NetInfo - network control & monitor system. (Version 0.0.4)

14th April 2005

Contents

1 Preface

Lately, the worldwide boom of network technologies has resulted in the new hardware and software projects, utilities and protocols for different data transmission jobs. More and more people become involved in this business. A small company or an institute can afford a local network with fast Internet access. That is why a problem of network control arises.

2 Version notes

Attention! Please report all encountered bugs, mistakes or your comments to netinfo-devel@lists.sycore.org
Netinfo-0.0.4 (current version) has new additional features:

1. Daemon for monitoring.
2. Support for the counters of lists (list objects).
3. Counters SMNP, SNMP (MSc-based) and EXEC.

Netinfo-0.0.3 version contains: formed kernel (list-link structure & lexical parser). Most of the operations are tested, however, the version could not be called 'stable' so far.

When writing a service reload script (service.sh), keep in mind that the program uses UNICODE (utf-8). In case of non-English console, first, drop the locale (export LC_ALL=C) or use the iconv utility, e.g. (Russian console) like this:

/etc/init.d/apache restart | iconv -f koi8-r -t utf8

3 Acknowledgments

I am deeply indebted to my scientific leader Mr. Kirill Yu. Bogachev for his help, advice and encouragement, which resulted in significant improvement of the product. I am also grateful to Alexei Pustynsev (English translation), Moscow State University and System Core Solutions team (http://en.sycore.org).

4 Purpose

This product intends as a multifunctional and universal software for network control, enabling the flexible change of network settings, services, individual network parts, computers and special equipment. As a result, we would like to get software that would provide the user-administrator at a remote computer with graphical web interface to control the network.

5 System overview

5.1 List-link structure, lists.

The system stores any information in the form of dynamic structure of tables (lists). The basic operations with tables like adding, deleting and editing a parameter/column are supported. Usually, the table has some columns of two types:

1. Typical column with fixed data (see below).
2. Link to another system table (column).

The lists of the table are grouped through the link-columns according to the lists of another table.

5.2 Lists and database

All list-link data are directly mapped into the relation database (PostgreSQL). When being created, the lists and their fields are given unique tags , which are actually the names used by the database for the corresponding lists or fields.

5.3 Data types.

The available data type scheme is complete and consistent. Any column is assigned one of the next types:

- Text - any sequence of symbols.
- Text area - the same as above, but displayed in a multi-line manner by the GUI.
- Integer number.
- Float number.
- Network address like: x.x.x.x/y, where y is a network mask (the number of non-zero bits on the left). The mask is void and, therefore, omitted if an ordinary ip-address is entered.
- Host IP-address.
- MAC-address - the 6-byte network computer address.
- Flag - field that has 2 values (on/off).

All entered data are being checked to comply with the data type of the column.

5.4 The control of network services.

One of the main features of the system is the creation of configuration files based on the information obtained from the lists and preliminary made templates. Netinfo has its own lexical parser, which can multiply the lines and blocks of the templates to create a real configuration file. Once written, the set of the templates makes the rewriting of the configuration file unnecessary whenever the system data changes.

6 User's guide.

6.1 Installation.

Netinfo requires the next packages:

1. Apache web-server (any version)
2. PHP-4 compiled with the following options: –with-zlib –with-dom –with-gd –with-psql –enable-xslt –with-sablot
3. PostgreSQL (7.3, 7.4)
4. Sablotron (xslt-parser)
5. libxml2
6. libgd

Netinfo has a GNU standard set of configuration and installation tools:

- –prefix - the system installation directory.
- –www-prefix - the installation directory of php scripts.
- –with-www-lang - web-interface language (en, ru).
- –with-apache-user - the user that runs the apache web server.

Attention! To make the system safer, all scripts are installed with minimum rights and owned by the user that is running Apache. The username is specified in httpd.conf as a value of 'User' directory.

After configuring do:

```
make
make install
```

When installation is over, you should set the database parameters in the menu "Setup->Database Setup".

6.2 Graphical user interface.

The graphical user interface (GUI) is available at:

http://<servername>/<path_to_netinfo_scripts_from_document_root>/index.php

Let's consider the short description of the GUI. The system has the primary /secondary menu and control panel. The basic menu items are:

1. Lists. Here is the main list of all system dynamic lists. The user can perform the following basic operations:
   - View list. ().
   - Add record. ().
   - Delete record. ().
   - Find record. (). The search for a record is done according to the value of fields.
   - Edit table (). The type of data entered must be the same as the type of field.
   - Add column (field) ().
   - Add link ().
2. Templates. Basic operations on the templates:
   - Add template ().
   - Delete template ().
   - Edit template ().
   - View configuration result ().
   - Rewrite configuration, reload services ().
3. Database setup. Each installed database has its name. If the database communicates with the client through a unix-socket one is to leave the field of ip-address/host blank.
4. List setup. Provides basic operations on lists. The user can add, edit or delete fields and lists, change the type of field (if possible), change the order of fields (), set default values and do other things.

