Важно понимать, что если у кого-то есть доступ к какому-либо из ваших SNMP-устройств на чтение и запись, то при помощи SNMP он может взять эти устройства под контроль (например, настроить интерфейсы маршрутизатора, отключить порты или даже изменить таблицы маршрутизации). Один из способов защиты строк community – использование виртуальных частных сетей (VPN – Virtual Private Network) для обеспечения шифрования сетевого трафика. Другой способ – часто менять строки community. Для небольшой сети смена строк community не представляет трудности, но для сети, охватывающей целые кварталы и города и имеющей десятки (сотни или тысячи) управляемых узлов, смена строк community может стать проблемой. В качестве легкого решения можно написать простой Perl-сценарий, использующий SNMP для изменения строк community на ваших устройствах.

Структура информации для управления

До сих пор для обозначения рабочих параметров устройств, поддерживающих SNMP, мы пользовались термином *информация для управления*. Однако мы очень мало сказали о том, из чего на самом деле состоит информация для управления или как она представлена. Первый шаг к пониманию того, какую информацию устройство может предоставить, – это разобраться в том, как в контексте SNMP представляются данные. В стандарте «The Structure of Management Information Version 1» (Структура информации для управления) (SMIv1, RFC 1155) сделано именно это: он точно определяет, как управляемым объектам[1] присваиваются имена, и указывает связанные с ними типы данных. В стандарте «The Structure of Management Information Version 2» (SMIv2, RFC 2578) представлены дополнения для SNMPv2. Мы начнем с рассмотрения SMIv1, а в следующем разделе обсудим SMIv2[2].

Определение управляемых объектов может содержать три атрибута:

Имя

Имя, или идентификатор объекта (OID – Object Identifier), уникально определяет управляемый объект. Имена обычно бывают двух видов – цифровые и «удобочитаемые». В любом случае имена получаются длинные и неудобные. В SNMP-приложениях делается очень много работы для того, чтобы помочь вам удобно ориентироваться в пространстве имен.

---

[1] В оставшейся части этой книги мы будем говорить об *информации для управления* как об *управляемых объектах* (во множественном числе). Аналогично конкретная информация для управления (например, рабочее состояние интерфейса маршрутизатора) будет называться *управляемым объектом* (в единственном числе).

[2] Следует отметить, что используемая версия SMI не связана с используемой версией SNMP.

Тип и синтаксис

Тип данных управляемого объекта определяется при помощи части языка ASN.1 (Abstract Syntax Notation One). ASN.1 – это способ указать, как данные представляются и передаются между менеджерами и агентами в контексте SNMP. Преимущество ASN.1 заключается в том, что он независим от машины. Это означает, что ПК с ОС Windows 2000 может связываться с машиной Sun SPARC и не беспокоиться о таких вещах, как порядок байтов.

Кодировка

Конкретный управляемый объект кодируется в строку октетов с использованием правил «Basic Encoding Rules» (BER стандарта ASN.1). Правила BER определяют, как объекты кодируются и декодируются, чтобы их можно было транспортировать по среде передачи, например Ethernet.

Задание имен OID

Управляемые объекты организованы в древовидную иерархию. Эта структура является основой схемы присвоения имен в SNMP. Идентификатор объекта состоит из ряда целых чисел, определяемых узлами дерева и разделенных точками. Несмотря на то что есть удобочитаемая форма, более дружественная, чем строка чисел, эта форма представляет собой не что иное как ряд имен, разделенных точками, каждое из которых представляет узел дерева. Вы можете пользоваться самими числами или последовательностью имен, представляющих числа. На рис. 2.2 изображено несколько верхних уровней дерева (мы намеренно исключили некоторые ветви дерева, которые здесь для нас не важны).

В дереве объектов самый верхний узел называется корнем; все, из чего исходят дочерние узлы, называется субдеревом[1]; а все узлы, которые не имеют дочерних, называются концевыми узлами (листьями). Например, на рис. 2.2 начальная точка дерева называется корневым узлом. Его субдерево состоит из *ccitt(0)*, *iso(1)*, и *joint(2)*. На данной иллюстрации *iso(1)* – единственный узел, содержащий субдерево; оба других узла являются концевыми. *ccitt(0)* и *joint(2)* не имеют отношения к SNMP, поэтому не будут обсуждаться в этой книге[2].

Далее в этой книге мы сосредоточимся на субдереве *iso(1).org(3).dod(6).internet(1)*, которое в форме OID представляется как *1.3.6.1* или *iso.org.dod.internet*. У каждого управляемого объекта есть цифровой идентифи-

---

[1] Обратите внимание, что термин *ветвь* иногда используется как взаимозаменяемый с термином *субдерево*.

[2] Субдерево *ccitt* администрируется Консультативным комитетом по международной телефонной и телеграфной связи (CCITT); субдерево *joint* администрируется совместно ISO и CCITT. Как мы уже сказали, ни одна из ветвей не имеет ничего общего с SNMP.

катор OID и соответствующее текстовое имя. Обозначение в виде разделенных точками чисел используется для представления управляемого объекта внутри агента; текстовое имя, как доменное имя, соответствующее IP-адресу, избавляет людей от необходимости запоминать длинные, сложные строки чисел.

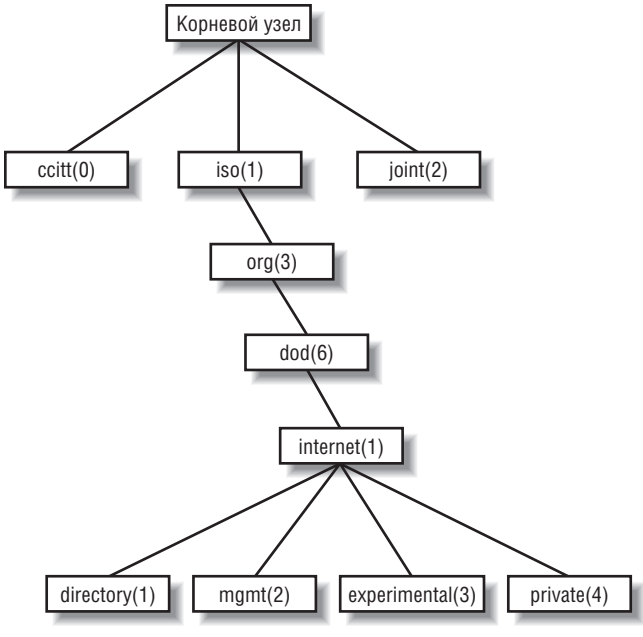

Рис. 2.2. *Дерево объектов SMI*

Ветвь *directory* в настоящее время не используется. Ветвь *management*, или *mgmt*, определяет стандартный набор управляемых объектов Интернета. Ветвь *experimental* зарезервирована для целей тестирования и исследования. Объекты ветви *private* определяются в одностороннем порядке, то есть за определение объектов этой ветви люди и организации отвечают сами. Вот определение субдерева *internet*, а также четырех его субдеревьев:

```
internet      OBJECT IDENTIFIER ::= { iso org(3) dod(6) 1 }
directory     OBJECT IDENTIFIER ::= { internet 1 }
mgmt          OBJECT IDENTIFIER ::= { internet 2 }
experimental  OBJECT IDENTIFIER ::= { internet 3 }
private       OBJECT IDENTIFIER ::= { internet 4 }
```

Первая строка описывает *internet* как OID *1.3.6.1*, который определяется (::= – это оператор определения) как субдерево *iso.org.dod*, или *1.3.6*. Следующие четыре определения похожи, но они относятся к другим ветвям, принадлежащим *internet*. Для ветви *directory* определение

`{ internet 1 }` показывает нам, что она является частью субдерева *internet* и что ее OID – *1.3.6.1.1*. Для *mgmt* OID – *1.3.6.1.2* и т. д.

В настоящее время в субдереве *private* есть одна ветвь. Она используется для того, чтобы предоставить производителям аппаратного и программного обеспечения возможность определить свои собственные частные объекты для любого типа аппаратных или программных средств, которыми они хотят управлять при помощи SNMP. Ее определение в SMI следующее:

```
enterprises   OBJECT IDENTIFIER ::= { private 1 }
```

За все назначения номеров для частных структур физических лиц, учреждений, организаций, компаний и т. д.[1] сейчас отвечает IANA (Internet Assigned Numbers Authority). Список всех номеров частных структур можно найти по адресу *http://www.iana.org/assignments/enterprise-numbers*. Например, номер частной структуры Cisco Systems – 9, поэтому базовый OID для ее частного пространства объектов определен как *iso.org.dod.internet.private.enterprises.cisco*, или *1.3.6.1.4.1.9*. В рамках этой частной ветви Cisco вправе действовать по своему усмотрению. Такие компании, как Cisco, производящие сетевое оборудование, часто определяют объекты в своих частных структурах. Это позволяет обогатить информацию для управления, которую можно получить из стандартного набора управляемых объектов, определенного в ветви *mgmt*.

Регистрировать собственные номера частных структур могут не только компании. Это может сделать любой, и это бесплатно. Веб-форма для регистрации номеров частных структур находится на сайте *http://www.iana.org/cgi-bin/enterprise.pl*. После заполнения формы, в которой запрашивается такая информация, как название вашей организации и контактные данные, ваш запрос должен быть одобрен приблизительно в течение недели. Зачем вам может потребоваться зарегистрировать свой номер? Когда вы лучше ознакомитесь с SNMP, вы столкнетесь с тем, что параметры, которые вы хотите отслеживать, не входят ни в одну MIB, общую или частную. Со своим собственным номером структуры вы сможете создать свою частную MIB, которая позволит вам отслеживать именно то, что вы хотите. Вам потребуется проявить изобретательность для улучшения своих агентов, чтобы они смогли найти нужную вам информацию, но выполнить это вполне реально.

Определение OID

Атрибут `SYNTAX` обеспечивает определение управляемых объектов при помощи части языка ASN.1. В SMIv1 определено несколько типов данных, важнейших для управления сетями и сетевыми устройствами. Следует иметь в виду, что эти типы данных – просто способ определить, какой

[1] В этой книге термин *частная структура* будет использоваться для обозначения ветви *enterprises*.

