3

Начинаем работу с PostgreSQL

В этой главе мы рассмотрим шаги, необходимые для установки PostgreSQL в различных операционных системах. Если вы работаете в системе Linux одной из последних версий, то, возможно, PostgreSQL уже установлен или имеется на установочных дисках. Для тех же, кто работает в UNIX, мы расс мотрим процесс компиляции исходного кода для платформы UNIX.

Мы также увидим, как установить PostgreSQL на платформе Windows, где перед инсталляцией PostgreSQL придется установить дополнительное программное обеспечение, моделирующее окружение UNIX.

По ходу дела коснемся следующих тем:

- Устанавливать или обновлять?
- Установка PostgreSQL из дистрибутива Linux
- Состав дистрибутива PostgreSQL
- Установка PostgreSQL из исходного кода и запуск PostgreSQL
- Создание баз данных
- Создание и заполнение таблиц
- Остановка PostgreSQL
- Установка PostgreSQL в Windows
- Cygwin – UNIX-среда в Windows
- Службы IPC в Windows
- Компиляция и конфигурирование PostgreSQL в Windows
- Автоматический запуск PostgreSQL в Windows

В настоящее время продолжается работа над версией PostgreSQL для Solaris. Пройдет еще некоторое время, прежде чем завершится ее окончательное тестирование. Поэтому установка PostgreSQL на Solaris не рассмотрена в этой книге. Энтузиасты PostgreSQL, желающие работать в Solaris, могут узнать последние новости по адресу *http://www.greatbridge.org* и даже принять участие в работе.

Устанавливать или обновлять?

Этот вопрос определенно возникнет у любителей Linux, решивших использовать PostgreSQL. В действительности на него нет однозначного ответа. Последняя версия на момент написания книги – PostgreSQL-7.1.2. Текущую версию можно получить с сайта *http://www.postgresql.org* и с перечисленных на нем ftp-серверов.

Если в вашем распоряжении имеется свежий дистрибутив Linux, то весьма вероятно, что в него входит и последняя версия PostgreSQL. Если же Linux установлен из более раннего дистрибутива, включающего версию 7.1.x, то можно обновить PostgreSQL до последней версии, загрузив соответствующий пакет с любого из сайтов, посвященных PostgreSQL. Однако в этой книге мы предполагаем, что читатель является новичком в PostgreSQL, и поэтому рассматриваем новую установку, а не трудоемкую и утомительную процедуру обновления.

Горячие поклонники Linux, предпочитающие обновления и имеющие в этом опыт, могут получить информацию по обновлению на сайте PostgreSQL.

В следующих разделах мы опишем установку PostgreSQL на UNIX-подобных платформах, на Linux и на Microsoft Windows. Процедура установки практически одинакова во всех разновидностях Linux.

Есть два основных способа установки PostgreSQL:

- установка PostgreSQL из дистрибутива Linux;
- установка PostgreSQL из исходного кода.

Мы начнем с установки PostgreSQL из дистрибутива Linux.

Установка PostgreSQL из дистрибутива Linux

Возможно, наиболее простой способ установки PostgreSQL на Linux состоит в использовании предварительно откомпилированных двоичных пакетов, доступных в формате менеджера пакетов RedHat (RedHat Package Manager, RPM) во многих дистрибутивах. Ко времени написания этой книги имелись пакеты RPM для следующих версий:

- RedHat 6.x и 7.x
- SuSe 6.4 и 7.x

- TurboLinux 6.x
- Caldera OpenLinux eServer 2.3
- Mandrake 7.x
- LinuxPPC 2000

Для установки полнофункциональной СУБД нам понадобятся как минимум базовый и серверный пакеты. Полный список пакетов приведен в табл. 3.1:

Таблица 3.1. Пакеты, необходимые для установки полнофункциональной СУБД

Пакет	Описание
postgresql	Базовый пакет
postgresql-server	Программы создания и запуска сервера
postgresql-devel	Заголовочные файлы и библиотеки для разработки
postgresql-jdbc	Драйверы JDBC для доступа к PostgreSQL из Java
postgresql-odbc	Драйверы ODBC для доступа к PostgreSQL
postgresql-perl	Интерфейс Perl для PostgreSQL
postgresql-python	Интерфейс Python для PostgreSQL
postgresql-tcl	Интерфейс Tcl для PostgreSQL
postgresql-test	Тестовые программы PostgreSQL
postgresql-tk	Оболочка и графический интерфейс Tk

К имени файла добавляется номер версии соответствующего пакета. Рекомендуется устанавливать пакеты с одинаковыми номерами версий и ревизий. Номер ревизии – это последняя цифра в номере версии, то есть x в 7.1.x.

Для установки пакетов используется программа RPM Package Manager. Перед тем как начинать установку, убедитесь, что вы зарегистрированы как суперпользователь (`superuser/root`). Для установки RPM-пакетов подойдет любой графический менеджер пакетов, например GnoRPM или Kpackage. Можно также скопировать все файлы RPM в один каталог и как суперпользователь (`superuser/root`) выполнить команду:

```
$ rpm -i *.rpm
```

Эта команда распаковывает пакет и копирует находящиеся в нем файлы в предназначенные для них каталоги.

Можно выполнить установку PostgreSQL и из пакета, входящего в состав дистрибутива Linux, такого как RedHat или SuSe. Например, в SuSe 7.x для установки PostgreSQL можно запустить утилиту YaST (рис. 3.1) и выбрать пакеты из списка, приведенного в табл. 3.1.

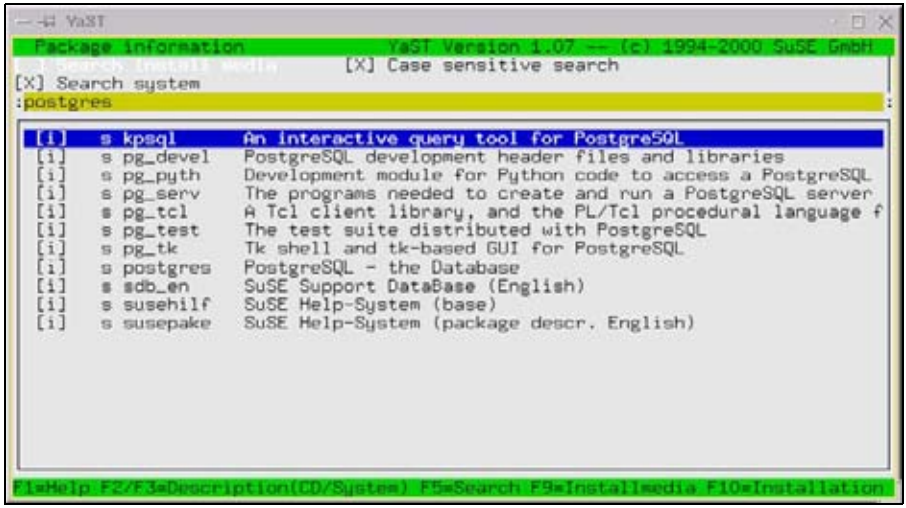

Рис. 3.1. Утилита установки YaST

В качестве альтернативы можно загрузить или заказать на CD коммерческую версию (само программное обеспечение остается бесплатным) PostgreSQL, включающую инсталлятор и дополнительную документацию. Одна из таких коммерческих версий доступна на сайте *http://www.greatbridge.com* компании Great Bridge.

