NetInfo - система управления и мониторинга состояния сети. (Версия 0.0.3-pre2)

11 апреля 2005 г.

Содержание

1 Предисловие

В последнее время время в мире наблюдается расцвет сетевых технологий, создаются новые аппаратные средства, програмные комлексы, утилиты, протоколы, предназначенные для решения тех или и иных задач по передачи данных. Всё больше и больше людей с самой разной квалификацией оказываются задействованы в этой области. Локальная сеть со скоростным выходом в интернет сейчас вполне может позволить себе небольшое предприятие или институт. Поэтому возникает задача програмного управления сетевым оборудованием.

2 Замечания о версии

Внимание! Прошу сообщать о всех неточностях, ошибках, а так же высказывать любые пожелания по-поводу системы Netinfo на электронный адрес: netinfo-devel@lists.sycore.org

Версия Netinfo-0.0.4 (текущая). В версии появился демон мониторинга. Поддержка счётчиков для списков (обьектов из списков). SNMP, SNMP (MSc-based) и EXEC счётчики.

Версия Netinfo - 0.0.3. В версии в большей степени сформировано ядро системы (списочо-ссылочная структура + синтаксис лексического анализатора). Большинство операций протестировано, однако версию нельязя назвать стабильной.

В программе используется кодировка UNICODE (utf-8), следует иметь это в виду при написании скрипта перезапуска сервисов (service.sh). При использование русскоязычной консоли в кодировке koi8-r нужно либо предварительно сбросить локаль (export LC_ALL=C) либо использовать iconv например так

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

3 Цель работы

Цель данной работы - создание многофункционального универсального програмного средства управления локальной сетью, позволяющего гибко настраивать параметры сети, контролировать работу сетевых сервисов, отдельных сегментов сети, машин и специализированного оборудования. В итоге хотелось получить программный комплекс, предостовляющий пользователю-администратору на удалённой машине графический веб-интерфейс для управления сетью.

4 Общее представление об устройстве системы

4.1 Списочно-ссылочная структура, списки.

Любая информация в системе хранится в виде динамической структуры таблиц (списков). Поддерживаются базовые операции с таблицами: добавление / удаление / редактирование парамеров / редактирование столбцов. Таблица может состоять из некоторого количества столбцов двух типов:

1. Обычный типизированный - может содержать данные одного типа (см. далее).
2. Ссылка. Содержит ссылку на другую таблицу в системе.

С помощью ссылок-столбцов реализуется механизм групировки записей (строк таблицы) одной таблицы по записям другой. В целом списочно-ссылочная структура хранения данных позволяет удобным образом хранить, редактировать и отображать данные о системе.

4.2 Списки и база данных.

Вся информация, представленная в виде списков и связей между ними, имеет прямое отбражение на реляционную базу данных (PostgreSQL). Списки и поля списков имеют уникальные имена (назначаются тольковатем при создании), именуемые *тегом*. С точки зрения базы данных *тег* списка или поля - это соответственно имя таблицы или имя поля таблицы.

4.3 Типизация данных

В системе реализован механизм типизации, обеспечения целостности и непротиворечивост, любому столбцу назначается один из следующих типов:

- Текст - любая последовательность символов.
- Текстовое поле - то же, только при отображении в графической системе используется многострочное поле.
- Целое число.
- Число.
- Адрес сети - адрес вида: x.x.x.x/y, где y - маска сети (количество не нулевых бит слева). Может быть вырожденной (ip-адрес) - в этом случае маска не записывается.
- IP-адрес хоста.
- MAC-адрес - 6-ти байтный адрес машины в сети 802.3.
- Флаг - поле, имеющее два значения.

При любом вводе данных осуществляется проверка на корректность введённых данных в соответствии с типом столбца.

4.4 Механизм управлением конфигурацией сетевых сервисов

Озна из основных возможностей системы - построение конфигурации (конфигурационных файлов) на основе информации из списков и предварительно составленных шаблонов. Встроенный в систему лексический анализатор позволяет "размножать" строки и блоки шаблона, получая реальный конфигурационный файл. Составив один раз требуемый набор шаблонов, можно избавить себя от необходимости вручную переписывать конфигурацию при каждом изменении данных системы.