вид информации может содержать управляемый объект. Рассматриваемые нами типы аналогичны тем, которые вы можете встретить в компьютерном языке программирования, таком как C. В табл. 2.1 перечислены поддерживаемые типы данных для SMIv1.

Таблица 2.1. Типы данных SMIv1

Тип данных	Описание
INTEGER	32-битное число, используемое для обозначения перечислимых типов данных в контексте одного управляемого объекта. Например, рабочий статус интерфейса маршрутизатора может быть исправен, неисправен или тестируется. Для перечислимых типов 1 обозначает исправность, 2 – неисправность, а 3 – тестирование. Нулевое значение (0) не должно использоваться в перечислимом типе в соответствии с RFC 1155.
OCTET STRING	Строка из нуля и более октетов (более известных как байты), обычно используемая для представления текстовых строк, а иногда и для представления физических адресов.
Counter	32-битное число с минимальным значением 0 и максимальным значением $2^{32} - 1$ (4 294 967 295). При достижении максимального значения счетчик снова возвращается к нулю и стартует заново. В основном он используется для отслеживания такой информации, как количество отправленных и полученных интерфейсом октетов или количество ошибок и отклонений пакетов, зафиксированных на интерфейсе. Counter монотонно возрастает в том смысле, что его значения при нормальной работе никогда не уменьшаются. При перезагрузке агента все значения Counter должны быть установлены в ноль. Чтобы определить, можно ли сказать что-то полезное о последовательных запросах значений Counter, используют разность. Для расчета разности значение Counter запрашивается как минимум два раза подряд и берется разность между результатами запросов, произведенных через некоторый временной интервал.
OBJECT IDENTIFIER	Строка разделенных точками чисел, представляющая управляемый объект в дереве объектов. Например, *1.3.6.1.4.1.9* представляет OID частной структуры Cisco Systems.
NULL	В настоящее время не используется в SNMP
SEQUENCE	Определяет списки, содержащие ноль или более других типов данных ASN.1.
SEQUENCE OF	Определяет управляемый объект, состоящий из SEQUENCE (последовательности) типов данных ASN.1.
IpAddress	Представляет 32-битный адрес IPv4. 128-битные адреса IPv6 не рассматриваются ни в SMIv1, ни в SMIv2.
NetworkAddress	То же самое, что и тип IpAddress, но может представлять различные типы сетевых адресов.

Тип данных	Описание
Gauge	32-битное число с минимальным значением 0 и максимальным значением $2^{32} - 1$ (4 294 967 295). В отличие от Counter, Gauge может произвольно увеличиваться и уменьшаться, но никогда не может превзойти максимального значения. Скорость интерфейса маршрутизатора измеряется типом Gauge.
TimeTicks	32-битное число с минимальным значением 0 и максимальным значением $2^{32} - 1$ (4 294 967 295). TimeTicks измеряет время в сотых долях секунды. При помощи этого типа данных измеряется время работы устройства.
Opaque	Позволяет занести любую другую кодировку ASN.1 в OCTET STRING.

Цель всех этих типов объектов – определять управляемые объекты. В главе 1 мы сказали, что MIB – это логическая группировка объектов в соответствии с их принадлежностью к конкретной задаче управления, производителю и т. д. MIB можно рассматривать как спецификацию, которая определяет управляемые объекты, поддерживаемые производителем или устройством. Например, у Cisco определены буквально сотни MIB для ее широкого ассортимента продукции. В частности, устройство Catalyst и маршрутизатор серии 7000 имеют разные MIB. У обоих устройств есть особые характеристики, требующие специальных возможностей по управлению. MIB производителей обычно распространяются в виде удобочитаемых текстовых файлов, которые можно просмотреть (или даже изменить) в стандартном текстовом редакторе, например *vi*.

В большинстве современных NMS-продуктов поддерживается компактная форма MIB, определяющая набор управляемых объектов для всех типов устройств, за управление которыми они отвечают. Администраторы NMS обычно переводят MIB производителей в формат, который может использовать NMS. После загрузки или перевода MIB администраторы могут обращаться к управляемым объектам при помощи цифрового или удобочитаемого идентификатора объекта.

Важно знать, как читать и понимать файлы MIB. Следующий пример – разобранная версия MIB-II (символы -- предваряют комментарий):

```
RFC1213-MIB DEFINITIONS ::= BEGIN

     IMPORTS
             mgmt, NetworkAddress, IpAddress, Counter, Gauge,
             TimeTicks
                     FROM RFC1155-SMI
             OBJECT-TYPE
                     FROM RFC-1212;
```

Таблица 2.7 (продолжение)

Сообщение об ошибке SNMPv2	Описание
NoCreation(11)	Вы попытались установить несуществующую переменную или создать переменную, которой нет в MIB
InconsistentValue(12)	Целостность переменной MIB нарушена, и она не принимает запросы set
ResourceUnavailable(13)	Для выполнения операции set нет системных ресурсов
CommitFailed(14)	Общее сообщение для всех остальных ошибок операции set
undoFailed(15)	Операция set не выполнена, и агент не смог отменить все предыдущие операции set до точки сбоя
authorizationError(16)	Команда SNMP не смогла пройти аутентификацию; другими словами, кто-то указал неправильную строку community
notWritable(17)	Переменная не принимает set, даже несмотря на то, что должна
inconsistentName(18)	Неудачная попытка установить значение переменной, потому что целостность переменной была каким-то образом нарушена

Ловушки SNMP

Ловушка – это способ, посредством которого агент сообщает NMS о сбоях[1]. В разделе «Менеджеры и агенты» главы 1 мы рассматривали понятие ловушки на общем уровне, теперь мы обсудим ее чуть подробнее. На рис. 2.9 изображена последовательность генерации ловушки.

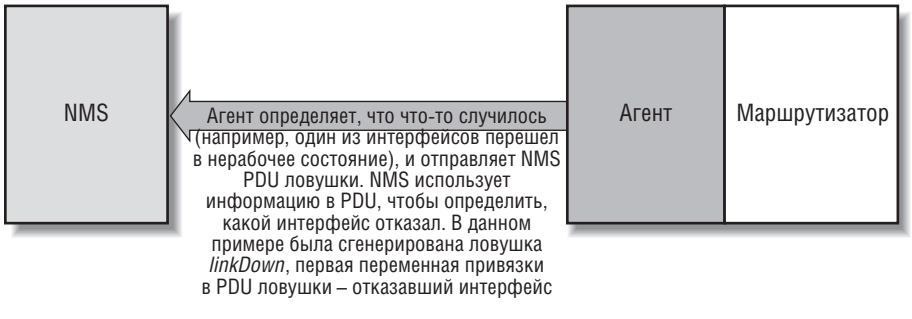

Рис. 2.9. *Последовательность генерации ловушки*

[1] На самом деле это не всегда так. Это может быть, например, сообщение о том, что маршрутизатор наконец заработал. – *Прим. науч. ред.*

Ловушка исходит от агента и отправляется получателю в соответствии с настройкой самого агента. Обычно получатель ловушки – IP-адрес NMS. NMS не отправляет агенту подтверждения, поэтому агент не может узнать, дошла ли ловушка до NMS. Так как SNMP использует UDP и ловушки предназначены для сообщения о проблемах в сети, ловушки особенно подвержены потерям и часто не доходят до получателей. Однако из-за того, что ловушки могут потеряться, они не становятся менее полезными; в хорошо спланированной среде они являются неотъемлемой частью управления сетью. Лучше, если ваше оборудование будет пытаться сообщить вам, что что-то не так, даже если сообщение никогда до вас не дойдет, чем просто оставит все как есть, чтобы вам пришлось догадываться, что случилось. Вот несколько ситуаций, о которых могут сообщить ловушки:

- Сетевой интерфейс устройства (где запущен агент) отключился.
- Сетевой интерфейс устройства (где запущен агент) снова заработал.
- Входящий вызов на модемный блок не смог установить соединение с модемом.
- Вентилятор на коммутаторе или маршрутизаторе отказал.

Когда NMS получает ловушку, ей нужно знать, как эту ловушку интерпретировать; то есть ей нужно знать, что означает ловушка и как интерпретировать переносимую ею информацию. В первую очередь ловушка определяется по групповому номеру ловушки. Групповых номеров ловушки семь (0–6), они приведены в табл. 2.8. Групповая ловушка 6 – это специальная обобщенная категория для «особых частных» ловушек, то есть определенных производителями или пользователями и не попадающих в шесть категорий групповых типов. Далее особые частные ловушки определяются по частному идентификатору (то есть идентификатору объекта *enterprise-id* где-то в ветви *enterprises* дерева MIB, *iso.org.dod.internet.private.enterprises*) и индивидуальному номеру ловушки (*specific-trap-number*), выбранному структурой, определившей ловушку. Таким образом, идентификатор объекта особой частной ловушки выглядит как *enterprise-id.specific-trap-number*. Например, когда Cisco определяет особые ловушки для своей частной MIB, она располагает их все в дереве частной MIB (*iso.org.dod.internet.private.enterprises.cisco*). Как мы увидим в главе 9, вы можете определять свои собственные частные ловушки; единственное требование – зарегистрировать свой частный номер в IANA.

Обычно ловушка содержит информацию. Как можно предположить, эта информация имеет вид объектов MIB и их значений; ранее было сказано, что такие пары объект–значение называются переменными привязки. Для групповых типов ловушек от 0 до 5 информация о том, что содержит ловушка, обычно встроена в программу NMS или получателя ловушки. Переменные привязки, которые входят в особые частные ловушки, установлены тем, кто определил ловушку. Например, при сбое модема в модемном блоке агент блока может отправить NMS

ловушку, информируя ее об этом инциденте. Скорее всего, ловушка будет частной, определенной производителем блока. Ее содержимое определяется производителем, но оно наверняка будет содержать достаточно информации для точного определения места сбоя (например, позицию и канал модемной карты в блоке).