Разработка PostgreSQL не прекращается, и, следовательно, время от времени будут выходить новые версии. Установка «вручную» из пакетов RPM имеет то преимущество, что позволяет обновлять установленную версию до текущей, просто повторяя процедуру. Для этого достаточно сообщить программе установки, что она должна выполнить обновление, а не первоначальную установку, заменив параметр `-i` на `-u`:

```
$ rpm -u *.rpm
```

Настоятельно рекомендуется сохранить резервную копию базы данных перед началом обновления. Подробные инструкции по выполнению обновления доступны на сайте PostgreSQL.

Резервное копирование и восстановление базы данных подробно рассматриваются в главе 11.

Состав дистрибутива PostgreSQL

К сожалению, во время написания этой книги стандарт на установку PostgreSQL еще не был выработан. Такой стандарт в большинстве случаев стал бы благом, хотя иногда и мог бы создавать некоторые трудности.

В состав PostgreSQL входят ряд приложений, утилит и каталогов с данными. Его главное приложение (`postmaster` или `postgres`) содержит

серверный код, отвечающий за обслуживание запросов на доступ к данным от клиентов. Утилиты, такие как `pg_ctl`, применяются для управления главным серверным процессом, который должен выполняться постоянно, обеспечивая работу сервера. В каталоге `data` находятся все файлы, необходимые базе данных. В них хранятся не только таблицы и записи, но и системные параметры.

По умолчанию устанавливаются все компоненты PostgreSQL, перечисленные в табл. 3.2, каждый в соответствующем подкаталоге. В качестве общего каталога (и каталога по умолчанию при установке из исходного кода) обычно используется `/usr/local/pgsql`.

Главный каталог PostgreSQL включает в себя следующие подкаталоги:

Таблица 3.2. Компоненты PostgreSQL

Каталог	Описание
`bin`	Приложения и утилиты, например `pg_ctl` и `postmaster`
`data`	Файлы базы данных
`doc`	Документация в формате HTML
`include`	Заголовочные файлы для разработки приложений PostgreSQL
`lib`	Библиотеки для разработки приложений PostgreSQL
`man`	Руководство по PostgreSQL
`share`	Примеры файлов конфигурации

Недостаток работы с единственным каталогом в том, что неизменные файлы программ и изменяющиеся данные хранятся совместно, а это не всегда удобно.

Файлы, используемые PostgreSQL, можно разделить на несколько категорий. Файлы программ не могут и не должны изменяться во время работы. Файлы журналов, создаваемые сервером, содержат информацию о доступе к базе данных и могут быть очень полезны в решении проблем. Они растут по мере того, как в них добавляются новые записи. Файлы данных – это сердце системы, в них хранится вся информация всех баз данных.

Из соображений эффективности и удобства администрирования файлы разных типов можно поместить в разные каталоги. Гибкость PostgreSQL позволяет хранить файлы программ, системных журналов и данных в различных местах, а дистрибутивы Linux позволяют извлечь из этой гибкости максимум выгоды.

Например, в SuSE 7.0 программы PostgreSQL размещаются вместе с остальными приложениями в `/usr/bin`, для журнальных файлов отводится `/var/log/postgresql`, а данные хранятся в `/var/lib/pgsql/data`. Благодаря этому можно легко разделить процедуры резервного копирования наиболее важных файлов данных и менее критичных журнальных файлов и т. п.

В других дистрибутивах могут применяться собственные схемы размещения файлов. Для просмотра списка файлов, установленных определенным пакетом, можно использовать утилиту RPM, запустив ее с такими параметрами:

```
$ rpm -q -l postgres
/usr/bin/createdb
...
/usr/share/man/manl/vacuum.1.gz
$
```

Чтобы просмотреть все установленные файлы, придется запустить `rpm` для каждого из пакетов, входящих в PostgreSQL:

```
$ rpm -q -l pg_serv
/sbin/conf.d/SuSEconfig.Postgres
...
/var/lib/pgsql/data/pg_options
$
```

В разных дистрибутивах названия пакетов могут слегка отличаться, в данном случае серверному пакету дано имя `pg_serv`. Вместо `rpm` можно использовать графический менеджер пакетов, например `kpackage` (рис. 3.2), поставляемый в комплекте с оболочкой KDE:

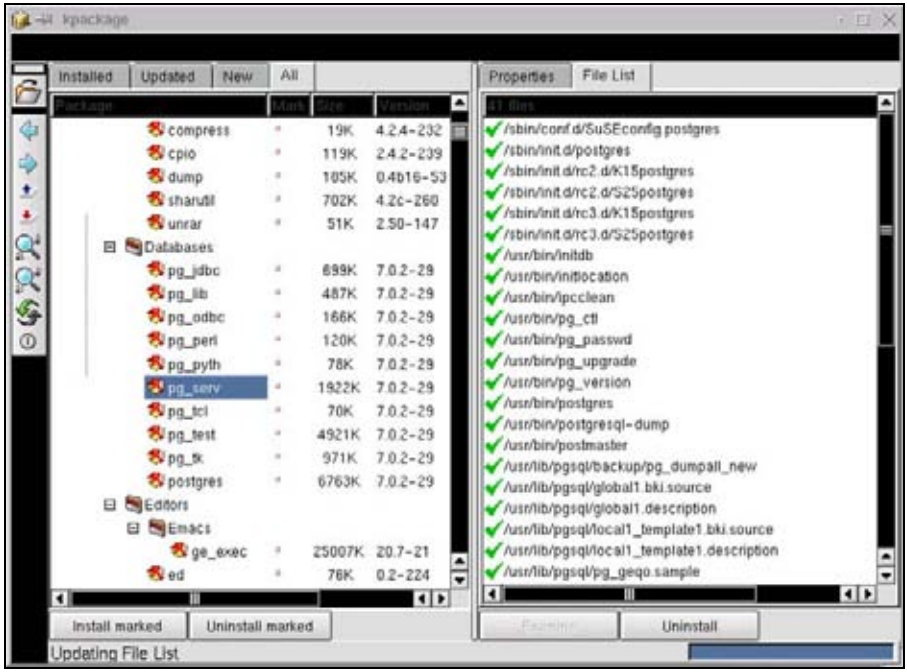

Рис. 3.2. Графический менеджер пакетов kpackage

Недостаток установки из дистрибутива Linux в том то, что не всегда очевидно, где какие файлы находятся. Так что тому, кто захочет обновить версию, может понадобиться настойчивость для того, чтобы разобраться в инсталляции.

Иногда бывает удобнее установить PostgreSQL из исходного кода. Тот, кто не планирует это делать, может пропустить следующий раздел и перейти к разделу «Запуск PostgreSQL».

Установка PostgreSQL из исходного кода

В то время как RPM пакеты предназначены для установки PostgreSQL в системах семейства Linux, исходный код позволяет собрать и установить PostgreSQL практически на любой UNIX-совместимой системе.

Исходный код последней версии, а часто и следующей бета-версии PostgreSQL можно получить на сайте *http://www.postgresql.org*. Если только вы не планируете находиться на переднем крае разработок, вполне можете ограничиться последней стабильной версией. На CD также записаны несколько предыдущих версий. Информацию о последних версиях и их доступности на CD можно получить на сайте PostgreSQL.