5 Руководство пользователя

5.1 Установка системы

Для установки Netinfo требуется наличие следующих пакетов:

1. Apache web-server (любой версии)
2. PHP-4, откомпилированный с следующими опциями: *–with-zlib –with-dom –with-gd –with-psql –enable-xslt –with-sablot*
3. PostgreSQL (7.4 и выше)
4. Sablotron (xslt-parser)
5. libxml2
6. libgd
7. ucd-snmp

Пакет Netinfo включает себя стандартный для GNU-проектов набор средств для конфигурирования и установки системы.

Опции скрипта *configure*:

- *–prefix* - каталог для установки системы.
- *–www-prefix* - каталог для установки php-скриптов.
- *–with-www-lang* - язык для web-интерфейса (en, ru).

- *–with-apache-user* - пользователь системы, под которым выполняется веб-сервер apache.

Внимание! Из моображений безопасности все скрипты устанавливаются с минимальными правами, их “владельцем” должен быть пользователь, из-под котрого запускается веб сервер apache или виртуальный сервер. Имя пользователя указано в директиве User конфигурационного файла apache.

После этого следует выполнить:

```
make
make install
```

После установки следует установить параметры БД в меню “Настройки->Настройки БД”

5.2 Графический интерфейс пользователя

Графический интерфейс доступен по адресу

```
http://<имя сервера>/<путь к каталогу скриптов от корня сервера>/index.php
```

Коротко рассмотрим его возможности. В системе реализовано первичное / вторичное меню и панель управления. Основные пункты меню:

1. **Списки.** В этом разделе представлены общий список динамических списков системы, предоставляются основные пользовательские операции над списками:
   - Показать список ().
   - Добавление записи ().
   - Удаление записи ().
   - Поиск записи по полям (). Поиск осуществляется в соответствии с типами, например для текстовых строк осуществляется поиск вхождений, для ip-адреса - вхождение в подсеть.
   - Редактировать таблицу (). Ввод должен осуществляться только в соответствии с типом.
   - Добавить столбец ().
   - Добавить ссылку ().
2. **Шаблоны.** Реализуются базовые операции над шаблонами:
   - Добавление шаблон ().
   - Удаление шаблон ().
   - Редактировать шаблон ().
   - Посмотреть конфигурацию ().
   - Обновить конфигурацию, перезапустить сервисы ().
3. **Настройки Базы Данных.** Имя БД - имя создаваемой при установке БД. В случае если БД работает с клиентом по unix-сокету следует оставить пустую строку в поле ip-адрес/хост.
4. **Настройки списков.** Предоставляет базовые операции над списками - добавление/удаление/редактирование а также смена типа (если возможно преобразование), изменение порядка следования полей (), установка значений по-умолчанию и прочее.

5.3 Общие принципы построения конфигурации.Синтаксис шаблонов.

Контроль сетевых сервисов в системе осуществляется с помощью модификации конфигурационных файлов. Для каждого сервиса предполагается построение шаблона, из которого впоследствии в соответствии с информацией в списках будет строится конфигурация для данного сервиса. Рассмотрим процесс построения конфигурации на конкретной задаче. Предположим, что у нас имеется локальная (например офисная) сеть из некоторого количества машин и машина-шлюз, на которой установлена система управления сетью Netinfo и мы хотим реализовать возможность автоматического построения конфигурационного файла для сервиса DNS на основе динамически меняющихся данных (записей). Для начала заведём список пользователей с тэгом *users* и следующими полями:

- host (тег - host, тип - 'имя машины')
- ip-адрес (тег - *ip*, тип - 'ip-адрес')
- mac-адрес (тег - *mac*, тип - 'MAC-адрес')

Кроме перечисленных полей Вы можете добавить любые другие поля, имеющие информационный характер. При составлении шаблонов мы будем использовать только эти поля.

Ниже приведён франмент шаблона для конфигурационного файла службы доменных имён BIND:

```
; 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)%%
```

Особый интерес представлет собой последняя строка шаблона :

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

Ключевые слова, заключённые в '%%' представляют собой функции шаблона. При генерации конфигурационного файла из шаблона строки, не содержащие ключевых слов будут оставлены без изменения. Если строка содержит ключевые слова, то строка будет "размножена" по количеству получаемых данных, при этом в каждую строку будут подставлены данные, полученные из запроса к базе данных. В данном случае тэг %%SQL("SELECT host, ip FROM users;")%% шаблона говорит лексическому анализатору о том, что для получения данных требуется выполнить запрос, указанный в скобках, а тэги %%ROW(0)%% и %%ROW(1)%% указывают на то, что данные из первой и второй колонок выборки следует поместить на указанные места в строке (вместо этих тегов).