Таблица 2.8. Групповые типы ловушек

Групповое имя и номер ловушки	Определение
coldStart (0)	Показывает, что агент перезагрузился. Все переменные управления будут сброшены; в частности, для переменных типа Counter и Gauge будут установлены значения 0. Польза ловушки coldStart заключается в том, что она может использоваться для определения момента, когда новое устройство подключается к сети. Когда устройство включается, оно отправляет эту ловушку своему получателю. Если получатель ловушки установлен правильно (то есть это IP-адрес вашей NMS), NMS может получить эту ловушку и определить, нужно ли ей управлять устройством
warmStart (1)	Показывает, что агент реинициализировался. Никакие переменные управления сброшены не будут
linkDown (2)	Отправляется при отключении одного из интерфейсов устройства. Первая переменная привязки определяет номер отключившегося интерфейса в таблице *interfaces*
linkUp (3)	Отправляется, когда один из интерфейсов устройства снова включился. Первая переменная привязки определяет, какой интерфейс включился
authentificationFailure(4)	Показывает, что кто-то пытался опросить агента с некорректной строкой community; полезно при определении попыток получения несанкционированного доступа к какому-либо устройству
egpNeighborLoss(5)	Показывает, что EGP-сосед отключился
enterpriseSpecific (6)	Показывает, что ловушка частная. Производители и пользователи определяют свои собственные ловушки в ветви *private-enterprise* дерева *object* SMI. Для правильной обработки этой ловушки NMS должна декодировать индивидуальный номер ловушки, который входит в SNMP-сообщение

В главе 1 мы сказали, что RFC 1697 определяет RDBMS MIB. Одна из ловушек, определяемых этой MIB, – *rdbmsOutOfSpace*.

11

Адаптация SNMP к вашей среде

SNMP может сильно упростить вашу жизнь как системного администратора, выполняя многие задачи, которые вам пришлось бы либо делать вручную, либо автоматизировать за счет написания какого-нибудь хитрого сценария. Позаботиться о большей части ежедневного системного мониторинга относительно легко: SNMP может запрашивать данные об использовании дискового пространства, уведомлять вас о синхронизации зеркальных томов или записывать, кто входит в систему и выходит из нее. В этой главе представлены некоторые интересные сценарии для автоматизации обычных задач системного администрирования. SNMP-сценарии в данной главе отражают лишь несколько задач, которые SNMP позволяет выполнить; используйте их в качестве отправного момента для своих идей.

Общая программа для генерации ловушек

В главе 9 были приведены ряд сценариев для сбора SNMP-информации при помощи Perl, программы snmptrap из OpenView и некоторые другие инструменты. Вот как мы использовали snmptrap для генерации ловушки, которая предоставляла нам информацию о некоторых проблемах с базой данных:

```
$ /opt/OV/bin/snmptrap -c public nms .1.3.6.1.4.1.2789.2500 "" 6 3003 "" \
.1.3.6.1.4.1.2500.3003.1 octetstringascii "Oracle" \
.1.3.6.1.4.1.2500.3003.2 octetstringascii "Backup Not Running" \
.1.3.6.1.4.1.2500.3003.3 octetstringascii "Call the DBA Now for Help"
```

Способ отправки ловушки при помощи Perl несколько сложнее, но не слишком:

```
#!/usr/local/bin/perl
# Имя файла: /opt/local/perl_scripts/snmptrap.pl
```

```
use SNMP_util "0.54";  # Это загрузит модули BER и SNMP_Session

snmptrap("public\@nms:162", ".1.3.6.1.4.1.2789", "sunserver1", 6, 1247,
         ".1.3.6.1.4.1.2789.1247.1", "int", "2448816");
```

В данной главе мы не будем подробно рассматривать, как писать такие команды, а уделим внимание их разумному использованию. Нам может потребоваться включить такие команды в сценарии загрузки или в другие программы через средства интеграции. Мы начнем с написания кода, записывающего успешные входы в систему.

Сценарии в данной главе и все примеры кода, приведенные в данной книге, можно загрузить с сайта *http://www.oreilly.com/catalog/esnmp2*.

Кто заходит на мою машину? (I-Am-In)

Когда пользователи UNIX входят в систему, система автоматически выполняет профиль; для пользователей оболочек Bourne, Korn или *bash* общий профиль системы называется */etc/profile*. Для пользователей *csh* и *tcsh* есть похожий файл (*/etc/login*). Мы можем воспользоваться SNMP для записи входов в систему, добавив к этим профилям ловушку. Само по себе это не так интересно, потому что UNIX уже ведет журнал входа пользователей в систему. Но, допустим, вы наблюдаете за несколькими десятками машин и не хотите проверять журнал событий каждой машины. Добавление ловушки к системному профилю позволяет отслеживать вход во все системы из одного места. Кроме того, это повышает безопасность журнала событий. Умному пользователю не так трудно удалить файл *wtmp*, в котором хранятся записи UNIX о входе в систему. При использовании SNMP для ведения журнала информация хранится на другом узле, контроль над которым должен быть лучше[1].

Для генерации ловушки вызовите в */etc/profile* внешнюю программу */opt/local/mib_programs/os/iamin* (эту же программу можно вызвать в */etc/login*). Вот код *iamin*:

```
#!/usr/local/bin/perl
#
# Имя файла: /opt/local/mib_programs/os/iamin

chomp ($WHO = `/bin/who am i \| awk \{\`print \$1\`\}`);

exit 123 unless ($WHO ne ``);

chomp ($WHOAMI = `/usr/ucb/whoami`);
chomp ($TTY = `/bin/tty`);
```

[1] Да, умный пользователь может перехватить и изменить SNMP-пакеты, переписать профиль оболочки и сделать много чего еще, чтобы не попасть в журнал. На самом деле нас не интересует обеспечение невозможности обхода журнальных записей; мы просто хотим его затруднить.

```
chomp ($FROM = `/bin/last \-1 $WHO \| /bin/awk \{\'print \$3\'\}`);

if ($FROM =~ /Sun|Mon|Tue|Wed|Thu|Fri|Sat/) { $FROM = "N/A"; }

# DEBUG BELOW
# print "WHO :$WHO:\n"; print "WHOAMI :$WHOAMI:\n"; print "FROM :$FROM:\n";

if ("$WHOAMI" ne "$WHO") { $WHO = "$WHO\-\>$WHOAMI"; }

# Отправка ловушки с помощью Net-SNMP
#
system "/usr/local/bin/snmptrap nms public .1.3.6.1.4.1.2789.2500 '' 6 1502 ''
.1.3.6.1.4.1.2789.2500.1502.1 s \"$WHO\"
.1.3.6.1.4.1.2789.2500.1502.2 s \"$FROM\"

# Отправка ловушки с помощью Perl
#
# использование SNMP_util "0.54";  # Это загрузит модули BER и SNMP Session
#snmptrap("public\@nms:162", ".1.3.6.1.4.1.2789.2500",
#          mylocalhostname, 6, 1502,
#".1.3.6.1.4.1.2789.2500.1502.1", "string", "$WHO",
#".1.3.6.1.4.1.2789.2500.1502.2", "string", "$FROM",
#".1.3.6.1.4.1.2789.2500.1502.3", "string", "$TTY");

# Отправка ловушки с помощью snmptrap OpenView
#
#system "/opt/OV/bin/snmptrap -c public nms .1.3.6.1.4.1.2789.2500
#        \"\" 6 1502 \"\"
#.1.3.6.1.4.1.2789.2500.1502.1 octetstringascii \"$WHO\"
#.1.3.6.1.4.1.2789.2500.1502.2 octetstringascii \"$FROM\"
#.1.3.6.1.4.1.2789.2500.1502.3 octetstringascii \"$TTY\"";
#

#
print "\n###############\n";
print "#   NOTICE   \# - You have been logged: :$WHO: :$FROM: :$TTY: \n"; #
print "###############\n\n";
```

Этот сценарий оказался несколько больше, чем предполагалось, потому что нужно будет убрать ряд записей под символами начала комментария. Например, многие программы запускаются в оболочке и потому вызывают одни и те же профили оболочки. Следовательно, нам нужно выяснить, был ли профиль вызван человеком; если нет, мы выходим[1]. Следующий шаг – больше узнать о личных данных пользователя, то есть откуда он входит в систему и кто он на самом деле, – нам не нужно,

[1] Это определение также не получится, если пользователь временно переключается на другого пользователя при помощи su. В правильно спроектированной среде пользователям на самом деле не нужно так часто пользоваться su – использование sudo или создание соответствующих групп должно существенно снизить потребность в использовании su.

Зная, как должен выглядеть результат vxprint, мы можем написать операторы Perl, которые определяют, когда генерировать ловушку. Эта задача занимает большую часть подпрограммы `get_vxprint`. Мы также знаем, какие типы сообщений об ошибках будут создаваться. Наш сценарий пытается игнорировать всю информацию с исправных дисков и сортировать сообщения об ошибках. Например, если поле `STATE` содержит `NEEDSYNC`, зеркальные диски, скорее всего, не синхронизировали и с томом нужно как-то разобраться. Сценарий не обрабатывает конкретно этот случай в явном виде, но он фиксируется в общей категории по умолчанию.

Сам механизм отправки ловушек связан с большим количеством переменных. В сущности, мы используем одну из рассмотренных нами утилит для отправки ловушек; идентификатор частной структуры – *.1.3.6.1.4.1.2789.2500*; частный номер ловушки – *1000*; и включены четыре переменных привязки, которые сообщают имя узла, имя тома, состояние тома и группу дисков.

Как и в случае с предыдущим сценарием, этот сценарий легко запускать периодически и просматривать результаты в любой используемой вами программе для управления сетью. Также нетрудно понять, как можно разработать похожие сценарии, которые формируют отчеты других программ контроля состояния.