6.3 The general principles of configuration and the syntax of template.

The system is controlled by modifying the configuration files. Normally, a template, which will be used for configuration of a given service according to the information from lists, is to be created for each service. Consider configuration of a concrete job. We assume that some computers, one of which is a router running the Netinfo system, form a local network (e.g. an office network) and we need a configuration file for the DNS service to be created automatically, depending on the variable data (records). First, we create a list of users with the tag"users" and the following fields:

- host (tag - 'host', type - 'hostname')
- ip-address (tag - 'ip', type - 'ip-address')
- mac-address (tag-'mac', type-'MAC-address')

You can add another information fields as well. Only these fields will be used for creating the DNS template.

A piece of template for the DNS configuration file is shown below:

```
; MACHINE-GENERATED FILE - DO NOT EDIT
$TTL            86400
@  IN  SOA      gw.rector.msu.su. root.gw.rector.msu.su. (
                2003070818      ; Serial
                21600           ; Refresh 6 hours
```

```
                3600          ; Retry   1 hour
                2592000       ; Expire  30 days
                86400 )       ; Minimum 24 hours
;
            IN      A       172.16.0.1
            IN      NS      gw.rector.msu.su.
;
            IN      MX      1   gw
;
gw          IN      MX      10  gw
;
mail        IN      MX      10  gw
mail        IN      A       172.16.0.1
;
ns          IN      CNAME   mail
smtp        IN      A       172.16.0.1
pop         IN      CNAME   smtp
mx          IN      A       172.16.0.1
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
%%SQL("SELECT host, ip FROM users;")%%   %%ROW(0)%%  IN    A    %%ROW(1)%%
```

The last line is of particular interest:

```
%%SQL("SELECT host, ip FROM users;")%%   %%ROW(0)%%  IN    A    %%ROW(1)%%
```

The keywords between '%%' are the template functions. Those lines of the generated configuration file that have no keywords remain unchanged. If a line has some keywords, it will be multiplied according to the amount of data being received, in the meantime, the data obtained from querying the database will be entered into every line. Here, the template tag %%SQL("SELECT host, ip FROM users;")%% tells the lexical parser to do the SQL-query "SELECT host, ip FROM users;" while the tags %%ROW(0)%% and %%ROW(1)%% of the line will be replaced by the first and second column data respectively. Depending on the 'users' list content, all said above may result in the following:

```
; MACHINE-GENERATED FILE - DO NOT EDIT
$TTL          86400
@  IN  SOA    gw.rector.msu.su. root.gw.rector.msu.su. (
                2003070818    ; Serial
                21600         ; Refresh 6 hours
                3600          ; Retry   1 hour
                2592000       ; Expire  30 days
                86400 )       ; Minimum 24 hours
;
            IN      A       172.16.0.1
            IN      NS      gw.rector.msu.su.
;
            IN      MX      1   gw
;
gw          IN      MX      10  gw
;
mail        IN      MX      10  gw
mail        IN      A       172.16.0.1
;
ns          IN      CNAME   mail
smtp        IN      A       172.16.0.1
pop         IN      CNAME   smtp
mx          IN      A       172.16.0.1
;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
gw        IN   A   172.16.0.1
user1     IN   A   172.16.0.2
user2     IN   A   172.16.0.3
user3     IN   A   172.16.0.4
```

We got the BIND configuration associated with the current table of users. When the template for the configuration file of the network service is done, the network administrator may forget everything about it since all the data related to adding a user, deleting a user or changing the settings of user will be automatically updated. Whenever the system is changed, it regenerates the configuration files and restarts the network services automatically (this can be done manually too). Let's see the tags, which are used by the lexical parser during the creation of configuration files, in detail.

6.4 Tags.

Tag is a keyword between the symbols '%%'. Block is a collection of lines between the symbols %%%. The writing of tag is case sensitive and has no spaces. However, this rule does not apply to the SQL-query. For each parameter, a line or a block containing a tag will be duplicated , while those lines that have no tags are unaltered, as was said above. For example, consider the following block.

```
%%%
host %%GetHost()%%
{
hardware ethernet %%GetMac()%%;
fixed-address %%GetIp()%%;
}
%%%
```

The lexical parser treats the block as a single unit so that it will be "multiplied" as a whole.