Исходный код распространяется в двух формах. Полный комплект исходного кода находится в файле с именем вроде

```
postgresql-7.1.2.tar.gz
```

Во время написания книги этот файл имел размер около 7,5 Мбайт. Кроме того, для облегчения загрузки исходный код упаковывается в несколько файлов меньшего размера:

```
postgresql-7.1.2.base.tar.gz
postgresql-7.1.2.docs.tar.gz
postgresql-7.1.2.opt.tar.gz
postgresql-7.1.2.test.tar.gz
```

Разумеется, эти имена тоже меняются в соответствии с номером текущей версии.

Компиляция PostgreSQL выполняется очень просто. Если вы уже компилировали продукты с открытым исходным кодом, то проблем не должно возникнуть. Впрочем, их не должно возникнуть, даже если это ваша первая попытка компиляции такого продукта. Для выполнения компиляции понадобится система Linux или UNIX с полностью установленной средой разработки, которая включает в себя компилятор C и GNU-версию утилиты `make`, необходимую для сборки системы.

Дистрибутивы Linux обычно поставляются со средой разработки, включающей GNU-инструментарий от фонда свободно распространяе-

мых программ FSF. Сюда входит превосходный компилятор GNU C (gcc), являющийся стандартным в Linux. Программы GNU доступны и для большинства других UNIX-платформ, поэтому мы рекомендуем использовать их для компиляции PostgreSQL. Последнюю версию инструментария можно получить по адресу *http://www.gnu.org*. Сразу после установки среды разработки можно переходить к компиляции PostgreSQL.

Скопируйте архивный файл с исходным кодом в какой-либо подходящий для компиляции каталог. Этот каталог – еще не то место, куда будет установлен PostgreSQL, он будет другим. Подходящим выбором будет подкаталог вашего домашнего подкаталога, поскольку для компиляции не требуются права суперпользователя, они понадобятся только при установке. Распакуйте архив:

```
$ tar zxvf postgres-7.1.2.tar.gz
```

Авторы предпочитают хранить исходные тексты в специальном каталоге для продуктов с открытым кодом /usr/src, но подойдет любой каталог, где достаточно свободного места для выполнения компиляции (понадобится около 50 Мбайт).

В процессе разархивирования будет создан новый каталог с именем, соответствующим версии PostgreSQL. Перейдите в этот каталог:

```
$ cd postgres-7.1.2
```

Там вы обнаружите файл INSTALL, содержащий подробные инструкции по сборке – на тот случай, если изложенный ниже метод автоматической сборки не даст результата.

В процессе сборки используется скрипт конфигурации configure, позволяющий настроить параметры сборки в соответствии с имеющимся окружением. При запуске без параметров configure руководствуется значениями по умолчанию:

```
$ ./configure
creating cache ./config.cache
checking host system type... i686-pc-linux-gnu
checking which template to use... linux
...
$
```

Скрипт configure устанавливает значения переменных, управляющих процессом сборки PostgreSQL, с учетом типа платформы, особенностей компилятора и т. д. Нюансы использования скрипта configure с различными значениями переменных выходят за рамки этой книги. Описание всех параметров можно найти в руководстве man.

Место установки выбирается скриптом configure автоматически. По умолчанию основным рабочим каталогом PostgreSQL назначается /usr/local/pgsql, а приложения и данные располагаются в его подкаталогах.

Запуск `configure` с параметрами позволяет изменить расположение каталогов. Чаще всего подставляются параметры, описанные в табл. 3.3:

Таблица 3.3. Параметры configure

Параметр	Описание
`--prefix=PREFIX`	Устанавливать в каталог `PREFIX`. По умолчанию – `/usr/local/psql`
`--bindir=DIR`	Устанавливать программы в `DIR`. По умолчанию – `PREFIX/bin`
`--with-tcl`	Компилировать с поддержкой приложений Tcl, таких как pgAccess
`--with-perl`	Компилировать с поддержкой приложений Perl
`--with-python`	Компилировать модуль поддержки Python

С помощью ключа `--help` можно получить полный список настроечных параметров:

```
$ ./configure --help

Usage: configure [options] [host]
Options: [defaults in brackets after descriptions]
Configuration:
  --cache-file=FILE       cache test results in FILE
...
$
```

Нет необходимости определять место расположения базы данных и журнальных файлов на данном этапе. Эти параметры могут быть заданы при запуске серверного процесса после того, как он уже установлен.

Закончив конфигурирование, можно приступать к сборке с помощью программы `make`.

> ***В процессе сборки PostgreSQL участвует ряд сложных make-файлов, управляющих компиляцией. Поэтому рекомендуется использовать GNU-make. В Linux эта версия установлена по умолчанию, но на других UNIX-платформах может потребоваться отдельная установка GNU-make. Часто ей дают имя*** `gmake`***, для того чтобы отличать от программы*** `gmake`***, поставляемой вместе с операционной системой. В дальнейшем обсуждении под make понимается GNU-make.***

Компилируем программы:

```
$ make
...
All of PostgreSQL successfully made. Ready to install.
```

Если все настроено правильно, будут выведены множество сообщений о компиляции и завершающее сообщение об успешном окончании сборки.

После завершения программы make необходимо скопировать программы в их рабочий каталог. Для этого мы также используем make, но прежде надо получить права суперпользователя:

```
$ su
# make install
...
Thank you for choosing PostgreSQL, the most advanced open source database
engine.
# exit
$
```

Вот, собственно, и все. Программы, составляющие сервер базы данных PostgreSQL, установлены.

Такой же результат мы получили бы при установке из пакетов RPM. Настало время выяснить, как же запускается PostgreSQL.

Запуск PostgreSQL

Основной процесс PostgreSQL – postmaster – это особого вида программа. Она полностью отвечает за доступ к данным – от всех пользователей и ко всем базам данных. Пользователю должны быть доступны его собственные данные и недоступны данные других пользователей, если он соответствующим образом не авторизован. Для этого серверный процесс должен владеть всеми файлами данных, запрещать прямой доступ к ним для обычных пользователей и выдавать разрешение на доступ, проверяя права пользователя, пославшего запрос.

Для управления доступом к данным в PostgreSQL реализована концепция псевдопользователей. Пользователь по имени postgres создается с единственной целью – владеть файлами данных. Регистрация в системе под этим именем с целью получения неавторизованного доступа невозможна. Программа postmaster по этому имени организует доступ к файлам данных от имени других пользователей.

Таким образом, первым шагом к созданию работающей базы данных PostgreSQL будет создание пользователя postgres.

В разных системах процедура создания пользователя может выглядеть по-разному. Пользователи Linux могут (зарегистрировавшись как root) выполнить команду useradd:

```
# useradd postgres
```

В других UNIX-системах может потребоваться создание домашнего каталога, редактирование файлов конфигурации или запуск административных программ. О программах администрирования можно узнать в документации по операционной системе.

Теперь надо создать и произвести начальную инициализацию базы данных и запустить процесс postmaster.

Используем утилиту initdb для инициализации, указав ей место в файловой системе, где должны быть расположены файлы данных. Предварительно надо создать (из-под пользователя root) каталог и назначить пользователя postgres его владельцем:

```
# mkdir /usr/local/pgsql/data
# chown postgres /usr/local/pgsql/data
```

Здесь выбрано расположение базы данных по умолчанию. Как говорилось ранее, каталог может быть и другим.