Проверка дискового пространства

В агенте OpenView есть объект *fileSystemTable*, содержащий статистику об использовании диска и других параметрах файловой системы. На первый взгляд он кажется очень полезным: вы можете применять его для поиска имен файловых систем, количества свободных блоков и т. д. Но у него есть несколько нюансов, и для эффективного использования этой таблицы нам потребуются ловкие приемы. Проход по *fileSystemTable.fileSystemEntry.fileSystemDir* (*.1.3.6.1.4.1.11.2.3.1.2.2.1.10*) выводит список смонтированных файловых систем[1]:

```
[root][nms] /opt/OV/local/bin/disk_space> snmpwalk spruce \
.1.3.6.1.4.1.11.2.3.1.2.2.1.10
fileSystem.fileSystemTable.fileSystemEntry.fileSystemDir.14680064.1
: DISPLAY STRING- (ascii):  /
fileSystem.fileSystemTable.fileSystemEntry.fileSystemDir.14680067.1
: DISPLAY STRING- (ascii):  /var
fileSystem.fileSystemTable.fileSystemEntry.fileSystemDir.14680068.1
: DISPLAY STRING- (ascii):  /export
fileSystem.fileSystemTable.fileSystemEntry.fileSystemDir.14680069.1
: DISPLAY STRING- (ascii):  /opt
```

[1] Мы опустили начальную часть *.iso.org.dod.internet.private.enterprises.hp.nm.system.general* в результатах прохода в целях экономии места.

```
fileSystem.fileSystemTable.fileSystemEntry.fileSystemDir.14680070.1
: DISPLAY STRING- (ascii):  /usr
fileSystem.fileSystemTable.fileSystemEntry.fileSystemDir.41156608.1
: DISPLAY STRING- (ascii):  /proc
fileSystem.fileSystemTable.fileSystemEntry.fileSystemDir.41680896.1
: DISPLAY STRING- (ascii):  /dev/fd
fileSystem.fileSystemTable.fileSystemEntry.fileSystemDir.42991617.1
: DISPLAY STRING- (ascii):  /net
fileSystem.fileSystemTable.fileSystemEntry.fileSystemDir.42991618.1
: DISPLAY STRING- (ascii):  /home
fileSystem.fileSystemTable.fileSystemEntry.fileSystemDir.42991619.1
: DISPLAY STRING- (ascii):  /xfn
```

Давайте подумаем, как бы мы писали программу, проверяющую объем доступного дискового пространства. На первый взгляд кажется, что это просто. Но эта таблица содержит ряд объектов, которые не являются файловыми системами в общепринятом смысле; например, */proc* предоставляет доступ к запущенным в системе процессам и не отражает системы хранения. Из-за этого при опросе количества свободных блоков возникает проблема: в */proc* свободных блоков не будет, а в */dev/fd*, которая представляет дисковод для гибких дисков, свободные блоки будут, только если в дисководе находится диск. Вы можете предположить, что */home* будет вести себя, как нормальная файловая система, но на этом сервере она монтируется автоматически, поэтому ее поведение непредсказуемо; она может и не монтироваться, если не используется. Следовательно, если бы мы выполнили запрос количества свободных блоков, используя объект *fileSystem.fileSystemTable.fileSystemEntry.fileSystemBavail*, последние пять экземпляров могли бы в нормальных условиях возвратить 0. Поэтому результаты, которые мы получим от опроса всех объектов файловой системы, не имеют смысла без дальнейшей интерпретации. Как минимум, нам нужно выяснить, какие файловые системы нам важны, а какие нет. Скорее всего, это потребует разумного выбора номеров экземпляров.

Когда мы обнаружили эту проблему, то заметили, что у всех файловых систем, которые мы хотели проверить, начальные цифры в номерах экземпляров были одинаковыми, то есть *fileSystemDir.14680064.1*, *fileSystemDir.14680067.1*, *fileSystemDir.14680068.1* и т. д. Это наблюдение оказалось менее полезным, чем представлялось, – со временем мы узнали, что на других серверах могут быть разные начальные цифры в номерах экземпляров, более того, номера экземпляров на любом сервере могут меняться. Но даже если номер экземпляра изменяется, первые цифры этих номеров, по всей видимости, одинаковы для всех дисков или файловых систем одного типа. Например, у дисковых массивов могут быть номера экземпляров *fileSystemDir.12312310.1*, *fileSystemDir.12312311.1*, *fileSystemDir.12312312.1* и т. д., а у внутренних дисков – *fileSystemDir.12388817.1*, *fileSystemDir.12388818.1*, *fileSystemDir.12388819.1* и т. д.

Поэтому работать с номерами экземпляров можно, но трудно – нет ничего статического, что легко можно опросить. Невозможно просто ска-

- *Cammer* можно расширить для хранения его результатов в базе данных. Затем сценарий управления коммутатором для получения нужной информации об индексах можно настроить для считывания из базы данных.

Беспроводные сети

В данном разделе будет рассмотрено, как собирать управляющую статистику с беспроводных точек доступа при помощи MIB 802.11. MIB IEEE 802.11 доступна бесплатно на многих сайтах, в том числе *http://www.cs.ucla.edu/~hywong1/doc/IEEE802dot11-MIB.my*. Сама MIB достаточно насыщенная. Подробное рассмотрение этой MIB выходит за рамки данной главы. Вместо этого мы представим сценарий, который может собирать определенные данные с беспроводной точки доступа. Вот он:

```
#!/usr/bin/perl

use SNMP;
$SNMP::use_sprint_value = 1;
&SNMP::loadModules('IEEE802dot11-MIB');

my $host = "192.168.1.4";
my $sess = new SNMP::Session(DestHost => $host,
                             Version => 2,
                             Community => "public");

my %wapStats;
my $var = new SNMP::Varbind(['dot11CurrentChannel']);
do {
  $val = $sess->getnext($var);
  my $channel = $var->[$SNMP::Varbind::val_f];
  my $ifIndex = $var->[$SNMP::Varbind::iid_f];
  my($ssid, $mac, $manufacturer, $model, $rtsFailureCount,
  $ackFailureCount, $fcsErrorCount) = $sess->get([
        ['dot11DesiredSSID',$ifIndex],
        ['dot11MACAddress',$ifIndex],
        ['dot11ManufacturerID',$ifIndex],
        ['dot11ProductID',$ifIndex],
        ['dot11RTSFailureCount',$ifIndex],
        ['dot11ACKFailureCount',$ifIndex],
        ['dot11FCSErrorCount',$ifIndex]
        ]);
  $wapStats{$ifIndex} = "$channel,$ssid,$mac,$manufacturer,"
  $wapStats{$ifIndex} .= "$model,$rtsFailureCount,$ackFailureCount,
                          $fcsErrorCount";
}unless($sess->{ErrorNum});

foreach my $key (sort keys %wapStats){
        my($channel, $ssid, $mac, $manufacturer, $model,
```

```
        $rtsFailureCount, $ackFailureCount, $fcsErrorCount) =
                split(/,/,$wapStats{$key});

        print "WAP $ssid with MAC Address $mac (Manufacturer: $manufacturer,
Model: $model, Channel: $channel, ifIndex: $key)\n";
        print "\tdot11RTSFailureCount: $rtsFailureCount\n";
        print "\tdot11ACKFailureCount: $ackFailureCount\n";
        print "\tdot11FCSErrorCount: $fcsErrorCount\n";
}
```

Во-первых, заметьте, что мы используем Perl-модуль Net-SNMP. Во-вторых, обратите внимание на загруженный нами модуль MIB:

```
&SNMP::loadModules('IEEE802dot11-MIB');
```

Это очень важно. Это позволяет нам пользоваться в сценарии текстовыми именами объектов, а не цифровыми идентификаторами. Первое, что мы делаем, – устанавливаем переменную привязки для *dot11CurrentChannel*. Она определяется в MIB 802.11 следующим образом:

```
dot11CurrentChannel OBJECT-TYPE
        SYNTAX INTEGER (1..14)
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
            "The current operating frequency channel of the DSSS PHY.
             Valid channel numbers are as defined in 15.4.6.2"
            (Текущий рабочий частотный канал DSSS PHY.
             Допустимые номера каналов определены в 15.4.6.2)
    ::= {   dot11PhyDSSSEntry 1 }
```

Это используемый в данный момент идентификатор канала. Эта запись находится в таблице, содержащей и другие объекты. Нас интересует только канал, используемый в данный момент. Кроме того, обратите внимание, что эта таблица проиндексирована по *ifIndex*, объекту, который вы можете помнить по субдереву *interfaces*. Мы выполняем над этим объектом операцию `getnext`, которая возвращает идентификатор канала и *ifIndex*. Затем мы выполняем операцию `get` для семи объектов. Первые четыре – это SSID, MAC-адрес, идентификатор производителя и идентификатор продукта соответственно. Три остальных могут показаться непонятными. Скоро станет очевидно, почему мы их выбрали. Во-первых, вот три последних объекта из MIB:

```
dot11RTSFailureCount OBJECT-TYPE
        SYNTAX Counter32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
        "This counter shall increment when a CTS is not received
         in response to an RTS."
        (Этот счетчик увеличивается на 1, если в ответ на RTS
         не получено CTS.)
```

```
    ::= { dot11CountersEntry 8 }

dot11ACKFailureCount OBJECT-TYPE
        SYNTAX Counter32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "This counter shall increment when an ACK is not received
             when expected."
            (Этот счетчик увеличивается на 1, если не получено
             ожидаемое ACK.)
    ::= {  dot11CountersEntry 9 }

dot11FCSErrorCount OBJECT-TYPE
        SYNTAX Counter32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
            "This counter shall increment when an FCS error
             is detected in a received MPDU."
            (Этот счетчик увеличивается на 1, если
             в полученном MPDU обнаружена ошибка FCS.)
    ::= {  dot11CountersEntry 12 }
```

Прежде чем более подробно рассматривать эти объекты, давайте взглянем на выходные данные сценария:

```
$ ./wapstats.pl
WAP "thehouse2" with MAC Address 0:f:b5:3:fd:6f (Manufacturer: NETGEAR,
Model: WG302, Channel: 10, ifIndex: 1)
        dot11RTSFailureCount: 0
        dot11ACKFailureCount: 0
        dot11FCSErrorCount: 0
```

Как вы можете видеть, мы получили некоторую идентифицирующую информацию о WAP. Конечно, вы можете расширить этот сценарий и собирать другие элементы MIB 802.11 или иную статистику из MIB *interfaces*.