There are 4 tag groups, defined by the system. The other groups are defined by the user.

- %%SQL()%%, %%ROW(0)%%, %%ROW(1)%%, ... The argument of the tag is a double-quoted SQL-query to the PostgreSQL database. The result returned by the database will displace the tags %%ROW%%.
- %%define%%, %%SQL()%%, %%FuncName1()%%, %%FuncName2()%% - This is a definition of the user multi-line function. The SQL-query, which is defined as an argument of the tag %%SQL()%%, is "connected" to the functions %%FuncName1()%%, %%FuncName2()%%, etc. The names of the functions are defined by the user and can be used after the block, meanwhile, they will be replaced with the data from the associated columns of the SQL-query. For example,

  ```
  %%%
  %%define%%
  %%SQL("SELECT host, ip, mac FROM users;")%%
  %%GetHost()%%
  %%GetMac()%%
  %%GetIp()%%
  %%%

  %%GetHost()%%  IN   A   %%GetIp()%%
  ```

  Here, we define user tags %%GetHost()%%, %%GetIp()%% É %%GetMac()%%, which can be hereafter used without paying attention to the SQL-query. A line or a block that contains the tag "define" is not entered into the resulted configuratoin file. Obviously, the tags from different blocks will never be included into the same line.

  To get the user-defined tags (functions) better tuned, it is possible to use the functions with following arguments:

%%%
%%define%%
%%SQL("SELECT host, ip, mac FROM users WHERE ip <<= $1;")%%
%%GetHost($1)%%
%%GetMac($1)%%
%%GetIp($1)%%
%%%

%%GetIp(172.16.0.0/16)%% IN A %%GetMac(172.16.0.0/16)%%

This example shows three functions with the parameter '$1', which is a subnetwork address used by the SQL-query to restrict the field scanning so that the fields with the same IP addresses are scanned only.

- %%oldefine%%, %%SQL()%%, %%FuncName1()%%, %%FuncName2()%% - This is a definition of the single-line user function. The only difference between it and the multi-line function is that the single-line function uses just the first line (record) of the result returned by the SQL-query. A template line containing the single-line functions only, will become a line of the configuration file, into which the data returned by the SQL-query are entered, e.g.:

  %%%
  %%define%%
  %%SQL("SELECT ip FROM users WHERE host='$1';")%%
  %%GetUserIp($1)%%
  %%%

  ClientHost IN A %%GetUserIp(ClientHost)%%

  As a result, the tag %%GetUserIp(ClientHost)%% will be replaced with the ip-address of the host called 'ClientHost'. These tags can be used along with whatever tags and in arbitrary quantities because the result of the query is just a record .

- %%include()%%- This one inserts a template, which has a filename used as a double-quoted argument of the tag, e.g.:
  Template 'include_mould':

  %%%
  %%define%%
  %%SQL("SELECT ip FROM users WHERE host='$1';")%%
  %%GetUserIp($1)%%
  %%%

  Template 'BIND':

  %%include("include_mould")%%
  ClientHost IN A %%GetUserIp(ClientHost)%%

  Here, the lines are simply taken from the template containing the filename "include_mould" for further substitution. Usually, the included templates have the blocks "define" and "oldefine" only.

6.5 Built-in functions.

The information obtained from the lists and, hence, from the database may need to be modified sometimes. The use of PostgreSQL procedures written in plsql, perlpu or another programming language can be a simple way to achieve this. On the other hand, the built-in functions of the lexical parser, which deal with the multi-line tags only (as of version 0.0.3), can be used too:
[1]

- LastByte1 () - returns the last byte of the ip-address, the argument .
- LastByte2 () - the same but returns two bytes separated by ".".

[1] For developers: all built-in functions can be found in the file php/addon.php in $GLOBALS["addon"] array. You can simply write a built-in function of your own. A built-in function gets the value of $in and writes the result into the $out. If an error occurs, the function returns "false."

- LastByte2Rev() - the same two bytes separated by "." but reversed. These functions are useful for building BIND configuration (the reverse-zone configuration).

Example:

%%LastByte2Rev(GetIp())%% IN PTR %%AvGroupHost(eth1)%%.myhostname.org.

The future versions of the software are planned to have more built-in functions.