Для того чтобы инициализировать базу данных, необходимо запустить соответствующие программы PostgreSQL от имени пользователя postgres, как это показано ниже.

Для того чтобы зарегистрироваться как postgres, надо сначала войти в систему как суперпользователь (root), а затем выполнить команду su:

```
$ su
# su - postgres
pg$
```

Теперь все запускаемые программы получают права пользователя postgres и имеют доступ к файлам базы данных PostgreSQL. Для наглядности приглашение оболочки, запущенной под пользователем postgres, обозначено как pg$.

> ***Не следует поддаваться искушению упростить процесс, запуская программы с правами пользователя*** root. ***Серверные процессы, запущенные с правами суперпользователя, могут ослабить защищенность системы. В случае сбоев в серверном процессе пользователь может получить возможность проникновения в систему по сети. По этой причине программа*** postmaster ***отказывается запускаться под пользователем*** root.

Инициализируем базу данных программой initdb:

```
pg$ /usr/local/pgsql/bin/initdb -D /usr/local/pgsql/data
This database system will be initialized with the user name "Postgres".
This user will own all the data files and must also own the server process.
...
Success.
pg$
```

Если все пройдет хорошо, получим новую, совершенно пустую базу данных, расположенную в каталоге, который был указан параметром -D.

Теперь запустим собственно серверный процесс. Снова укажем параметр -D, чтобы сообщить программе postmaster, где находятся файлы базы данных. Для того чтобы пользователи могли получить доступ к данным по сети, следует указать параметр -i, разрешающий удаленный доступ:

```
pg$ /usr/local/pgsql/bin/postmaster -i -D /usr/local/pgsql/data >logfile 2>&1 &
```

По умолчанию PostgreSQL не разрешает удаленный доступ. Для того чтобы разрешить устанавливать соединения, нужно отредактировать файл конфигурации `pg_hba.conf`. Этот файл находится там же, где и файлы базы данных (в нашем примере – `/usr/local/pgsql/data`), и содержит записи, разрешающие или запрещающие удаленным пользователям соединяться с базой данных. Формат файла достаточно простой, а типовой файл, поставляемый с PostgreSQL, содержит множество комментариев, которые помогут добавить новые записи. Имеется возможность выдавать разрешения отдельным пользователям, компьютерам или группам компьютеров на доступ к отдельным базам данных.

Добавим запись, разрешающую любому компьютеру локальной сети устанавливать соединение с любой базой данных без аутентификации. Для того чтобы задать другую политику доступа, обратитесь к комментариям в файле конфигурации.

Добавим в конец файла `pg_hba.conf` строку такого вида:

```
host    all    192.168.0.0    255.255.0.0    trust
```

Это означает, что все компьютеры, IP-адрес которых начинается с 192.168, имеют доступ ко всем базам данных.

Мы также перенаправили вывод процесса в файл (`logfile` в домашнем каталоге пользователя `postgres`) и объединили стандартный вывод со стандартным выводом ошибок, с помощью конструкции `2>&1`. Разумеется, можно выбрать и другое расположение журнального файла, перенаправив в него вывод.

Теперь попробуем установить соединение с базой данных, чтобы убедиться, что она работает. Для выполнения простых задач администрирования, таких как создание пользователей, таблиц и баз данных, применяется утилита `psql`. Ниже в этой главе мы с ее помощью создадим и заполним пробную базу данных. А сейчас попробуем просто присоединиться к базе данных. Полученный ответ говорит о том, что процесс `postmaster` успешно запустился:

```
pg$ /usr/local/pgsql/bin/psql
psql: FATAL 1: Database "postgres" does not exist in the system catalog.
```

Пусть вас не смущает сообщение об ошибке: по умолчанию `psql` соединяется с базой данных на локальной машине и пытается открыть базу данных с тем же именем, что и у пользователя, запустившего программу. Попытка закончилась неудачно, т. к. мы не создавали базу данных с именем `postgres`. Однако такой ответ означает, что процесс `postmaster` выполняется и выдает диагностические сообщения.

Чтобы сообщить программе `psql`, с какой базой данных следует устанавливать соединение, надо указать параметр `-d`. Только что установленная система PostgreSQL содержит несколько баз данных, используемых в качестве основы для вновь создаваемых баз. Одна из них на-

зывается template1. При необходимости с этой базой данных можно установить соединение и выполнить какие-либо административные задачи.

Для того чтобы проверить работу в сети, установим соединение с другой машины, на которой также установлена PostgreSQL. Параметром -h определим компьютер (по имени или по IP-адресу) и соединимся с одной из системных баз данных (поскольку собственных баз мы еще не создали):

```
remote$ psql -h 192.168.0.66 -d template1
Welcome to psql, the PostgreSQL interactive terminal.

Type:    \copyright for distribution terms
       \h for help with SQL commands
       \? For help on internal slash commands
       \g or terminate with semicolon to execute query
       \q to quit
template1=# \q

remote$
```

Последнее, что следует сделать – настроить автоматический запуск серверного процесса postmaster при каждом рестарте машины.

Собственно, надо лишь обеспечить запуск программы postmaster во время загрузки. Здесь, опять-таки, нет единых стандартов для всех версий Linux и UNIX. Детальное описание можно найти в документации по системе.

Если PostgreSQL был установлен из дистрибутива Linux, то, скорее всего, процедура запуска уже настроена RPM-пакетом. В системе SuSE Linux при переходе в многопользовательский режим PostgreSQL запускается автоматически с помощью командного файла /etc/rc.d/init.d, вызывающего программу postgres.

Простейший способ создать программу запуска самостоятельно состоит в том, чтобы написать командный файл, запускающий postmaster с нужными параметрами, и вызывать его из файлов автозапуска, расположенных в /etc/rc.d. Не забывайте, что процесс postmaster должен запускаться пользователем postgres.

Здесь представлен сценарий, запускающий PostgreSQL, установленный из исходных текстов, с параметрами по умолчанию:

```
#!/bin/sh

# Script to start and stop PostgreSQL

SERVER=/usr/local/pgsql/bin/postmaster
PGCTL=/usr/local/pgsql/bin/pg_ctl
PGDATA=/usr/local/pgsql/data
```

```
OPTIONS=-i
LOGFILE=/usr/local/pgsql/data/postmaster.log

case "$1" in
    start)
        echo -n "Starting PostgreSQL..."
       su -l postgres -c "nohup $SERVER $OPTIONS -D $PGDATA >$LOGFILE 2>&1 &"
        ;;
    stop)
        echo -n "Stopping PostgreSQL..."
        su -l postgres -c "$PGCTL -D $PGDATA stop"
        ;;
    *)
        echo "Usage: $0 {start|stop}"
        exit 1
        ;;
esac
exit 0
```

В Debian Linux может потребоваться замена `su -l` ***на*** `su -`.

