these docs.
- htmlpod (see the htmlpod section) makes pod files from html files (roughly).
- podhtml (see the podhtml section) makes html files from pod files.
- podlatex (see the podlatex section) makes LaTeX files from pod files.
- podpdf (see the podpdf section) makes PDF files from pod files.
- rrd-report (see the rrd-report section) produces reports from a raw rrd.

These are release-related scripts:

- convert-config-links (see the convert-config-links section) - copies links to files (just read it)

20.2 availability-report

20.2.1 Usage:

```
%
availability-report version 1.19
```

```
usage: availability-report [options]
where options are:
    -c      use colors in the output
    -d ddd  set debugging output to level 'ddd'
    -f fff  set config-dir to 'fff' [/home/remstats/etc/config]
    -h      show this help
    -H HHH  show only hosts HHH (comma-separated list) [all]
    -G GGG  show only groups GGG (comma-separated list) [all]
    -R RRR  show only rrds RRR (comma-separated list) [all]
    -g      show group summary
    -t ttt  availability for time-period ttt (start,finish)
```

20.2.2 Description:

This is mainly intended to be called from availability-report.cgi (see the availability-report.cgi section) . It provided a report on "availability" of specified RRD variables, by default, all that have definitions in the availability (see the availability section) config-file. Exactly what it means for a variable to be "available" is up to you. It's intended to give some measure of when a host or service isn't useable, so, e.g. the default definition of availability for the `ping` RRD variable `rcvd` (number of ping responses received) is:

```
%
ping rcvd MINIMUM $>$ 0
```

In english, the `rcvd` variable is considered unavailable if:

```
%
- it is less than or equal to zero (I.E. it didn't respond to ping)
- there is no data available for that time period
```

N.B.: The interaction between rrd archive consolidation and the xff value, (see rrdcreate), can result in longer periods of unavailability or conversely, masking periods of unavailability. Choose the consolidation function carefully to make sure you're getting the best data possible.

20.3 cleanup - removes stale, old files

20.3.1 Usage:

```
%
cleanup version 1.4
usage: ../cleanup [options]
where options are:
    -d nnn  enable debugging output at level 'nnn'
    -f fff  use 'fff' for config-dir [/home/remstats/etc/config]
    -h      show this help
```

20.3.2 Description:

It removes old collector data from /home/remstats/data/LAST, old logs from /home/remstats/data/LOGS, old traceroute data from /home/remstats/data/TRACEROUTES and old images from all the host subdirectories of /home/remstats/html.

Run it out of cron every now and then, say once a day, with a line like:

```
%
0 2 * * * /home/remstats/bin/cleanup
```

20.4 convert-config-links

20.4.1 Usage:

```
%
usage: ../convert-config-links [-h]
```

20.4.2 Description:

The problem is that an upgrade installation of remstats will overwrite the config-base directory, but previous installations of remstats created new configuration directories as symlinks to config-base. Some of these files need to be changed and some are commonly changed, specificly:

```
%
alerts alert-destination-map general html links tools
```

Installing a new version of remstats will overwrite config-base, including these files.

`convert-config-links` is a conversion tool for upgrading from remstats versions before 1.0A. It will convert the commonly changed config-files from symlinks to copies of the appropriate files from config-base. In remstats versions after 1.0A the new-config (see the new-config section) program will "Do the Right Thing" (TM) and make copies by itself, so you'll only have to run this once.

If you're installing remstats for the first time, you can ignore this program.

20.5 genindex - make an index from output of podhtml

20.5.1 Usage:

```
%
genindex version 1.4
usage: genindex [options] file ...
where options are:
```

```
-d enable debugging output
-f fff  use 'fff' format for output (html, pod or text)[pod]
-h show this help
```

20.5.2 Description:

`genindex` reads the index output of podhtml (see the podhtml section) and builds a crude index. It's mainly for using your browser's `find` command on and it's not pretty. Show me something simple and better and I'll use it.

20.6 genmenu - generate a collapsing menu

20.6.1 Usage:

```
%
genmenu version 1.5
usage: ../genmenu [options] pagename menufile
where options are:
-d enable debugging output
-h show this help
```

20.6.2 Description:

`genmenu` reads the menu description file and generates a vertical menu-bar, collapsed according to which pagename you gave it. This requires all the documentation to be rebuilt whenever you change the menu definition, but avoids having to use JavaScript.