Теперь об этих трех объектах. Первый из них, *dot11RTSFailureCount*, в принципе, означает, что после отправки сообщения о готовности к отправке (RTS – Ready To Send) разрешение на отправку (CTS – Clear To Send) получено не было. *dot11ACKFailureCount*, как сказал Мэттью Гэст (Matthew Gast), автор книги издательства O'Reilly «Wireless Networks: The Definitive Guide», «непосредственно отслеживает количество потерянных входящих подтверждений. Если передается кадр, который требует подтверждения, а подтверждение не приходит, этот счетчик увеличивается на 1». *dot11FCSErrorCount* – это счетчик ошибок последовательности проверки кадров (FCS – Frame Check Sequence). В принципе, FCS – это дополнение к кадру для контроля ошибок. Если данный счетчик растет, это может быть признаком проблем с подсистемой базовой

станции (BSS – Base Station Subsystem), то есть с функциями точки доступа, связанными с радиопередачей. Как следует из описаний, эти три объекта сообщают о каком-то сбое.

Макс Бейкер (Max Baker) из Калифорнийского университета в Санта-Крузе предложил способ автоматической установки канала точки доступа в соответствии с показателем. Этот показатель рассчитывается из трех вышеупомянутых объектов. Предлагается следующая формула:

```
Показатель = .2 * Ошибки FCS + .4 * Ошибки RTS + .4 * Ошибки ACK
```

Мы можем внести этот показатель в наш сценарий при помощи следующей подпрограммы:

```
sub scoreChannel {
  my($rtsFailureCount, $ackFailureCount, $fcsErrorCount) = @_;
  return (.2 * $fcsErrorCount + .4 * $rtsFailureCount +
    .4 * $ackFailureCount);
}
```

Давайте снова запустим сценарий *stats* с этим дополнением:

```
$ ./wapstats.pl
WAP "thehouse2" with MAC Address 0:f:b5:3:fd:6f (Manufacturer: NETGEAR,
Model: WG302, Channel: 10, ifIndex: 1)
        dot11RTSFailureCount: 0
        dot11ACKFailureCount: 0
        dot11FCSErrorCount: 0
        Channel score: 0
        (Показатель канала: 0)
```

Чем ниже показатель, тем лучше. Мы можем найти канал с наименьшим показателем одним из двух способов:

- Все время активно наблюдать за точкой доступа и фиксировать показатель для определенного канала. Проблема с этим подходом заключается в том, что вы должны периодически изменять номер канала, чтобы другие беспроводные устройства могли выбрать его и начать использовать. Затем вам нужно в течение некоторого времени отслеживать эти переменные-показатели ошибок, изменять настройку канала и, позволив сотрудникам подключиться к новому каналу, собирать переменные ошибок; эти действия нужно выполнить для всех каналов, которые вы можете установить на точке доступа. Изменить канал также можно посредством SNMP (чуть ниже мы рассмотрим это более подробно).
- Изменять каналы при помощи SNMP и примерно минуту использовать каждый из них (такое решение предлагает Макс). По истечении минуты собрать статистику, подсчитать показатель и перейти к следующему каналу. По завершении процесса выясните, показатель какого канала наименьший, и установите его для точки доступа. Проблема этого подхода в том, что вам придется координировать изменение каналов с кем-то или чем-то, кто находит новую настройку

канала и начинает ею пользоваться. Это похоже на первый подход, но, так как вы наблюдаете за каждым каналом в течение краткого периода, вам может потребоваться каким-то образом автоматизировать устройство, которое, собственно, осуществляет подключение. Имейте в виду, что беспроводные карты могут автоматически находить канал.

Обнаружив канал с наименьшим показателем, вы можете заставить точку доступа использовать этот канал, применив тот же самый объект, которым мы воспользовались для первоначального обнаружения канала, – *dot11CurrentChannel*. Для установки нового канала беспроводной точки доступа можно воспользоваться следующим кодом:

```
$sess->set([['dot11CurrentChannel',$ifIndex,$newChannel,"INTEGER"]]);
```

При помощи SNMP можно управлять даже беспроводными устройствами. С разумным и творческим подходом вы сможете ввести беспроводные точки доступа в архитектуру управления сетью и эффективно ими управлять.

SNMP: объектно-ориентированный способ

Пакет для Perl SNMP::Info был разработан в Калифорнийском университете в Санта-Крузе. Официальный веб-сайт этого пакета – *http://snmp-info.sourceforge.net*.

SNMP::Info основан на Perl-модуле Net-SNMP. Он позволяет получать с устройства различную информацию без необходимости знать какие-либо идентификаторы объектов, MIB и т. д., и это выполняется при помощи объектной ориентации. Как он это делает? Он поддерживает развитый список MIB и может определить тип устройства, которое вы пытаетесь опросить. Если он знает об устройстве, вы можете воспользоваться заранее определенными методами для получения информации об интерфейсах и других данных. Вот пример сценария, который собирает информацию об интерфейсах коммутатора:

```
#!/usr/bin/perl

use SNMP::Info;

my $info = new SNMP::Info(
                            # Автоматическое определение более
                            # конкретного класса устройства
                            AutoSpecify => 1,
                            Debug       => 0,
                            # Остальное передается SNMP::Session
                            DestHost    => '192.168.0.148',
                            Community   => 'public',
                            Version     => 2
                          ) or die "Can't connect to device.\n";
```

12

MRTG

MRTG (Multi Router Traffic Grapher – система построения графиков трафика нескольких маршрутизаторов) – бесплатный и полностью перестраиваемый инструмент анализа тенденций, простой в настройке и использовании. Это на удивление маленькая программа, потому что в ней не реализован тяжеловесный пользовательский интерфейс. MRTG просто генерирует графики в формате GIF или PNG, которые внедряются в стандартные HTML-страницы. Следовательно, вы можете просматривать выходные данные MRTG при помощи любого веб-браузера и даже обеспечить доступность этих отчетов по всей своей сети, используя веб-сервер.

Несмотря на то что лучше всего MRTG приспособлен для отображения графиков использования интерфейсов маршрутизаторов, его можно настроить для построения графиков таких параметров, как использование памяти, средняя нагрузка и использование диска на серверном оборудовании. MRTG особенно полезен при выявлении резких повышений какого-либо параметра на продолжительное время, что является признаком проблем с емкостью и необходимости расширения. Например, вы можете обнаружить, что в часы наиболее интенсивной работы интерфейс T1 перегружается и вам нужно перейти на более широкий канал или что требуется увеличить объем памяти сервера. Подобным образом MRTG может сообщить, что сетевые подключения используют лишь часть доступной пропускной способности, а следовательно, вы можете отказаться от нескольких каналов T1 и снизить свои телекоммуникационные расходы.

Во многих местах, где востребован MRTG, его базовые возможности по построению графиков используются для планирования и распределения емкости. MRTG не предоставляет тонких статистических инструментов, которые требуются для расчета базовых показателей или прогноза времени необходимой модернизации сети. Однако он может быть

очень полезным в организациях, не имеющих ресурсов для покупки полнофункционального пакета анализа тенденций. Базовые показатели и прогнозы бесценны, но графики MRTG могут предоставить аналогичную информацию, легко воспринимаемую визуально; ваши глаза очень хорошо выявляют типичное поведение и тенденции, даже если они и не способны предоставить вам статистический анализ, который может понравиться руководству.

У MRTG есть много возможностей, позволяющих настроить его работу. Обсуждение каждой возможности выходит за рамки данной главы; вместо этого мы рассмотрим, как установить MRTG и использовать его базовые возможности по построению графиков. Также мы покажем, как можно настроить MRTG для сбора системной информации с сервера.

Важно понимать, что MRTG – это не решение для NMS. Несмотря на то что из-за своих возможностей по построению графиков на первый взгляд он кажется подобным NMS, на самом деле это простая система опроса, которая очень разумно организует выдаваемые выходные данные. Он выполняет те же самые операции `get`, которые выполняла бы NMS, но обнаружение и решение проблем – не его задача. У него нет ни средств подачи сигналов или обработки ловушек, ни возможности устанавливать значения объектов. Он создан просто для того, чтобы в графическом виде представить, как работает ваша сеть. Если вас интересует пакет NMS с открытым исходным кодом, посмотрите OpenNMS (*http://www.opennms.org*). Этот и другие пакеты NMS с открытым кодом описаны в приложении G.

Использование MRTG

Прежде чем начать пользоваться MRTG, вам нужно загрузить и установить программу. Основной веб-сайт MRTG – *http://www.mrtg.org*. Ссылка на загрузку переводит вас в каталог, поддерживаемый изобретателем и главным разработчиком MRTG, Тобиасом Этикером (*http://people.ee.ethz.ch/~oetiker/webtools/mrtg/pub/*). В этом каталоге помимо последней есть ряд старых версий MRTG. Мы загрузили файл из списка *mrtg-2.10.15.tar.gz* (UNIX-версия). В данной главе мы сосредоточимся на этой версии.

Для запуска MRTG нужны программы третьих сторон: Perl версии 5.004_5 (или более новой) и библиотеки *gd*, *libpng* и *zlib*. В MRTG используется Perl-реализация SNMP, поэтому вам не нужно беспокоиться о поиске и установке каких-либо библиотек для SNMP. Определить свою версию Perl (и ее местоположение) можно при помощи команды `perl -v`. Она может выдать или не выдать много информации. Если информация имеется, в первой строке будет указана установленная у вас версия Perl. Если вы получите какую-либо ошибку вида «команда не найдена», возможно, Perl не установлен. В любом случае последняя версия Perl есть на сайте *http://www.perl.com*.

Библиотека *gd* используется для генерации GIF-изображений, отображаемых MRTG. Ее можно загрузить с сайта *http://www.boutell.com/gd/*. Два других пакета, *libpng* и *zlib*, также используются для различных аспектов создания графических изображений. Они доступны на сайте *http://sourceforge.net/projects/libpng/*.

Убедившись, что на вашей машине установлены Perl, *gd*, *libpng* и *zlib*, загрузите и распакуйте UNIX-версию MRTG с помощью следующих команд:

```
[root][linuxserver] > cd /usr/local
[root][linuxserver] > tar -zxvf mrtg-2.10.15.tar.gz
```

После распаковки перейдите в созданный каталог (который должен называться *mrtg-2.10.15*) и прочтите указания по установке в файле *README*. Для сборки MRTG выполните три команды:

```
[root][linuxserver] ~/mrtg-2.10.15> ./configure
[root][linuxserver] ~/mrtg-2.10.15> make
[root][linuxserver] ~/mrtg-2.10.15> make install
```

Все три команды выдают много выходных данных, которые мы опустили. Команда `configure` проверяет наличие в системе инструментов, необходимых для сборки MRTG. Она будет сообщать вам, чего не хватает и где это взять. Запуск `make` собирает MRTG, но не утруждайтесь ее запуском, если команда `configure` не была успешно выполнена; MRTG не будет собран, если все не было правильно установлено и настроено. Наконец, `make install` устанавливает MRTG и помещает связанные с ним файлы в соответствующие места. Опять же, не запускайте `make install`, если предыдущая команда `make` была прервана с ошибками. Местоположение исполняемых файлов MRTG по умолчанию – */usr/local/mrtg-2/bin*. Возможно, вам нужно будет добавить этот каталог в путь поиска.