Поместите приведенную программу в исполняемый файл и назовите его, к примеру, MyPostgreSQL. Выполните команду `chmod`, чтобы сделать файл исполняемым:

```
# chmod a+rx MyPostgreSQL
```

Теперь следует сделать так, чтобы файл вызывался, когда PostgreSQL запускается и останавливается при запуске и остановке сервера:

```
MyPostgreSQL start
MyPostgreSQL stop
```

В системах, использующих команды инициализации в стиле System V (а это большинство дистрибутивов Linux), достаточно поместить команду в соответствующее место. Например, в SuSE Linux следует поместить файл в `/etc/rc.d/init.d/MyPostgreSQL` и для автоматического запуска и остановки PostgreSQL при переходе системы в многопользовательский режим и обратно создать следующие символические ссылки на него:

```
/etc/rc.d/rc2.d/S25MyPostgreSQL
/etc/rc.d/rc2.d/K25MyPostgreSQL
/etc/rc.d/rc3.d/S25MyPostgreSQL
/etc/rc.d/rc3.d/K25MyPostgreSQL
```

Обратитесь к документации по операционной системе за дополнительными сведениями о командах запуска.

Пришло время создать базу данных.

Создание базы данных

Позже мы вернемся к администрированию баз данных и рассмотрим его более подробно, а сейчас, когда у нас уже есть работающая база данных, рассмотрим эту тему вкратце. Создадим простую базу данных, которую назовем bpsimple, и будем использовать ее в учебных целях. В дальнейшем в примерах, демонстрирующих возможности PostgreSQL, будем иметь в виду именно ее.

Прежде чем начать, убедимся, что PostgreSQL запущен, для этого найдем postmaster в списке работающих процессов:

```
$ ps -el | grep post
```

Если процесс с именем postmaster запущен (имя на дисплее может быть сокращено), значит, сервер PostgreSQL работает.

Каждый пользователь PostgreSQL может создавать собственные базы данных и управлять доступом к хранимым в них данным. Но прежде чем создавать базы данных, необходимо создать в системе записи о пользователях PostgreSQL. Для этого запустим утилиту createuser.

С помощью команды su (из-под пользователя root) зарегистрируемся как пользователь postgres. Затем запустим утилиту createuser, регистрирующую пользователя. Пользователю с указанным именем будет разрешена работа с PostgreSQL. Зарегистрируем пользователя neil (уже созданного в UNIX/Linux):

```
$ su
# su - postgres
pg$ /usr/local/pgsql/bin/createuser neil
Shall the new user be able to create databases? (y/n) y
Shall the new user be able to create new users (y/n) n
CREATE USER
pg$
```

Здесь пользователю neil предоставлено право создавать новые базы данных, но не дано право создавать других пользователей.

Теперь, когда зарегистрирован пользователь PostgreSQL с такими правами, можно создать базу данных. Вернитесь в собственную сессию (не в сессию root) и введите:

```
$ /usr/local/pgsql/bin/createdb bpsimple
CREATE DATABASE
$
```

Теперь можно установить соединение (локальное) с сервером, используя интерактивный терминал psql:

```
$ /usr/local/pgsql/bin/psql -d bpsimple
Welcome to psql, the PostgreSQL interactive terminal.
...
bpsimple=#
```

Вы зарегистрировались в PostgreSQL и можете выполнять некоторые команды. Для возврата в оболочку выполните команду /q.

Создание таблиц

Таблицы в базе bpsimple можно создавать, вводя команды в ответ на приглашение psql, но гораздо проще будет загрузить файл с набором сценариев с сайта издательства Wrox, распаковать его и запустить командой \i <имяфайла>, которая выполняет команды, считывая их из файла. Команды находятся в обычном текстовом файле, так что при желании их можно редактировать в любом текстовом редакторе:

```
bpsimple=# \i create_tables.sql
CREATE TABLE
...
bpsimple=#
```

Можно рекомендовать написание командного файла для создания схемы (то есть таблиц, индексов, процедур). В этом случае при необходимости повторного создания базы данных можно будет запустить готовый сценарий. То же относится и к изменениям в схеме.

Файл create_tables.sql, входящий в набор, содержит SQL-операторы создания таблиц:

```
create table customer
(
    customer_id             serial                      ,
    title                   char(4)                     ,
    fname                   varchar(32)                 ,
    lname                   varchar(32)          not null,
    addressline             varchar(64)                 ,
    town                    varchar(32)                 ,
    zipcode                 char(10)             not null,
    phone                   varchar(16)                 ,
    CONSTRAINT              customer_pk PRIMARY KEY(customer_id)
);


create table item
(
    item_id                 serial                      ,
    description             varchar(64)          not null,
    cost_price              numeric(7,2)                ,
    sell_price              numeric(7,2)                ,
    CONSTRAINT              item_pk PRIMARY KEY(item_id)
);


create table orderinfo
(
```

```
    orderinfo_id            serial                      ,
    customer_id             integer             not null,
    date_placed             date                not null,
    date_shipped            date                        ,
    shipping                numeric(7,2)                ,
    CONSTRAINT              orderinfo_pk PRIMARY KEY(orderinfo_id)
);

create table stock
(
    item_id                 integer             not null,
    quantity                integer             not null,
    CONSTRAINT              stock_pk PRIMARY KEY(item_id)
);

create table orderline
(
    orderinfo_id            integer             not null,
    item_id                 integer             not null,
    quantity                integer             not null,
    CONSTRAINT              orderline_pk PRIMARY KEY(orderinfo_id, item_id)
);

create table barcode
(
    barcode_ean             char(13)            not null,
    item_id                 integer             not null,
    CONSTRAINT              barcode_pk PRIMARY KEY(barcode_ean)
);
```

Удаление таблиц

Если в какой-то момент мы решим удалить все таблицы и начать все сначала, это можно будет сделать. Необходимый для этого набор команд находится в файле `drop_tables.sql`:

```
drop table barcode;

drop table orderline;

drop table stock;

drop table orderinfo;

drop table item;

drop table customer;
```

```
drop sequence customer_customer_id_seq;

drop sequence item_item_id_seq;

drop sequence orderinfo_orderinfo_id_seq;
```

Будьте осторожны: удаляя таблицы, вы удаляете и все хранящиеся в них данные!

Если этот сценарий запускается после создания таблиц, то запустите повторно `create_tables.sql` перед тем, как заполнять таблицы данными.

Заполнение таблиц

Последний по порядку, но не по важности шаг – заполнение таблиц, то есть занесение в них данных.

Фигурирующие в примерах данные имеются в файле `pop_tablename.sql`. Вы, конечно, можете взять и собственные – с учетом того, что и результаты будут отличаться от приведенных в книге. Поэтому, если вы еще не стали специалистом, наверное, лучше использовать наши данные.

Наличие переносов строк обусловлено лишь форматом печатной страницы, команды можно вводить по одной на строке. Не забывайте о завершающей точке с запятой, она сообщает `psql`, где заканчивается SQL-команда.