I couldn't find a simple stand-alone program that did this, so here you are. It doesn't require remstats.

20.6.3 The Menu Definition File

The file has a simple format. Blank lines and lines beginning with '#' are ignored. The other lines look like:

```
%
[tabs]pagename [page title]
```

The number of `tabs` shows the level of sub-menus, making the definition file easy to grasp at a glance. Note that the `tabs` are actual tab characters. The `page title` (optional) is what shows up in the menu, while the `pagename` is used to make the URL to link to. If the page name is `xyz`, then a link to `xyz.html` is produced. If the `page title` is missing, then the `pagename` is used instead.

20.7 podhtml - convert HTML to POD

20.7.1 Usage:

```
%
htmlpod htmlfile $>$podfile
```

20.7.2 Description:

Quick and Dirty.

I only wrote it to do the first cut for converting my old html-based documentation to pod format. It's by no means complete or even correct. However, it did convert over 90% of the HTML markup that I was using.

Use it if you want, but don't complain about it without providing a patch to fix your complaint.

20.8 podhtml - translate a POD file to HTML

20.8.1 Usage:

```
%
podhtml version 1.5
usage: podhtml [options] podfile
where options are:
-d ddd  enable debugging output
-h      show this help
-s sss  use 'sss' as the suffix for html files [.html]
-u uuu  use 'uuu' as a URL prefix
```

20.8.2 Description:

See the docs for `pod2html`. The only changes that I made intentionally from how `pod2html` does things are:

- if a line looks blank it's treated as blank. I prefer to avoid surprises.
- I added a new `=exec` which executes a command line and inserts the output of stdout into the resulting HTML as a <PRE> section. This was so that I could get the latest usage message from programs inserted without having to run each program separately, save its output in a file, and manually insert the file into the POD file.
- I also caused it to append to a file called `podhtml--rawindex` for each =head1 and =head2, a URL for that page and section and the contents of that =headN. This is used by genindex (see the genindex section) to make an index.

I wrote this version in frustration with the way pod2html does links. Or doesn't. I could never tell without trying whether it would generate a link or not.

20.9 podlatex - translate a POD file to LaTeX

20.9.1 Usage:

```
%
podlatex version 1.4
usage: podlatex [options] podfile
where options are:
-d ddd  enable debugging output
-h      show this help
-s sss  use 'sss' as the suffix for html files [.html]
-u uuu  use 'uuu' as a URL prefix
```

20.9.2 Description:

See the docs for `pod2latex`. The only changes that I made intentionally from how `pod2latex` does things are:

- if a line looks blank it's treated as blank. I prefer to avoid surprises.
- I added a new `=exec` which executes a command line and inserts the output of stdout into the resulting HTML as a <PRE> section. This was so that I could get the latest usage message from programs inserted without having to run each program separately, save its output in a file, and manually insert the file into the POD file.
- I changed the text wrapping links.

I wrote this version in frustration with the way pod2latex does links.

20.10 podpdf - translate a POD file to pdf

20.10.1 Usage:

```
%

../podpdf
[ --help --verbose <1|2 --paper <usletter> --podfile <file> ] <file>

--help
    displays this explanation of correct usage

--vebose <1|2>
```

```
    regulates the volume of progress comments: argument must be 1 or 2

--podfile <file>
    supplies the input file to process as an explicit parameter. The
    input file may also be supplied from STDIN or from the command
    line as the array element --paper.

Further information can be found in the POD section of Pod.pm. Enter:
    perl -e "use Pod::Pdf; pod2pdf('<your_library_path>/Pod/Pdf.pm')"
to get the POD in PDF format :)
```

20.10.2 Description:

See the docs for `Pod::PDF`. This is only a tiny wrapper for it.