6.6 Configuration rewriting and service restarting.

1. The special program called "update" rewrites the configuration files created by the system and restarts the services.

The system performs the following actions during the restart of the services ( ) :

- The lexical parser scans all the templates and generates the configuration files, which are written into the ./result directory (this directory stores the templates). The scanning stops if a syntax or writing error occurs .
- The "update" program starts from the current directory of scripts. The program gets the root privileges and saves the old versions of the configuration files. The list of configuration files includes the files, the templates of which have their fileds "configuration file directory" filled in. In this case, the path of the configuration file is like this: <directory>/<configuration(template)filename>. If the old configuration file, which is specified by the given path, is missing, the program exits with an error, otherwise the old configuration file is overwritten with a newly created one.
- The last step is running the script "service.sh", which is started by the "update" program. This script has other scripts that restart the required services, e.g.:
  service.sh

```
#!/bin/sh
/etc/init.d/bind9 restart
/etc/init.d/rc.iptables restart
```

7 Authorization.

There is only one user in this version of the software. The user's login is "admin" (without quotation marks), the password is just an empty string. You can change the password in the menu" "Settings"->change password.

8 Statistical observation subsystem.

8.1 Overview.

Statistical observation subsystem is mostly like the well known monitor program "mrtg", however, the subsystem spotlights some functional and sructural advanatges over the "mtrg" and runs faster. The subsystem is actually a monitor daemon, which is started from command line once and, then, controlled via web The statistical dynamic data are displayed in the form of differently detailed graphs over different periods of time. :

- Daily statistical graph is graduated in 5-min units.
- Weekly statistical graph is graduated in 30-min units.
- Monthly statistical graph is graduated in 2-hour units.
- Yearly statistical graph is graduated in 1-day units.

Besides, the information about average, maximum and current values is provided too.

8.2 Main operation principles

The statistical observation subsystem is based on the notion of counter. The counter is a system object (in the list of counters) that makes the data collection and graph plotting rules for the other system objects

8.3 Counters and their parameters.

You can find the list of counters by going to the "Settings" ->"GUI statistics" menu. Let's see what the configuration of counter can be in a simple example of monitoring router interfaces. First, we create a list called [2] "Interfaces" with the fields of 'ifname' (the interface name) and 'description'. Next, we add interfaces (the full list of interfaces can be obtained by running the 'ifconfig' command), e.g. eth0, lo, eth1 etc. To collect statistical data, we use 'contrib/ifstat.pl' script that is included in standard Netinfo software (here, the path is shown from the root of the program).

The script receives one argument - the name of interface - as a parameter and returns two numbers: the values of counter regarding incoming and outgoing traffic. These numbers are associated with the quantities of bytes passed through this interface once it has been started. Let's detail the parameters of counter:

- The name of counter. This is purely an information field. In our case, we may want to use a name like "Interface traffic accounting".
- Colour 1(in), Colour 2(out). These are the colours used to get the needed legends drawn on the corresponding graph. Legend is a curve that reflects the change of a parameter value. Netinfo graphs can use no more than 2 legends. Obviously, legends of the same graph are to have the same units of measurement. For observation of the traffic, the legends, usually, have two meanings: incomig and outgoing data flow (in/out). We use standard default colors for our counter. Green (#00ff00) is for incoming and blue (#0000FF) is for outgoing traffic.
- The type of counter is the type of source that collects and supplies data for statistics. There are 3 types supported in this version: an application being started, SNMP and SNMP Mac-based. We use a perl script in our example that is why we set the type of counter to be EXEC (started application).
- The value of counter. There are two values of counter - difference and absolute. Netinfo can deal with whatever values in principle but the scripts and SMNP-agents that supply the data for the collector (see below) may well return both actual values of the parameter and the values of some internal counter. The interface traffic is a typical example. The ifstat.pl script outputs the number of bytes passed through the interface but we are more interested in the average speed of data flow passing through the interface. So, to have the flow transformed into the needed format we should use the difference value of counter. The following are the examples of absolute value of the counter: UPS voltage, fan rotation speed, the use of memory.
- The transformation defines the units of measurement obtained from a script or an SNMP-agent, the units of measurement on the graph and, also, relations between them all. In our case, this is Byte / 5min -> Byte / sec as the script scans the data once every 5 minutes.
- List is a collection of objects that should be counted. In our example, it is the list of interfaces (interfaces).
- The template of the parameters of counter (SQL-query). This query defines the parameters used for processing each list that corresponds to a given counter. The template is an sql-query, the first column of which is the list of oid-identifiers that belong to the objects being processed and the other columns have the parameters of statistical observation. The templates will be specified for the associated types of counter.

8.4 EXEC - counters (outside applications).

An applivation being started is a script or a program that returns two numbers, which are connected to the values of legends and divided by the new line symbols ("\n"). The lines containing the values must be followed by an empty line. Here is a perl code example:

```
print $val1."\n".$val2."\n\n";
```

The following template format is used to define the parameters of this type of counter:

[2] äÁÌÅÅ ÐÏ ÔÅËÓÔÕ × ÓËÏÂËÁÈ ÕËÁÚÙ×ÁÀÔÓÑ ÔÜÇÉ ÓÏÏÔ×ÅÔÓÔ×ÕÀÝÉÈ ÐÏÌÅÊ É ÓÐÉÓËÏ×.