После сборки MRTG вам потребуется решить, куда помещать построенные им графики. Так как графики MRTG созданы для просмотра в веб-браузере, они часто хранятся в каталоге, видимом для веб-сервера. Однако на самом деле неважно, где они будут размещаться. Важнее то, кому вы хотите разрешить их просмотр. Скорее всего, вам не нужно, чтобы статистику вашей сети видели все. В небольшой сети эти графики можно разместить в каталоге вне поля зрения веб-сервера и пользоваться веб-браузером для просмотра отчетов в формате HTML в локальной файловой системе. В более крупной сети доступ к отчетам может потребоваться другим людям (например, персоналу по обслуживанию сети или руководству); чтобы разрешить доступ без публикации статистики своей сети для всех остальных, вам может потребоваться установить безопасный веб-сервер. В любом случае следующие команды, которые вам потребуется выполнить, выглядят примерно так:

```
[root][linuxserver] ~/mrtg-2.10.15> mkdir -p /mrtg/images
[root][linuxserver] ~/mrtg-2.10.15> cp ./images/mrtg*.gif /mrtg/images/
```

Первая команда создает каталог для хранения графиков, создаваемых MRTG. Вторая команда копирует некоторые изображения MRTG в этот каталог для дальнейшего использования в HTML-файлах. Далее в данной главе мы будем считать, что графики хранятся в каталоге */mrtg/images*.

Теперь вы готовы к настройке первого опрашиваемого устройства, которое в MRTG называется целью. Для определения опрашиваемых устройств и опций, применяемых при создании графиков, в MRTG используется файл конфигурации. Синтаксис этого файла сложен, но для помощи в его создании в MRTG есть инструмент под названием *cfgmaker*. Скорее всего, вам потребуется отредактировать файл вручную, но гораздо проще начать с рабочего шаблона. Вот как нужно выполнить *cfgmaker*:

```
[root][linuxserver] ~/mrtg-2.10.15> setenv PATH /usr/local/mrtg-2/bin:$PATH
[root][linuxserver] ~/mrtg-2.10.15> mkdir /mrtg/run
[root][linuxserver] ~/mrtg-2.10.15> cfgmaker --global 'WorkDir: \
/mrtg/images' --output /mrtg/run/mrtg.cfg public@10.0.0.1
```

Первый аргумент *cfgmaker* устанавливает в файле конфигурации переменную `WorkDir`. Она указывает MRTG, где хранить все данные, собираемые с устройств, которые требуется опрашивать. Второй аргумент указывает, куда нужно отправить выходные данные *cfgmaker*; в данном случае это файл */mrtg/run/mrtg.cfg*. Последний аргумент указывает устройство, которое мы хотим опросить, и строку community, используемую при его опросе, в формате *строка_сообщества@устройство*.

Выходные данные *cfgmaker* – это смесь команд и HTML. Он выполняет команды `getnext` для устройства, которое вы указали в командной строке, чтобы получить представление о том, сколько интерфейсов у этого устройства, какие из них работают, какие нет и т. д. Он проходит по дереву *iso.org.dod.internet.mgmt.mib-2.interfaces* (*1.3.6.1.2.1.2*) для определения общего количества интерфейсов в этой таблице. Затем он создает логические записи, представляющие собой список опрашиваемых устройств, только на самом деле список – это одно устройство, каждый интерфейс которого указан как цель. Например, *TR00ATL* – это вторая строка таблицы *interfaces* на нашем маршрутизаторе Cisco, поэтому *cfgmaker* создал запись `Target` под названием *10.0.0.1_2*. Если бы этот интерфейс занимал в таблице *interfaces* третью строку, то запись `Target` называлась бы *10.0.0.1_3*.

Вот сокращенная версия нашего файла *mrtg.cfg*:

```
EnableIPv6: no
WorkDir: /mrtg/images

Target[10.0.0.1_2]: 2:public@10.0.0.1:
SetEnv[10.0.0.1_2]: MRTG_INT_IP="10.0.0.1" MRTG_INT_DESCR="Serial0/1"
MaxBytes[10.0.0.1_2]: 192000
```

```
Title[10.0.0.1_2]: Traffic Analysis for 2 -- TROOATL
PageTop[10.0.0.1_2]: <H1>Traffic Analysis for 2 -- TROOATL</H1>
 <TABLE>
   <TR><TD>System:</TD>     <TD>TROOATL in </TD></TR>
   <TR><TD>Maintainer:</TD> <TD></TD></TR>
   <TR><TD>Description:</TD><TD>Serial0/1 [TRANSIT] T1 to NewSouth -
       CID unknown </TD></TR>
   <TR><TD>ifType:</TD>     <TD>frame-relay (32)</TD></TR>
   <TR><TD>ifName:</TD>     <TD>Se0/1</TD></TR>
   <TR><TD>Max Speed:</TD>  <TD>192.0 kBytes/s</TD></TR>
   <TR><TD>Ip:</TD>         <TD>router1</TD></TR>
 </TABLE>
```

Стоит немного подробнее рассмотреть формат файла конфигурации. Строки комментариев начинаются с символа #, в реальном файле конфигурации вы увидите много таких строк. Большая часть строк файла – либо команды, либо фрагменты HTML, которые будут использоваться в выходных файлах MRTG. Команды MRTG имеют формат *команда[ключ]: параметры*. Например, команда в третьей строке – это `Target`, ключ – `10.0.0.1_2`, а параметры – `2:public@10.0.0.1`. Ключ – это идентифицирующая строка, которая группирует записи в файле конфигурации и обеспечивает MRTG базовое имя файла для использования при генерации графиков и файлов HTML. В сложной среде MRTG может использоваться для мониторинга десятков единиц оборудования с сотнями интерфейсов; благодаря ключу в файле конфигурации поддерживается некое подобие порядка. Параметры – это, собственно, параметры для команды.

Теперь вы должны лучше понимать, что такое файл конфигурации. В первой строке указывается рабочий каталог, в котором MRTG будет размещать свои графики и файлы HTML. Это глобальная команда, поэтому ключа не нужно. Обычно рабочий каталог находится где-то в дереве веб-сервера, чтобы отчеты можно было просмотреть в веб-браузере. У себя мы установили */mrtg/images/*. Третья строка (`Target`) указывает MRTG, какое устройство нужно опросить. Формат этого параметра *интерфейс:строка_community@устройство*, в нашем случае это `2:public@10.0.0.1`. Устройство определяется по имени или IP-адресу узла; о строках community мы уже знаем. Так как MRTG – это инструмент лишь для сбора данных, строки community только для чтения будет достаточно. *интерфейс* указывает, какой интерфейс устройства опрашивать в соответствии с таблицей *ifTable* этого устройства. В данном случае мы опрашиваем интерфейс 2 из *ifTable*.

Строка `MaxBytes` устанавливает максимальное значение для параметров, которые MRTG будет считывать с интерфейса. По умолчанию MRTG считывает *ifInOctets* и *ifOutOctets*. Он пытается выбрать разумное максимальное значение в зависимости от типа интерфейса, который он должен считать с самого устройства. Так как это интерфейс Ethernet, MRTG устанавливает для `MaxBytes` значение `192000`. `Title` указывает заголовок HTML-страницы, генерируемой для графика. Наконец, `PageTop`

и следующие строки указывают MRTG, какую информацию размещать в верхней части HTML-страницы, содержащей графики использования. Команда содержит сам код HTML, который был сгенерирован *cfgmaker*.

В целом эта запись указывает MRTG опрашивать объекты по умолчанию (*ifInOctets* и *ifOutOctets*) в записи 2 таблицы интерфейсов устройства *10.0.0.1*. Следовательно, MRTG запустит команды `get` для идентификаторов объектов *.1.3.6.1.2.1.2.2.1.10.2* (*iso.org.dod.internet.mgmt.mib-2.interfaces.ifTable.ifEntry.ifInOctets.2*) и *.1.3.6.1.2.1.2.2.1.16.2* (*iso.org.dod.internet.mgmt.mib-2.interfaces.ifTable.ifEntry.ifOutOctets.2*). По умолчанию MRTG будет генерировать следующие графики:

- Дневной график со средними значениями за каждые 5 мин
- Недельный график со средними значениями за каждые 30 мин
- Месячный график со средними значениями за каждые 2 ч
- Годовой график со средними значениями за каждый день

Когда вы закончите, попробуйте запустить MRTG вручную, чтобы посмотреть, нет ли каких-либо проблем со сценарием конфигурации:

```
[root][linuxserver] ~/mrtg-2.10.15> mrtg /mrtg/run/mrtg.cfg
```