20.11 rrd-report - display summaries of an RRD file

20.11.1 Usage:

```
%
rrd-report version 1.6
usage: rrd-report [options] rrd-file
where options are:
-b bbb begin at time 'bbb' (see Note 3)
-c ccc  select data from consolidation-function 'ccc' [AVERAGE]
-d ddd  enable debugging output at level 'ddd'
-D DDD  show the dates as 'DDD' [both,pretty]
(none|both|start|finish),(raw|simple|pretty)
-e eee  end at time 'eee' (see Note 3)
-f fff  use report format 'fff' [simple]
(from 'simple', 'label', 'html')
-h      show this help
-i iii  report on intervals 'iii' (see Note 2) [1d]
-l      list the DS names in this rrd, no report
-n nnn  use 'nnn' as the format to print the numbers [%lf]
-s sss  summary on interval 'sss' (see Note 2) [1w]
-v vvv  show variables 'vvv' (var:cf comma-separated) [ALL]
Note: if report interval (-i) and summary interval (-s) are equal,
  no summary reporting is done.
Note 2: intervals are numbers of seconds, minutes, hours, days, weeks, Months
  or years.  E.G. "4w" for 4 weeks, "1M" for one month.
Note 3: Begin and End times are a unix timestamp (seconds since Jan 1, 1970) or
  plus-or-minus an interval, as in Note 2.  E.G. "-1w" means "one week ago".
```

20.11.2 Examples:

I hope that the above is enough to use it after seeing a few examples. Here's the equivalent of the command that created the RRD for the example.

rrdtool create ping.rrd \ DS:sent:GAUGE:600:0:10 \ DS:rcvd:GAUGE:600:0:10 \ DS:min:GAUGE:600:U:U \ DS:avg:GAUGE:600:U:U \ DS:max:GAUGE:600:U:U \ RRA:AVERAGE:0.1:1:600 \ RRA:AVERAGE:0.1:7:300 \ RRA:AVERAGE:0.1:30:300 \ RRA:AVERAGE:0.1:90:300 \ RRA:AVERAGE:0.1:365:300 \ RRA:MIN:0.1:1:600 \ RRA:MIN:0.1:7:300 \ RRA:MIN:0.1:30:300 \ RRA:MIN:0.1:90:300 \ RRA:MIN:0.1:365:300 \ RRA:MAX:0.1:1:600 \ RRA:MAX:0.1:7:300 \ RRA:MAX:0.1:30:300 \ RRA:MAX:0.1:90:300 \ RRA:MAX:0.1:365:300

See "man rrdcreate" for an explanation for the command itself. The fields are:

- sent/rcvd - number of ping packets sent/received
- min/avg/max - the round-trip-time (min, average and max) for the pings

Here's a default report from one of my ping RRDs:

```
%
\%rrd-report ping.rrd
[snip]
data 1999-10-25 17:20:49 1999-10-26 17:20:49 10.000000 10.000000 10.000000 9.864444 9.899444 9.934444 41.569556 42.210222 42.850889 45.626889 45.758833 45.890778 50.955111 51.253056 51.551000
data 1999-10-26 17:20:49 1999-10-27 17:20:49 10.000000 10.000000 10.000000 9.934444 9.987819 10.000000 39.536556 41.498103 46.124938 42.166222 45.386889 49.370000 50.955111 52.435532 54.635926
summary 1999-10-19 17:20:49 1999-10-26 17:20:49 10.000000 10.000000 10.000000 9.331111 9.932391 10.000000 38.317778 42.750146 48.318444 41.500556 46.736223 50.265778 49.122444 52.261605 59.867778
[snip]
data 1999-11-17 16:20:49 1999-11-18 16:20:49 10.000000 10.000000 10.000000 8.000000 9.934245 10.000000 36.400000 46.585421 50.000000 41.256667 49.592941 58.640000 49.036667 53.837716 117.240000
summary 1999-11-16 16:20:49 1999-11-18 16:20:49 10.000000 10.000000 10.000000 9.142857 9.929788 10.000000 38.285714 46.482773 50.000000 47.148095 49.503659 51.294286 50.000000 53.124615 65.880952
overall 1999-10-19 17:20:49 1999-11-18 16:20:49 9.978889 9.999918 10.000000 1.323333 9.876767 10.000000 6.194333 45.631272 76.394778 6.516556 48.842746 107.286556 6.971111 54.319770 179.554222
```

Each "data" line is a report for the interval covered by the two timestamps, (by default one day). The values are the requested (or in this case all) DS:CF combinations. The "summary" lines are just reports over a longer interval (by default one week). The "overall" line is for the whole selected time-period.