SELECT oid, <script being started>, <argument 1>, <argument 2> ... FROM <the name of table-list of the counter> ...

Again, I would like to stress that the oid-column must be used as the first column for all counter types. EXEC-counter options:

<script being started> - full path to the script to be run.
<argument1>, <argument2>... - options passed to the script.

We have the following template in our interface accounting example, asuming that the ifstat.pl script has been installed into the directory /usr/local/libexec/netinfo :

SELECT oid, '/usr/local/libexec/netinfo/ifstat.pl', ifname FROM interfaces;

So, for each data collecting interface the script ifstat.pl has only one argument - the name of this interface - when the script is started.

8.5 SNMP - counters

SMNP-counters query the SNMP-agents to get the data needed. The template of the counter has the following format:

SELECT oid, <the ip-address of SNMP-agent>, <comunity>,
<In mib>, <Out mib>
FROM <the name of table-list of the counter> ...

Options:

- <the ip-address of SNMP-agent>
- Community - is a peculiar password to access SMNP-agent. Usually, it is 'public' .
- <In mib>, <Out mib> mibs for incoming and outgoing traffic respectively (generally, the data for legends 1 and 2).

The parameters of whatever counter are very convenient to set up by using SQL and extended language of Postgre SQL. Below, there is an example of querying a 3COM-smart switch, which represents an SNMP-agent, for its port statistics:

SELECT oid, '172.16.0.1', 'public',
'interfaces.ifTable.ifEntry.ifInOctets.'||(port + 100),
'interfaces.ifTable.ifEntry.ifOutOctets.'||(port + 100) FROM switchports;

The list 'switchports' containing the field 'port', which stands for the port number, is presumed to be available. The switch numbers the ports from 100: 100, 101, 102, while one is to use the normal order in the table and start numbering from zero.

8.6 SNMP MAC-based counters.

Some switches, 3COM in particular, transform MAC-addresses into decimal format to use it in mibs connected to the statistics of given hosts. The accounting for these agents can be done with an ordinary SNMP-counter, though, a transformation procedure may need to be written in an enhanced language like PerlPI, which may be unavailable in the standard software. SNMP MAC-based counters do the transformation themselves , using the following format:

SELECT oid, <the ip-address of SNMP-agent>, <community>,
<In mib's prefix>, <Out mib's prefix>, <MAC>
FROM <the name of table-list of the counter> ...

Comparing with the previous type of counter, one additional option is ready:

- <MAC> -MAC-address of the host being analysed.

While 'In mib's prefix' and 'Out mib's prefix' are the prefixes for incoming and outgoing mib, the full mib is obtained by 'glueing' the correspondent prefix and the decimal format of mac-address together.
Example:

SELECT oid, '172.16.0.1', 'public',
'16.4.2.1.6.65.6.', '16.4.2.1.7.65.6.', mac FROM users;

The list 'users' with the field 'mac' (the mac-address of network user) is assumed to be available.

8.7 Collector

Collector (aka monitor daemon) is a background process that collects and analyses incoming data. Once accurately setup in the menu "Settings->Database", it needs no restarting as new counters appear.

9 FAQ

1. Collector is running, the data, however, do not seem to be incoming - average, maximum and current values are zero .
   The counters are likely to be improperly configured. Start the collector by entering this command from the console:

   netinfo --log-level=debug --verbose

   The output of the command may help to find what is wrong. Note that the collector starts the first scan of the counters straight off, though, the next iterations follow the schedule of 5-min intervals, during which the program may seem to be "sleeping" (but, surely, not "freezing").

2. On my attempt to go through the link I get something like this:
   Sorry, the $result variable the reason is that XML parser error 5: unclosed token and the error code is 2
   This is most likely due to the usage of encodings other than UTF8, e.g. when the script service.sh outputs in Russian and koi8-r is applied.

3. How can I locate the directories into which the software is installed? Where are the configuration file and the dump file?
   You may use the output of the netinfo-config command:

   netinfo-config --all

   Note, also, that all scripts of EXEC-counters supplied with the software are placed in the directories the names of which can be obtained by entering this command:

   netinfo-config --script-dir

10 Useful links.

http://netinfo.sf.net - project site.
http://postgresql.org - PostgreSQL documentation, SQL syntax.
netinfo-devel@lists.sourceforge.net - mailing list.