Если у MRTG не будет проблем с файлом конфигурации, он запустится без соответствующих ошибок. При возникновении проблем он будет предоставлять вам достаточно подробное их описание. Когда вы впервые запустите MRTG, он будет «жаловаться» на невозможность найти какие-либо журнальные файлы. Если вы запустите MRTG три раза, то увидите сообщения, похожие на следующие:

```
[root][linuxserver] ~/mrtg-2.10.15> mrtg /mrtg/run/mrtg.cfg
Rateup WARNING: /home/kschmidt/mrtg-2.10.15/bin/rateup could not read the
primary log file for 10.0.0.1_2
Rateup WARNING: /home/kschmidt/mrtg-2.10.15/bin/rateup The backup log file
for 10.0.0.1_2 was invalid as well
Rateup WARNING: /home/kschmidt/mrtg-2.10.15/bin/rateup Can't remove
10.0.0.1_2.old updating log file
Rateup WARNING: /home/kschmidt/mrtg-2.10.15/bin/rateup Can't rename
10.0.0.1_2.log to 10.0.0.1_2.old updating log file
Rateup WARNING: /home/kschmidt/mrtg-2.10.15/bin/rateup could not read the
primary log file for 10.0.0.1_3
Rateup WARNING: /home/kschmidt/mrtg-2.10.15/bin/rateup The backup log file
for 10.0.0.1_3 was invalid as well
Rateup WARNING: /home/kschmidt/mrtg-2.10.15/bin/rateup Can't remove
10.0.0.1_3.old updating log file
Rateup WARNING: /home/kschmidt/mrtg-2.10.15/bin/rateup Can't rename
10.0.0.1_3.log to 10.0.0.1_3.old updating log file

[root][linuxserver] ~/mrtg-2.10.15> mrtg /mrtg/run/mrtg.cfg
Rateup WARNING: /home/kschmidt/mrtg-2.10.15/bin/rateup Can't remove
10.0.0.1_2.old updating log file
```

```
Rateup WARNING: /home/kschmidt/mrtg-2.10.15/bin/rateup Can't remove
10.0.0.1_3.old updating log file

[root][linuxserver] ~/mrtg-2.10.15> mrtg /mrtg/run/mrtg.cfg
[root][linuxserver] ~/mrtg-2.10.15>
```

Как вы можете видеть, первый запуск программы вызвал несколько ошибок. Второй запуск вызвал только две ошибки, а в последний раз ошибок не было. При первом запуске MRTG эти ошибки – норма; не беспокойтесь о них.

Обратите внимание, что при запуске MRTG из командной строки вы можете увидеть следующее:

```
--------------------------------------------------------------------------
ERROR: Mrtg will most likely not work properly when the environment
       variable LANG is set to UTF-8. Please run mrtg in an environment
       where this is not the case. Try the following command to start:
      (MRTG, скорее всего, не будет правильно работать,
       когда для переменной среды LANG установлено значение UTF-8.
       Пожалуйста, не запускайте MRTG в такой среде.
       Для начала попробуйте следующую команду:)

       env LANG=C ./mrtg /mrtg/run/mrtg.cfg
--------------------------------------------------------------------------
```

Просто следуйте получаемым указаниям – и MRTG запустится нормально.

Следующий шаг – обеспечить запуск MRTG каждые 5 мин. MRTG необязательно должен запускать `root`; подойдет любой пользователь. Добавьте в запись *crontab* для соответствующего пользователя следующую строку:

```
*/5 * * * * /usr/local/mrtg-2/bin/mrtg /mrtg/run/mrtg.cfg
```

Она запускает MRTG каждый день каждые пять минут. Обратите внимание, что обозначение `*/5` характерно только для Linux; в других UNIX-системах вам придется указывать время в явном виде (`0,5,10,15,20,25,30,35,40,45,50,55`). В достаточно большой сети могут возникнуть проблемы, если MRTG не завершит все свои задачи по опросу до того, как начнется следующий цикл опроса. В таком случае установка пятиминутного интервала опроса может быть не лучшей идеей. Для определения интервала, лучше всего подходящего для вашей среды, вам может потребоваться поэкспериментировать.

Просмотр графиков

Построив несколько графиков, вы захотите взглянуть на них, чтобы увидеть результат. Для упрощения просмотра графиков в состав MRTG

входит сценарий *indexmaker*, который создает HTML-страницы индекса. Вот как запустить *indexmaker* для типичного набора графиков:

```
[root][linuxserver] ~/mrtg-2.10.15> indexmaker --title "Cisco to Internet" \
--filter name=~'10.0.0.1' --output /mrtg/images/cisco.html /mrtg/run/mrtg.cfg
```

Эта команда создает одну страницу индекса с графиком средних значений за 5 мин для каждой цели, указанной в файле *mrtg.cfg*. Имейте в виду, что цель – это интерфейс, с которого вы собираете данные. Если в маршрутизаторе четыре цели, в файле индекса появится четыре графика, которые будут указывать на дневной, недельный, месячный и годовой обобщенные графики для каждой цели. Параметр `--title` указывает *indexmaker*, какой заголовок использовать для файла индекса. Параметр `--filter name=~10.0.0.1` позволяет выбрать некоторые цели в файле *mrtg.cfg* при помощи регулярного выражения: мы указали *indexmaker* найти все цели, содержащие строку `10.0.0.1`. Параметр `--output` – это имя файла индекса. Последний аргумент командной строки – это полный путь к файлу конфигурации. В табл. 12.1 представлена краткая информация об этих и некоторых других полезных параметрах *indexmaker*.

Таблица 12.1. Параметры командной строки indexmaker

Параметр	Описание
`--title`	Указывает имя HTML-страницы
`--filter`	Указывает регулярное выражение, которое будет использоваться для поиска конкретной цели из файла *mrtg.cfg*. Эти соответствующие выражению цели используются для создания HTML-файлов отчетов
`--output`	Показывает полный путь к HTML-файлу, который предполагается создать. Выходные данные по умолчанию – стандартные
`--sort`	Отсортировать размещение графиков на странице индекса
`--columns`	Отсортировать графики на странице индекса по *x* столбцам, по умолчанию по 2
`--width`	Установить ширину графиков. По умолчанию не устанавливается
`--height`	Установить высоту графиков. По умолчанию не устанавливается
`--show`	Выберите график, который нужно показать на странице индекса. Значение по умолчанию – `day`. Другие возможности включают `week`, `month`, `year` и `none`

Чтобы увидеть полный перечень параметров *indexmaker*, запустите команду без каких-либо параметров. На рис. 12.1 изображено, как может выглядеть файл *cisco.html*, созданный *indexmaker*, когда он загружается в веб-браузер.

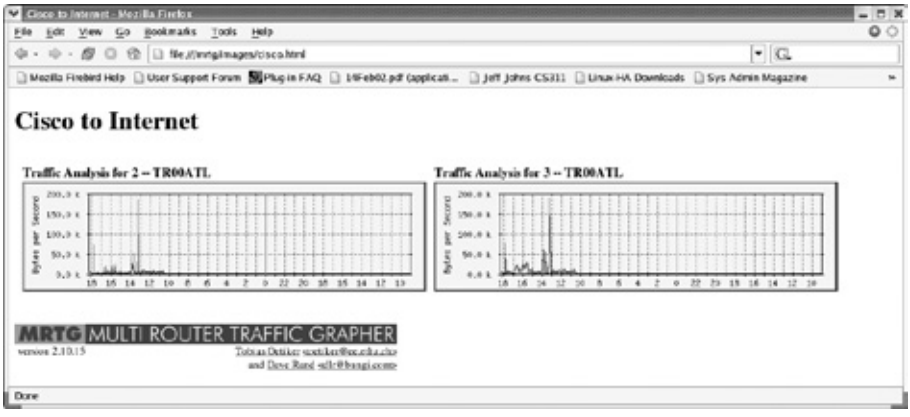

Рис. 12.1. *Обзор графиков Cisco*

На странице четыре графика, один для каждого работающего интерфейса (интерфейсы, которые функционировали на момент запуска `cfgmaker`) на маршрутизаторе. Эта строка содержит ссылки на другие страницы с более подробной информацией об отдельных интерфейсах. На рис. 12.2 изображены дневной, недельный, месячный и годовой графики нагрузки для интерфейса *TR00ATL*.

Дневной график (который на самом деле отражает 32-часовой период) обычно наиболее интересен. Он показывает средние значения трафика на этом интерфейсе за каждые пять минут. Входящий трафик (*ifInOctets*) представлен зеленым цветом, исходящий (*IfOutOctets*) – синим. Если бы мы щелкнули по одному из других интерфейсов на странице индекса Cisco (рис. 12.1), то увидели бы похожий график.

Вот и все, что касается просмотра графиков. MRTG хранит необработанные данные, которые он собирает, в неструктурированном текстовом файле, но благодаря разумным возможностям по обновлению журнальных записей размер файлов не выходит из-под контроля – он остается вполне приемлемым даже при активном применении MRTG.

Построение графиков других объектов

MRTG по умолчанию запрашивает переменные MIB *ifInOctets* и *ifOutOctets*, но помимо опроса различных типов устройств можно запрашивать значения других объектов и строить соответствующие графики. Сначала давайте организуем сбор MRTG количества входящих и исходящих октетов с сервера. Для этого запустите следующую команду:

```
[root][linuxserver] ~/mrtg-2.10.15> cfgmaker public@127.0.0.1 >> \
/mrtg2/run/mrtg.cfg
```