Результат будет примерно следующий (в зависимости от содержимого списка users):

```
; 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
```

В результате мы получаем зависимость настроек BIND от текущей таблицы пользователей. Один раз построив шаблон для конфигурационного файла сетевого сервиса, сетевой администратор может забыть о нём: при добавлении новых пользователей, удалении пользователей, изменении параметров некоторых пользователей - всё это будет автоматически отображаться в конфигурационных файлах. При любых изменениях система (автоматически или вручную) произведёт перегенерацию файлов конфигурации и перезапустит сетевые сервисы. Рассмотрим более подробно теги, используемые лексическим анализатором при построении конфигурационных файлов.

5.4 Тэги

Тэг шаблона представляет собой ключевое слово, заключённое в символы %%. Тэг чуствителен к регистру букв и не должен содержать пробелов. Тэг может рассматриваться в контексте строки или блока. В первом случае для каждого параметра дублируется строка, содержащая тэг, во втором - блок - совокупность строк, заключённых в символы %%%; строки, не содержащие тэги остаются без изменения, например:

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

В данном случае весь блок будет рассматриваться лексическим анализатором как единый, “размножаться” будет весь блок целиком.

Существует 4 группы тэгов, определённых в системе, все остальные определяются пользователем.

- **%%SQL()%%, %%ROW(0)%%, %%ROW(1)%%, ...** В качестве аргумента (в двойных кавычках) тег получает SQL-запрос к БД Postgres, результат будет подставляться вместо тегов %%ROW%%

- **%%define%%, %%SQL()%%, %%FuncName1()%%, %%FuncName2()%%** - определение пользовательской многострочной функции. SQL-запрос, указанный в качестве аргумента тега %%SQL()%% будет “привязан” к функциям %%FuncName1()%%, %%FuncName2()%% и т.д. Имена функций задаются пользователем и могут быть использованы после этого блока, при этом они будут заменятся значениями из соответствующих столбцов SQL-запроса, например:

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

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

В данном случае мы определяем пользовательские функции-теги %%GetHost()%%, %%GetIp()%% и %%GetMac()%%, которые можем использовать в дальнейшем, забыв про sql-запрос. Строка(блок) содержащая тег define не переходит в результирующий конфигурационный файл. Очевидно, что в одной строке не могут встречаться теги из разных define-блоков.
Для более гибкого построения пользовательских тегов-функций можно использовать функции с аргументами, например:

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

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

примере определены функции с параметром $1, имеющего смысл адреса подсети - SQL-запрос произведёт выборку только тех полей, у которых ip-адрес будет принадлежать указанной подсети.

- **%%oldefine%%, %%SQL()%%, %%FuncName1()%%, %%FuncName2()%%** - one-line definintion - определение однострочной пользовательской функции-тега. Отличается от многострочной только тем, что использует из результата, возвращаемого SQL-запросом, только первую строку (запись). В результирующей конфигурации строка шаблона, содержащая только однострочные функции перейдёт в одну строку с подставленными значениями из SQL-запроса, например:

```
%%%
%%define%%
%%SQL(“SELECT ip FROM users WHERE host='$1';”)%%
%%GetUserIp($1)%%                                                 В
%%%

ClientHost IN    A    %%GetUserIp(ClientHost)%%
```

результате будет найден и подставлен ip-адрес хоста с именем 'ClientHost'. Посольку результат выборки представляет собой одну запись, данные теги могут быть использованы в любом количестве с любыми другими тегами.

- **%%include()%%**- вствляет шаблон с именем файла, указанным в качестве аргумента тега в двойных кавычках, например:

Шаблон include_mould:

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

Произв

Шаблон BIND:

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

простая подстановка строк из шаблона с именем файла "include_mould". Как правило включаемые шаблоны содержат только блоки define и oldefine.

5.5 Встроенные функции.

Иногда возникает необходимость каким-либо образом изменять информацию, получаемую из списков (а, следовательно, из базы данных). Одним из простейших способов - это использование хранимых процедур БД postgres, написаных на каком-либо языке (plsql, perlpu и т.д.).