Таблица customer

```
insert into customer(title, fname, lname, addressline, town, zipcode, phone)
values('Miss','Jenny','Stones','27 Rowan Avenue','Hightown','NT2 1AQ','023
9876');
insert into customer(title, fname, lname, addressline, town, zipcode, phone)
values('Mr','Andrew','Stones','52 The Willows','Lowtown','LT5 7RA','876
3527');
insert into customer(title, fname, lname, addressline, town, zipcode, phone)
values('Miss','Alex','Matthew','4 The Street','Nicetown','NT2 2TX','010
4567');
insert into customer(title, fname, lname, addressline, town, zipcode, phone)
values('Mr','Adrian','Matthew','The Barn','Yuleville','YV67 2WR','487
3871');
insert into customer(title, fname, lname, addressline, town, zipcode, phone)
values('Mr','Simon','Cozens','7 Shady Lane','Oahenham','OA3 6QW','514
5926');
insert into customer(title, fname, lname, addressline, town, zipcode, phone)
values('Mr','Neil','Matthew','5 Pasture Lane','Nicetown','NT3 7RT','267
1232');
```

```
insert into customer(title, fname, lname, addressline, town, zipcode, phone)
values('Mr','Richard','Stones','34 Holly Way','Bingham','BG4 2WE','342 5982');
insert into customer(title, fname, lname, addressline, town, zipcode, phone)
values('Mrs','Ann','Stones','34 Holly Way','Bingham','BG4 2WE','342 5982');
insert into customer(title, fname, lname, addressline, town, zipcode, phone)
values('Mrs','Christine','Hickman','36 Queen Street','Histon','HT3 5EM','342
5432');
insert into customer(title, fname, lname, addressline, town, zipcode, phone)
values('Mr','Mike','Howard','86 Dysart Street','Tibsville','TB3 7FG','505
5482');
insert into customer(title, fname, lname, addressline, town, zipcode, phone)
values('Mr','Dave','Jones','54 Vale Rise','Bingham','BG3 8GD','342 8264');
insert into customer(title, fname, lname, addressline, town, zipcode, phone)
values('Mr','Richard','Neill','42 Thached way','Winersby','WB3 6GQ','505
6482');
insert into customer(title, fname, lname, addressline, town, zipcode, phone)
values('Mrs','Laura','Hendy','73 Margeritta Way','Oxbridge','OX2 3HX','821
2335');
insert into customer(title, fname, lname, addressline, town, zipcode, phone)
values('Mr','Bill','O\'Neill','2 Beamer Street','Welltown','WT3 8GM','435
1234');
insert into customer(title, fname, lname, addressline, town, zipcode, phone)
values('Mr','David','Hudson','4 The Square','Milltown','MT2 6RT','961 4526');
```

Таблица item

```
insert into item, (description, cost_price, sell_price) values('Wood Puzzle',
15.23, 21.95);
insert into item(description, cost_price, sell_price) values('Rubic Cube',
7.45, 11.49);
insert into item(description, cost_price, sell_price) values('Linux CD',
1.99, 2.49);
insert into item(description, cost_price, sell_price) values('Tissues',
2.11, 3.99);
insert into item(description, cost_price, sell_price) values('Picture
Frame', 7.54, 9.95);
insert into item(description, cost_price, sell_price) values('Fan Small',
9.23, 15.75);
insert into item(description, cost_price, sell_price) values('Fan Large',
13.36, 19.95);
insert into item(description, cost_price, sell_price) values('Toothbrush',
0.75, 1.45);
insert into item(description, cost_price, sell_price) values('Roman Coin',
2.34, 2.45);
insert into item(description, cost_price, sell_price) values('Carrier Bag',
0.01, 0.0);
insert into item(description, cost_price, sell_price) values('Speakers',
19.73, 25.32);
```

Таблица barcode

```
insert into barcode(barcode_ean, item_id) values('6241527836173', 1);
insert into barcode(barcode_ean, item_id) values('6241574635234', 2);
insert into barcode(barcode_ean, item_id) values('6264537836173', 3);
insert into barcode(barcode_ean, item_id) values('6241527746363', 3);
insert into barcode(barcode_ean, item_id) values('7465743843764', 4);
insert into barcode(barcode_ean, item_id) values('3453458677628', 5);
insert into barcode(barcode_ean, item_id) values('6434564564544', 6);
insert into barcode(barcode_ean, item_id) values('8476736836876', 7);
insert into barcode(barcode_ean, item_id) values('6241234586487', 8);
insert into barcode(barcode_ean, item_id) values('9473625532534', 8);
insert into barcode(barcode_ean, item_id) values('9473627464543', 8);
insert into barcode(barcode_ean, item_id) values('4587263646878', 9);
insert into barcode(barcode_ean, item_id) values('9879879837489', 11);
insert into barcode(barcode_ean, item_id) values('2239872376872', 11);
```

Таблица orderinfo

```
insert into orderinfo(customer_id, date_placed, date_shipped, shipping)
values(3,'03-13-2000','03-17-2000', 2.99);
insert into orderinfo(customer_id, date_placed, date_shipped, shipping)
values(8,'06-23-2000','06-24-2000', 0.00);
insert into orderinfo(customer_id, date_placed, date_shipped, shipping)
values(15,'09-02-2000','09-12-2000', 3.99);
insert into orderinfo(customer_id, date_placed, date_shipped, shipping)
values(13,'09-03-2000','09-10-2000', 2.99);
insert into orderinfo(customer_id, date_placed, date_shipped, shipping)
values(8,'07-21-2000','07-24-2000', 0.00);
```

Таблица orderline

```
insert into orderline(orderinfo_id, item_id, quantity) values(1, 4, 1);
insert into orderline(orderinfo_id, item_id, quantity) values(1, 7, 1);
insert into orderline(orderinfo_id, item_id, quantity) values(1, 9, 1);
insert into orderline(orderinfo_id, item_id, quantity) values(2, 1, 1);
insert into orderline(orderinfo_id, item_id, quantity) values(2, 10, 1);
insert into orderline(orderinfo_id, item_id, quantity) values(2, 7, 2);
insert into orderline(orderinfo_id, item_id, quantity) values(2, 4, 2);
insert into orderline(orderinfo_id, item_id, quantity) values(3, 2, 1);
insert into orderline(orderinfo_id, item_id, quantity) values(3, 1, 1);
insert into orderline(orderinfo_id, item_id, quantity) values(4, 5, 2);
insert into orderline(orderinfo_id, item_id, quantity) values(5, 1, 1);
insert into orderline(orderinfo_id, item_id, quantity) values(5, 3, 1);
```

Таблица stock

```
insert into stock(item_id, quantity) values(1,12);
insert into stock(item_id, quantity) values(2,2);
insert into stock(item_id, quantity) values(4,8);
```

```
insert into stock(item_id, quantity) values(5,3);
insert into stock(item_id, quantity) values(7,8);
insert into stock(item_id, quantity) values(8,18);
insert into stock(item_id, quantity) values(10,1);
```

Теперь, запустив сервер и создав базу данных, создав и заполнив таблицы, можно продолжить изучение PostgreSQL.

Остановка PostgreSQL

Очень важно, чтобы серверный процесс PostgreSQL был правильно остановлен. Только в этом случае он сможет завершить запись в базу данных и освободить занятую им разделяемую память.

Для того чтобы корректно остановить базу данных, зарегистрируйтесь как `postgres` или `root` и запустите утилиту `pg_ctl`:

```
# /usr/local/pgsql/bin/pg_ctl -D /usr/local/pgsql/data stop
```

Конечно, если созданы сценарии для запуска, то можно использовать команду:

```
# /etc/rc.d/init.d/MyPostgreSQL stop
```

Эти сценарии выполняют также корректное закрытие базы данных при остановке и перезагрузке машины.