Hmm. There's much too much there. What I'd really like to see is just the interesting stuff. I know how many pings I'm sending during this period (10), so drop that and just show the minimum min average avg and maximum max:

```
%
\% rrd-report -v rcvd:AVERAGE,min:MIN,avg:AVERAGE,max:MAX
data 1999-10-19 17:54:57 1999-10-20 17:54:57 9.820267 38.317778 43.948411 55.716667
data 1999-10-20 17:54:57 1999-10-21 17:54:57 9.966716 39.303333 46.180111 59.867778
data 1999-10-21 17:54:57 1999-10-22 17:54:57 9.907440 40.469000 48.496274 56.022222
data 1999-10-22 17:54:57 1999-10-23 17:54:57 9.977827 40.232333 47.571133 54.475062
[snip]
summary 1999-11-09 16:54:57 1999-11-16 16:54:57 9.950836 39.310056 52.578943 179.554222
data 1999-11-17 16:54:57 1999-11-18 16:54:57 9.934164 36.400000 49.606736 117.240000
```

```
summary 1999-11-16 16:54:57 1999-11-18 16:54:57 9.928672 38.285714 49.489729 65.880952
overall 1999-10-19 17:54:57 1999-11-18 16:54:57 9.876767 6.194333 48.842746 179.554222
```

Well, I can figure out when the period ended, so leave out the end-time, and I don't like seeing all those meaningless (in this case) decimal places, so how about:

```
%
\% rrd-report -D start,pretty -n \%.1lf -v rcvd:AVERAGE,min:MIN,avg:AVERAGE,max:MAX
[snip]
data 1999-11-14 17:27:04 10.0 40.0 49.4 88.7
data 1999-11-15 17:27:04 9.7 21.7 48.0 63.4
data 1999-11-16 17:27:04 9.9 38.3 49.4 61.3
summary 1999-11-09 17:27:04 10.0 39.3 52.6 179.6
data 1999-11-17 17:27:04 9.9 36.4 49.6 117.2
summary 1999-11-16 17:27:04 9.9 38.3 49.5 65.9
overall 1999-10-19 18:27:04 9.9 6.2 48.8 179.6
```

OK. I'd like to see the last year with a one-week interval, with no summaries. (Setting the report-interval to the same as the summary-interval drops summaries. You still get an overall line.)

```
%
\% rrd-report -D start,pretty -n \%.1lf -v rcvd:AVERAGE,min:MIN,avg:AVERAGE,ma
x:MAX -i 1w -s 1w
data 1998-11-19 09:04:43 NODATA NODATA NODATA NODATA
[snip]
data 1999-02-25 09:04:43 9.9 45.0 55.0 64.2
data 1999-03-04 09:04:43 10.0 43.9 54.5 64.3
[snip]
data 1999-11-11 09:04:43 10.0 39.3 51.7 179.6
data 1999-11-18 09:04:43 9.9 37.4 49.7 169.7
overall 1998-11-19 09:04:43 9.6 0.0 45.3 103.7
```

And for those of us who like to see it on the web: [You'll just have to look in the web version of this documentation to see what it looks like.]

Chapter 21

Thank-you's

21.1 Thank-you's

- Tobias Oetiker - for MRTG and RRDtool
- Larry Wall and the rest of the perl hackers - for making perl into the "swiss-army-chainsaw" of programming languages
- Vikas Aggarwal - for multiping from NOCOL, so pinging doesn't take forever
- Ehud Gavron and others - for the NANOG (see **http://www.nanog.org/**) traceroute.
- Will Maton - numerous suggestions and encouragement
- Adam Kennedy - initial pre-release testing
- Ken Filipps - suggestion to brand remstats port-probes
- Andrew Cochran - for Telnet.pm open error
  - for non interface-MIB interfaces suggestion, e.g. frame-relay
- Marek Snowarski - for installation and documentation feedback
- Matt Duggan - for suggestions (views and description magic-cookies)
- Steve Francis - for suggestions and installation feedback
- Alexander Reelsen - for suggesting that the df-* rrd shouldn't use K-btytes
  - for the masqueraded connection count code
- Jon Villarreal -for pointing out the error in alert time selection

I've probably forgotten some others. Please remind me if I've forgotten your contribution.
default