С другой стороны лексический анализатор предоставляет ряд встроенных функций, которые (в версии 0.0.3) могут использоваться только с многострочными тегами:[1]

- **LastByte1 ()** - возвращает последний байт ip-адреса, аргумент - ip-адрес.
- **LastByte2 ()** - то же, только два байта, разделённых символом ".".
- **LastByte2Rev()** - два байта в обратном порядке, разделённых символом ".". Эти функции полезны при построении файлов конфигурации BIND (конфигурация обратной зоны).

Пример:

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

В дальнейших версиях планируется расширить набор встроенных функций.

5.6 Перезапись конфигурации и перезапуск сервисов.

Перезаписыванием порождённых системой конфигурационных файлов и перезапуском сервисов в системе занимается отдельная программа update. При выполнении процедуры перезагрузки () происходит следующее:

- Лексический анализатор призводит разбор **всех** шаблонов и порождает конфигурационные файлы, которые записываются в директорию ./result (шаблоны хранятся в директории ./moulds). В случае ошибки синтаксиса или записи процесс останавливается.
- Запускается программа update из текущего каталога скриптов. Программа получает привелегии пользователя root и начинает сохранять старые версии конфигурационных файлов. В список перезаписываемых файлов конфигурации входят те, шаблоны которых имеют заполненное поле "Директория конфигурационного файла" , в этом случае путь файла конфигурации имеет вид: <Директория конфигурационного файла>/<имя шаблона>. В случае если старый конфигурационный файл по данному пути не обнаружен выполнение программы заканчивается с ошибой. В случае успеха старый конфигурационный файл переписывается новым (порождённым).
- В качестве завершающего этапа программа update запускает скрипт service.sh, в котором должны находится скрипты, перезапускающие требуемые сервисы, например:
servise.sh

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

[1] Для разработчиков: все встроенные функции описаны в файле php/addon.php в массиве $GLOBALS["addon"], не составит никакого труда при необходимости дописать свою встроенную функцию, получающую значение $in и записывающий результат по ссылке $out, в случае ошибки возвращается false, что приводит к остановке работы лексического анализатора.

Существует также возможность раздельного перезапуска сервисов - для этого существует иконка ( ) около каждого шаблона. В случае перезапуска сервиса по этой ссылке скрипт service.sh запускается с опциями, указанными в параметрах шаблона. При этом перезаписываются этот шаблон и шаблоны, имеющие такие же опции.

6 Авторизация.

В данной версии предусмотрен только один пользователь системы, login - admin, пароль после установки - пустая строка, сменить пароль можно в меню “Настройки”->сменить пароль

7 Подсистема сбора статистической информации.

7.1 Обзор.

Подсистема сбора данных статистики во многом напоминает широко используемую систему мониторинга mrtg, однако имеет ряд как функциональных, так и структурных преимуществ, кроме того превосходит последнюю по скорости работы.Подсистема состоит из демона мониторинга, который запускается с командной строки один раз и в дальнейшем управляется через web. Динамическая информация представляется в виде графиков статистики за разные промежутки времени и с разной степенью детализации:

- Дневной график - с детализацией по 5 мин.
- Недельный график - с детализацией по 30 минут.
- Месячный график - с детализацией по 2 часам.
- Годовой график -с детализацией по 1-му дню.

Кроме того предоставляется информация о среднем, максимальном и текущем значении.

7.2 Основные принципы работы.

В основу подсистемы сбора статистической информации положено понятие счётчика. *Счётчик* - это обьект в системе (в списке счётчиков), котрый задаёт правила сбора данных для других обьектов и представления графиков для них. Счётчик может быть привязан только к конкретному списку обьектов. Каждому счётчику назначается шаблон (sql-запрос), в соответствии с которым будет выполнятся обсчёт конкретных обьектов. Демон мониторинга (он же *коллектор*) собирает данные статистики, основываясь на параметрах счётчика и параметрах обьектов. Для этого он раз в пять минут инициирует процедуру опроса всех счётчиков, забор данных, их сохранение и упаковку усреднённых данных в вышестоящие таблицы (с меньшим уровнем детализации).За счёт упаковки осуществляется сохранение размеров хранимых данных для счётчиков без уменьшения значимости самих данных - за больший период требуется меньшая степень детализации.

7.3 Счётчики, параметры счётчиков.