Ресурсы

Дополнительную информацию об утилитах `psql`, `createuser` и `createdb` можно получить из справочных страниц, воспользовавшись командой `man`, и из другой документации, поставляемой с PostgreSQL. Для более удобной работы с PostgreSQL имеет смысл добавить каталоги с его приложениями и руководством в соответствующие пути поиска. В стандартной UNIX-оболочке для этого следует добавить следующие команды в свой стартовый файл в домашнем каталоге (`.profile` или `.bashrc`):

```
PATH=$PATH:/usr/local/pgsql/bin
MANPATH=$MANPATH:/usr/local/pgsql/man
export PATH MANPATH
```

Как уже упоминалось, исходные тексты текущей и последней из тестовых версий PostgreSQL можно получить на сайте *http://www.postgresql.org*. Другие источники сведений о PostgreSQL приведены в главе 18.

Установка PostgreSQL в Windows

Начнем этот раздел с хорошей новости для пользователей Windows. Хотя PostgreSQL и был разработан для UNIX-подобных платформ, при

его написании учитывались соображения переносимости. Сначала появилась возможность писать клиентские приложения PostgreSQL для Windows, а начиная с версии 7.1 сервер PostgreSQL может быть скомпилирован, установлен и запущен в системе Microsoft Windows NT4 и Windows 2000. Для этого придется установить дополнительное программное обеспечение, реализующее некоторые возможности UNIX в Windows.

Порядок установки во многом совпадает с тем, который применялся при установке на Linux и UNIX из исходных текстов. Отличия заключаются в том, что придется установить UNIX-подобную среду разработки в Windows и выполнить пару нетривиальных действий, чтобы процесс `postmaster` запускался автоматически. Тем не менее при необходимости это может быть сделано. В действительности, для целей тестирования новой базы данных и решения проблем с сетевыми соединениями бывает весьма полезно иметь такую утилиту PostgreSQL, как `psql`, на той же машине, на которой установлено клиентское приложение, даже если сам сервер не должен работать в Windows.

Cygwin – UNIX-среда для Windows

В помощь тем, кто хотел бы расширить возможности Microsoft Windows, компания Cygnus Solutions (теперь ставшая частью Red Hat) разработала набор библиотек, эмулирующих некоторые UNIX API в среде Windows NT и 2000. Эти DLL позволяют программам, написанным для UNIX или Linux, выполняться в Windows. Имеется ряд ограничений, в особенности в части прав доступа и пользовательских привилегий, связанных с различием архитектур операционных систем, но, тем не менее, это весьма полезное дополнение к открытому программному обеспечению.

Пакет называется Cygwin и может быть свободно получен с сайта *http://www.cygwin.com*. В этом разделе будет показано, как установить Cygwin на платформе Windows и использовать его для компиляции исходных текстов PostgreSQL.

Прежде всего, зайдите на сайт *http://www.cygwin.com* и пойдите по ссылке «Install Cygwin now» (рис. 3.3).

Рис. 3.3. Установка Cgwin

Это ссылка на программу установки, которую можно загрузить и выполнить на локальной машине, а можно запустить прямо с сайта. Но мы рекомендуем загрузить ее на Windows-машину и запускать с локального жесткого диска.

Полная дистрибутивная копия Cygwin имеет размер более 40 Мбайт.

Запустите программу `setup.exe`, чтобы начать установку (рис. 3.4).

Программа содержит пошаговые инструкции по загрузке и установке многочисленных пакетов, входящих в набор инструментов Cygwin.

На следующем шаге надо выбрать стратегию установки. Можно загружать файлы в процессе установки. Можно сначала только загрузить файлы, а устанавливать позже (рис. 3.5). Мы рекомендуем второй способ, в этом случае файлы можно будет использовать для повторной установки или для установки на другой машине.

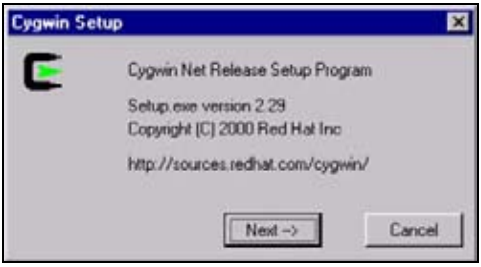

Рис. 3.4. Начало установки

Рис. 3.5. Выбор способа установки

После загрузки всех файлов из Интернета следует еще раз запустить программу `setup.exe` и выбрать пункт Install from Local Directory, указав местоположение загруженных файлов. Программа предлагает выбрать каталог для загружаемых файлов и задает вопрос о типе установленного соединения (рис. 3.6). Если установлен броузер Internet Explorer 5 и выше, то программа `setup.exe` может руководствоваться его настройками.

Рис. 3.6. Выбор типа соединения

Теперь надо выбрать сайт, с которого будет выполняться загрузка. Имеется ряд сайтов, расположенных по всему миру (рис. 3.7), которые хранят копии Cygwin, что позволяет избежать перегрузок на главном сайте. Рекомендуется выбирать сайт, географически наиболее близкий.

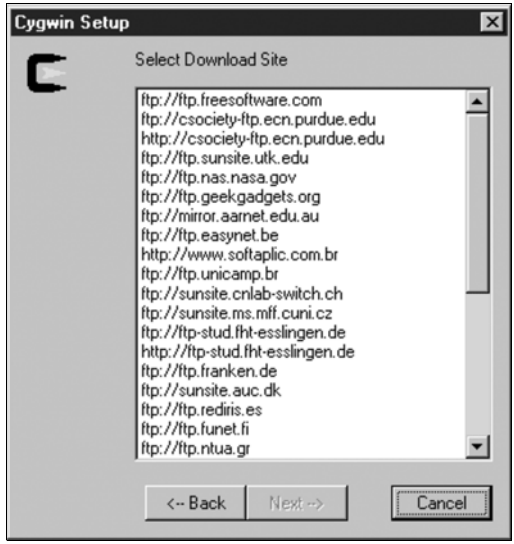

Рис. 3.7. Выбор сайта

Следующий шаг заключается в выборе пакетов для загрузки и установки из большого числа имеющихся в Cygwin (рис. 3.8). По умолчанию копируются только откомпилированные пакеты, хотя исходные тексты тоже доступны. Для компиляции PostgreSQL вполне подойдет выбор по умолчанию.

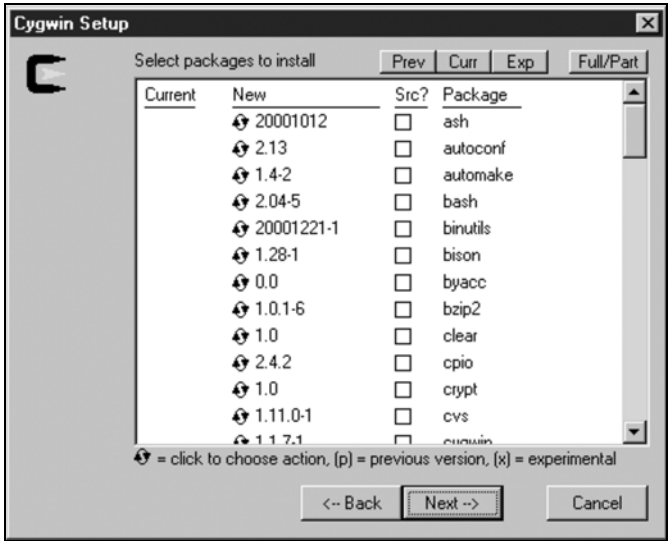

Рис. 3.8. Выбор пакетов

Если вы уверены, что какой-либо из пакетов вам не понадобится, можете пропустить его, установив признак `skip`. Если доступны несколь-

ко версий пакета, можно выбрать любую из них. Как правило, лучше выбирать более позднюю версию. Теперь программа `setup.exe` может начать загрузку (рис. 3.9):

Рис. 3.9. Загрузка началась...