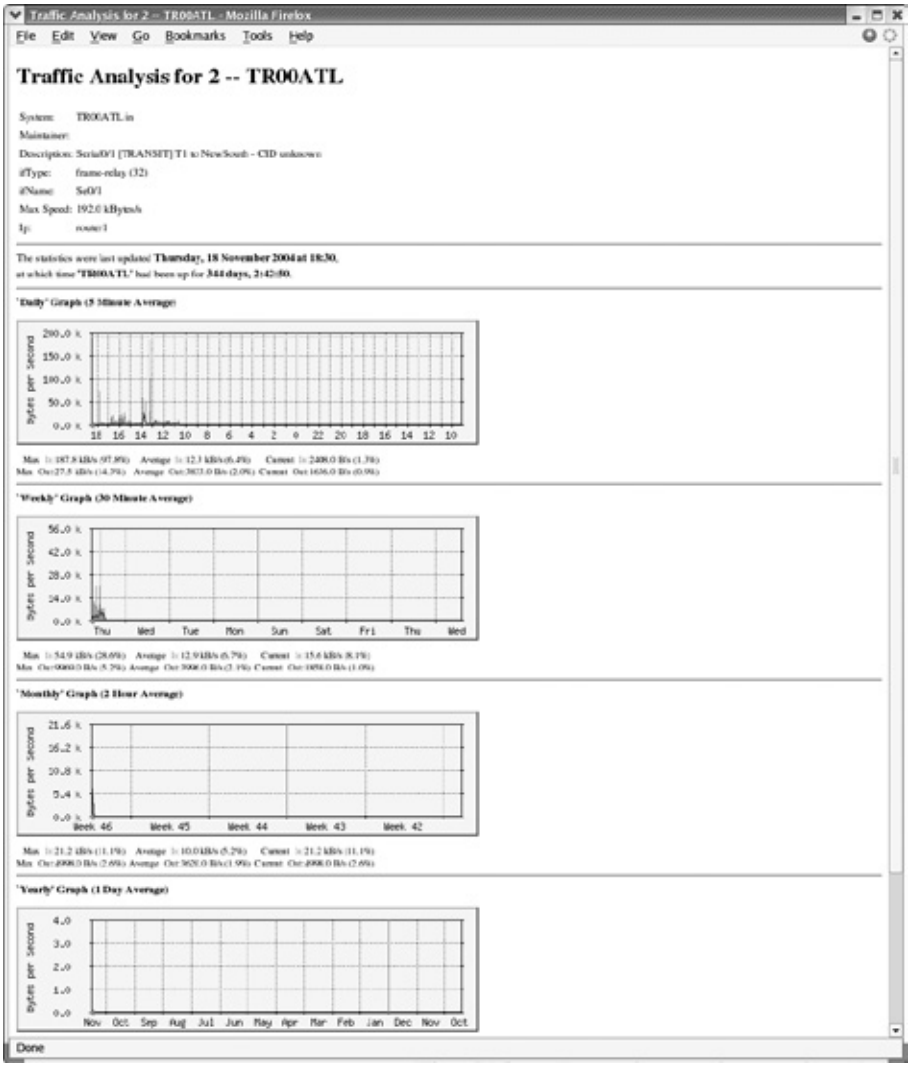

Рис. 12.2. *Дневной, недельный, месячный и годовой графики для Ethernet0*

Она почти идентична команде, которую мы запускали ранее в данной главе, кроме строки community и цели[1] (*public@127.0.0.1*). Мы добавили выходные данные в файл *mrtg.cfg*, а не указали файл для их вывода при помощи параметра `--output`; это позволяет нам добавить новый узел в существующий файл конфигурации, а не заводить новый файл. Так как в существующем файле уже указан рабочий каталог, мы также ис-

[1] Убедитесь, что на цели запущен SNMP-агент. Для обсуждения настройки различных SNMP-агентов для UNIX и Windows см. главу 6.

ключили параметр рабочего каталога (`--global 'WorkDir: ... '`). Команда `cfgmaker` добавляет в файл конфигурации ряд строк следующего вида:

```
Target[127.0.0.1_2]: 2:public@127.0.0.1:
SetEnv[127.0.0.1_2]: MRTG_INT_IP="" MRTG_INT_DESCR="eth0"
MaxBytes[127.0.0.1_2]: 12500000
Title[127.0.0.1_2]: Traffic Analysis for 2 -- box
PageTop[127.0.0.1_2]: <H1>Traffic Analysis for 2 -- box</H1>
 <TABLE>
   <TR><TD>System:</TD>     <TD>box in Atlanta,GA</TD></TR>
   <TR><TD>Maintainer:</TD> <TD>"kjs@guarded.net"</TD></TR>
   <TR><TD>Description:</TD><TD>eth0  </TD></TR>
   <TR><TD>ifType:</TD>     <TD>ethernetCsmacd (6)</TD></TR>
   <TR><TD>ifName:</TD>     <TD></TD></TR>
   <TR><TD>Max Speed:</TD>  <TD>12.5 MBytes/s</TD></TR>
 </TABLE>
```

Эти строки указывают MRTG, как опрашивать Ethernet-интерфейс сервера. Для интерфейса используется ключ `127.0.0.1`, а номер цели – 2. Почему 2? Помните, что `cfgmaker` проходит таблицу интерфейсов, чтобы определить, какие записи добавлять в файл конфигурации. Следовательно, вы увидите такой набор строк для каждого интерфейса устройства, в том числе интерфейса обратной связи. Номера целей фактически являются индексами в таблице интерфейсов; на этом сервере индекс интерфейса обратной связи – 1.

Теперь давайте создадим запись для построения графика числа пользователей, которые вошли в систему на сервере, и общего количества запущенных процессов. MRTG может построить графики этих параметров, но вам придется явно указать, графики каких переменных MIB нужно построить. Более того, вам потребуется указать две переменных – MRTG не будет показывать на графике только одну (это довольно странное ограничение, но оно, по крайней мере, логично: вспомните, что на графиках по умолчанию отображается количество и входящих, и исходящих октетов).

Сначала давайте рассмотрим переменные MIB, которые мы хотим отразить на графике. Две переменных, *hrSystemNumUsers* и *hrSystemProcesses*, определены как идентификаторы объектов *1.3.6.1.2.1.25.1.5.0* и *1.3.6.1.2.1.25.1.6.0* соответственно. *.0* на конце каждого идентификатора объекта показывает, что оба этих объекта являются скалярными переменными, а не частью таблицы. Оба объекта принадлежат MIB Host Resources (RFC 2790), которая определяет ряд управляемых объектов для системного администрирования. В некоторых агентах, работающих на серверных системах, эта MIB реализована, но, к сожалению, в агентах Microsoft и Solaris ее нет. Определения этих объектов следующие:

```
hrSystemNumUsers OBJECT-TYPE
    SYNTAX Gauge
    ACCESS read-only
    STATUS mandatory
```

```
        DESCRIPTION
            "The number of user sessions for which this host is storing state
             information. A session is a collection of processes requiring
             a single act of user authentication and possibly subject
             to collective job control."
             (Количество пользовательских сеансов, для которых
              этот узел хранит информацию о состоянии.
              Сеанс – это набор процессов, который требует одной
              аутентификации пользователя и, возможно, подчиняется
              коллективному контролю задач.)
        ::= { hrSystem 5 }

    hrSystemProcesses OBJECT-TYPE
        SYNTAX Gauge
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
            "The number of process contexts currently loaded
             or running on this system."
             (Количество контекстов процессов, загруженных
              или запущенных в данной системе в настоящий момент.)
        ::= { hrSystem 6 }
```

Запись, которую мы добавили в файл конфигурации, выглядит следующим образом:

```
Target[127.0.0.1_3]: 1.3.6.1.2.1.25.1.5.0&1.3.6.1.2.1.25.1.6.0:public@localhost
MaxBytes[127.0.0.1_3]: 512
Options[127.0.0.1_3]: gauge
Title[127.0.0.1_3]: Number of Users and Processes on localhost
YLegend[127.0.0.1_3]: Users/Processes
LegendI[127.0.0.1_3]: Users:
LegendO[127.0.0.1_3]: Processes:
PageTop[127.0.0.1_3]: <H1>Number of Users and Processes on localhost</H1>
 <TABLE>
   <TR><TD>System:</TD>     <TD>box in Atlanta,GA</TD></TR>
   <TR><TD>Maintainer:</TD> <TD>"kjs@guarded.net"</TD></TR>
 </TABLE>
```

В первой строке указано устройство, которое мы хотим опросить при помощи MRTG, а также два идентификатора объектов (*hrSystemNumUsers* и *hrSystemProcessess*), которые нужно отразить на графике. Этот оператор, очевидно, сложнее, чем оператор `Target`, рассмотренный нами ранее; его синтаксис *OID1&OID2:строка_community@устройство*. Идентификаторы объектов должны быть разделены символом амперсанда (`&`). Используя этот синтаксис, вы можете указать MRTG построить график двух любых скалярных переменных MIB.

В следующей строке мы устанавливаем `MaxBytes` равным `512`. Это максимальное значение для графика, значения больше 512 делаются равными 512 (забудьте о байтах; `MaxBytes` просто определяет максимальное значение). Для количества пользователей в системе это большое число,

вряд ли когда-нибудь в вашей системе одновременно будет так много пользователей. Это справедливо и для общего количества запущенных в системе процессов. Вы можете выбирать значения, разумные конкретно для вашей среды. Если вам нужны отдельные максимальные значения для каждого объекта, замените `MaxBytes` на две строки, устанавливающие `MaxBytes1` и `MaxBytes2`.

Команда `Options` новая; она позволяет изменять то, как MRTG воспринимает полученные данные. Единственный указанный нами параметр – `gauge`. Благодаря ему MRTG станет воспринимать собранную информацию как данные типа `Gauge`, а не типа `Counter`. Напоминаем, что данные типа `Counter` монотонно растут, а данные типа `Gauge` – нет. Поскольку в определениях MIB обоих объектов указан тип данных `Gauge`, этот параметр важен.

Параметры `YLegend`, `LegendI` и `LegendO` также являются новыми. `YLegend` просто меняет метку оси *Y* самого графика. Так как мы строим график количества пользователей и процессов, мы устанавливаем для `Legend` значение `Users/Processes`. Важно, чтобы легенда была короткой; если она слишком длинная, MRTG тихо ее игнорирует и не выводит в метке ничего. `LegendI` изменяет легенду под графиком для так называемой «входной переменной» (в данном случае это количество пользователей в системе – помните, что MRTG предполагает построение графика количества входящих и исходящих октетов). `LegendO` изменяет легенду для «выходной переменной» (общее количество запущенных в системе процессов). Терминология неудачная; просто помните, что MRTG всегда строит график для пары объектов и что легенда для входной переменной относится к первому объекту, а для выходной – ко второму.

После того как вы добавите эту запись в файл конфигурации и сохраните его, MRTG начнет собирать данные с устройства при каждом запуске. Если вы добавили соответствующую запись в файл *crontab*, то у вас все готово. Теперь мы воспользуемся *indexmaker*, чтобы создать интуитивно понятные файлы индекса для графиков сервера, как мы делали для графиков маршрутизатора. Команда для создания нового файла индекса аналогична той, которой мы пользовались, чтобы создать файл индекса Cisco:

```
[root][linuxserver] ~/mrtg-2.10.15> indexmaker --title "Linux Server" \
--filter name=~'127.0.0.1' --output /mrtg/images/linux.html /mrtg/run/mrtg.cfg
```

На рис. 12.3 изображена страница индекса для графиков сервера. На ней только два графика: один показывает трафик через интерфейс Ethernet, а второй – количество запущенных процессов и число пользователей в системе.

На рис. 12.4 изображены дневной, недельный, месячный и годовой графики для количества пользователей и процессов в системе.