Список счётчиков можно найти в меню настройки->статистика графического интерфейса. Приведём пример конфигурации счётчика. Поставим перед собой простую цель - мониторинг интерфейсов машины-шлюза. Создадим список[2] 'Интерфейсы' (interfaces) с полями 'имя интерфейса' (ifname) и описание ('description'). Добавим интерфейсы (список интерфейсов можно получить командой ifconfig), например: eth0, lo, eth1 и т.д. Для сбора статистики воспользуемся имеющимся в стандартной поставке Netinfo скриптом contrib/ifstat.pl (указан путь относительно корня пакета). Этот скрипт получает в качестве параметра один аргумент - имя интерфейса и выдаёт два числа - значение счётчика для входящего и исходящего траффика в байтах (эти числа соответствуют количеству байт прошедших через этот интерфейс с момента его старта).Рассмотрим более подробно параметры счётчика:

- **Имя счётчика, Описание** - сугубо информационные поля. Для нашего счётчика можно, к примеру, использовать имя ”Обсчёт траффика на интерфейсе”.

[2]Далее по тексту в скобках указываются тэги соответствующих полей и списков.

- **Цвет 1 (in), Цвет 2(out)** - цвета для отображения на графике соответствующих легенд. *Легенда* - это кривая, отображающая изменение значения одного параметра. В графиках Netinfo могут использоваться до двух легенд. Легенды одного графика по понятным причинам должны быть соизмеримы, т.е. иметь одинаковые единицы измерения. В случае измерения траффика легенды как правило имеют смысл входящего и исходящего потока (in/out).Для нашего счётчика будем использовать значения по-умолчанию, являющиеся стандартом: входящий трафик - зелёный цвет (#00FF00), исходящий - синий (#0000FF).

- **Тип счётчика** - тип источника, получающего данные для статистики. В данной версии поддерживается 3 типа: запускаемое приложение, SNMP и SNMP Mac-based. В нашем примере мы используем скрипт на perl, поэтому устанавливаем значение EXEC (запускаемое приложение).

- **Тип значения.** Существуют два типа значения счётчика - разностный (дифференциальный) и абсолютный. Netinfo в принципе может аккумулировать значения абсолютно любого рода, но скрипты и SNMP-агенты, выдающие данные для коллектора, могут давать как реальные значения параметра, так и значения некоторого внутреннего (накопительного) счётчика. Типичный пример - траффик на интерфейсе : результатом работы скрипта ifstat.pl является количество байт, прошедших через интерфейс, в то время как нас интересует средняя скорость прохождения данных через интерфейс. Для автоматического преобразования в требуемый формат (скорость) следует использовать дифференциальный тип счётчика. Примеры счётчиков с абсолютными типами значений: напряжение питания UPS, скорость оборотов кулера, расходование памяти.

- **Преобразование** -определяет единицы измерения значения, получаемые от скрипта или SNMP - агента и единицы измерения на графике, а также преобразования между ними. В нашем случае это будет: Byte / 5min -> Byte / sec, ввиду того, что скрипт опрашивает данные раз в пять минут.

- **Список** - список, обьекты которого подлежат “обсчёту”. В нашем примере это будет список интерфейсы (interfaces).

- **Шаблон праметров счётчика (SQL - запрос).** В соответствии с этим запросом определютс параметры обсчёта для каждого обьекта списка, которому соответствует данный счётчик. Шаблон - это sql-запрос, первой колонкой которого является список oid-идентификаторов обсчитываемых обьектов, а в последующих находятся параметры сбора статистики. Шаблоны будут более подробно рассмотрены для соответствующих типов счётчиков.

7.4 EXEC - счётчики (внешнее приложение).

Запускаечмое приложение - это скрипт или программа, выдающая два численных значения, соответствующих значениям легенд и разделённых символами перевода строки (“\n”). После строк, содержащих значения, должна следовать пустая строка. Пример кода на perl:

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

Для определения параметров этого типа счётчика используется шаблон следующего формата:

```
SELECT oid, <запускаемый скрипт>, <аргумент 1>, <аргу-
мент 2> ... FROM <имя таблицы-списка счётчика> ...
```

Ещё раз замечу, что использование в качестве первого столбца столбца oid обязательно для всех шаблонов всех типов счётчиков. Опции EXEC-счётчика:

- **<запускаемый скрипт>** - полный путь к скрипту для запуска.

- **<аргумент1>, <аргумент2>...** - опции, передаваемые скрипту.

В нашем примере обсчёта интерфейсов имеем следующий шаблон (Предполагаем, что скрипт ifstat.pl был установлен в каталог /usr/local/libexec/netinfo):

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

Таким образом для каждого отдельного интерфейса для сбора данных будет запускаться скрипт ifstat.pl с одним аргументом - именем этого интерфейса.

7.5 SNMP - счётчики

SNMP-счётчики для получения данных инициируют опрос SNMP-агентов. Шаблон счётчика имеет следующий формат:

```
SELECT oid, <ip-адрес SNMP-агента>, <comunity>,
<In mib>, <Out mib>
FROM <имя таблицы-списка счётчика> ...
```

Опции:

- **<ip-адрес SNMP-агента>**
- **Community** - своеобразный пароль на доступ к SNMP-агенту. Как правило 'public'.
- **<In mib>, <Out mib>** - mibs для входящего и исходящего траффка соответственно (В общем случае данные для легенд 1 и 2)

Параметры любого счётчика очень удобно составлять, пользуясь возможностями SQL и расширенного языка Postgre SQL. Ниже приведён пример опроса статистики портов интеллектуального коммутатора 3COM, выступающего в роли SNMP-агента:

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

Предполагается наличие списка switchports с полем port - номер порта. Сам коммутатор нумерует порты начиная со 100: 100, 101, 102, в то время как в таблице следует использовать нормальный порядок портов (начиная с нуля).

7.6 SNMP MAC-based счётчики.

Некоторые коммутаторы (в частности от 3COM) используют преобразование MAC-адреса в десятичный фомат для его использования в mibs, соответствующим статистике данных хостов. Обсчёт таких агентов может быть произведён с помощью обычного SNMP-счётчика, однако для этого потребуется написание процедуры преобразования на расширенном языке (вроде PerlPl), который может не входить в стандартную поставку. SNMP MAC-based счётчики сами выполняют это преобразование, их формат шаблона следующий:

```
SELECT oid, <ip-адрес SNMP-агента>, <comunity>,
<In mib's prefix>, <Out mib's prefix>, <MAC>
FROM <имя таблицы-списка счётчика> ...
```

По сравнению с предыдущим типом счётчика имеется одна дополнительная опция:

- **<MAC>** - MAC-адрес обсчитываемого хоста.

В то время как In mib's prefix и Out mib's prefix - это префиксы для входящего и исходящего mib, полный mib будет получен "склеиванием" соответствующего префикса и mac-адреса, представленного в десятичном виде.

Пример:

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

Предполагается наличие списка users с полем mac - mac-адрес машины пользователя в сети.

7.7 Коллектор

Коллектор (он же *демон мониторинга*) - фоновый процесс, занимающийся сбором и обработкой поступающих данных. Требует только верных настроек для базы данных, устанавливаемых в меню "Настройки->База Данных". После этого нет необходимости перезапускать его для новых счётчиков.

8 FAQ

1. *Коллектор запущен, однако данные не собираются - средние, максимальные и текущие значения равны нулю.*
   Скорее всего неправильно сконфигурированы счётчики. Запустите коллектор в консольном режиме командой:

   ```
   netinfo --log-level=debug --verbose
   ```

   Вывод может помочь найти ошибку. Учтите, что первый цикл обработки счётчиков коллектор запускает сразу, а дальнейшие попытки итерации выравниваются на каждые 5-минутные интервалы, в течении которых программа "засыпает" (но никак не "зависает").

2. *При попытке перейти по ссылке я получаю примерно следующий результат:*
   Sorry, the $result variable the reason is that XML parser error 5: unclosed token and the error code is 2
   Наиболее вероятный вариант - использование "чужой" кодировки, не UTF8, например в случае вывода скрипта service.sh на русском языке в кодировке koi8-r.

3. *Где я могу узнать пути установки дистрибудива, где находится конфигурационный файл и дамп-файл?*
   Можно воспользоваться выводом команды netinfo-config:

   ```
   netinfo-config --all
   ```

   Следует также иметь ввиду, что все скрипты EXEC-счётчиков (которые поставляются с дистрибутивом) находятся в каталоге, указанном в выводе команды:

   ```
   netinfo-config --script-dir
   ```

9 Полезные ссылки.

http://netinfo.sf.net - официальный сайт проекта.
http://postgresql.org - документация БД PostgreSQL, синтаксис языка SQL.
netinfo-devel@lists.sycore.org - mailing list.