По окончании загрузки можно запустить `setup.exe` еще раз и выбрать установку из локального каталога, в котором находятся загруженные файлы (рис. 3.10):

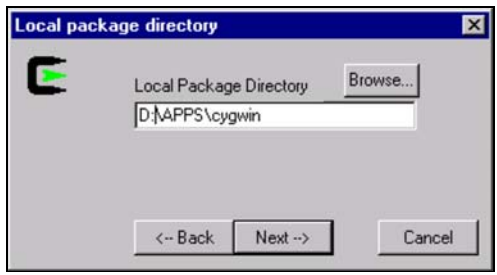

Рис. 3.10. Локальный каталог

Теперь пришло время выбрать место для установки Cygwin. В процессе установки Cygwin создает дерево каталогов, аналогичное традиционно используемому в UNIX и Linux. В нем присутствует корневой каталог с подкаталогами `bin`, `lib`, `usr` и т. д. Это дерево будет помещено в указанный каталог файловой системы NT/2000 (рис. 3.11):

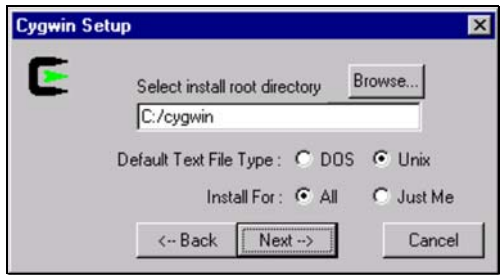

Рис. 3.11. Выбор каталога установки

Мы рекомендуем выбрать значение UNIX для типа текстовых файлов по умолчанию (Default Text File Type), поскольку в этом случае фай-

лы, создаваемые одними программами Cygwin, наверняка смогут быть прочитаны другими его программами.

При необходимости редактирования файлов Cygwin в приложениях Windows предпочтительно использовать редакторы, распознающие различные форматы текстовых файлов и умеющие с ними правильно обращаться. Хорошим выбором будет файловый редактор программиста (Programmer's File Editor, PFE), доступный на многих сайтах со свободно распространяемыми программами. Домашняя страница PFE находится по адресу *http://www.lancs.ac.uk/people/cpaap/pfe*. Имеются также версии замечательных UNIX-редакторов `vi` и `emacs` для Cygwin.

Если выбрать вариант установки Install For All (Устанавливать для всех), группа программ Cygwin будет доступна всем пользователям Windows, в противном случае – только вам. Рекомендуем выбрать установку для всех, чтобы пользователь `postgres` мог запускать программы Cygwin.

И наконец, надо выбрать программы, которые будут установлены; по умолчанию устанавливаются все загруженные пакеты.

По окончании установки на рабочем столе появятся значок и программная группа Cygnus (рис. 3.12), запускающая UNIX-оболочку в среде Windows:

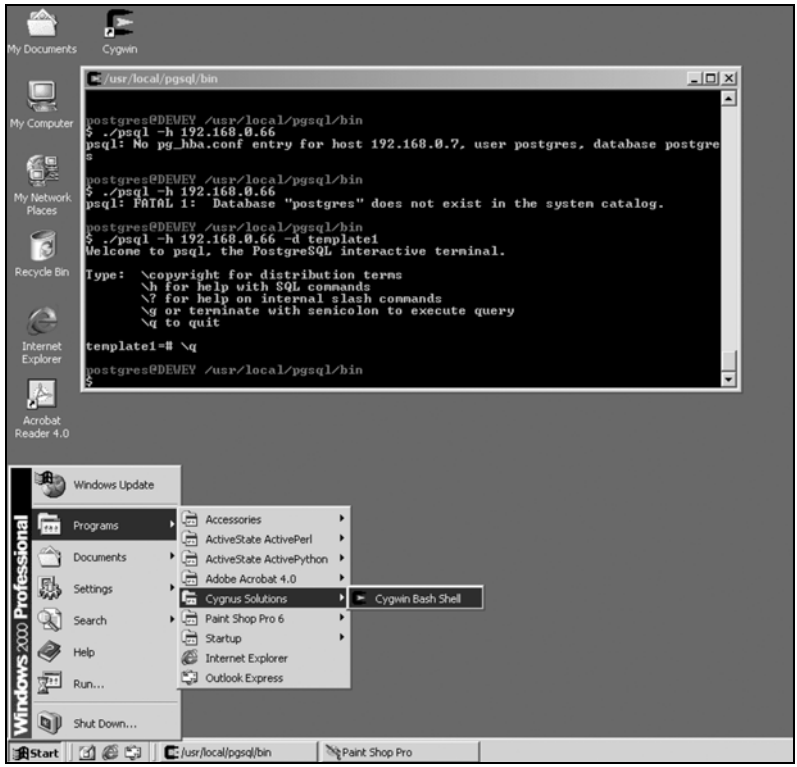

Рис. 3.12. Обновленный рабочий стол

Среди установленных программ Cygwin имеются компилятор GNU C, командная оболочка Bash и практически все необходимые средства для компиляции исходных текстов PostgreSQL.

Службы IPC для Windows

Еще один компонент программного обеспечения, необходимого для работы PostgreSQL под Windows, – это менеджер IPC, предназначенный для управления разделяемой памятью и семафорами. Соответствующий пакет, не входящий по умолчанию в Cygwin, называется CygIPC и доступен по адресу *http://www.neuro.gatech.edu/users/cwilson/cygutils/V1.1/cygipc*.

Установка CygIPC не вызывает проблем. Просто загрузите пакет и распакуйте его в корневом каталоге Cygwin:

```
$ cd /
$ tar zxvf cygipc-1.09-2.tar.gz
```

PostgreSQL для Cygwin

Начиная с версии 2.29 в Cygwin в качестве дополнительного пакета входит дистрибутив PostgreSQL, который может быть проинсталлирован программой установки Cygwin. Программы PostgreSQL помещаются в каталог `/usr/bin`, библиотеки – в `/usr/lib`, а разделяемые файлы – в `/usr/share`.

Компиляция PostgreSQL в Windows

Последовательность шагов по компиляции PostgreSQL в Windows NT/2000 практически та же, что и в Linux и UNIX, поэтому рассмотрим ее вкратце.

Скопируйте архив с исходными текстами (`postgresql-7.1.2.tar.gz`) в подкаталог корневого каталога Cygwin, например `/usr/src`.

> ***Корневой каталог Cygwin доступен из программы Windows Explorer как каталог, в который установлен Cygwin.***

Распакуйте исходные тексты и запустите `configure` и `make`, как и в случае с UNIX:

```
$ tar zxvf postgresql-7.1.2.tar.gz
$ cd postgres-7.1.2
$ ./configure
$ make
$ make install
```

Теперь PostgreSQL установлена в каталоге `/usr/local/pgsql`.

Конфигурирование PostgreSQL в Windows

Так же как в UNIX и Linux, серверный процесс PostgreSQL `postmaster` должен выполняться под специальным пользователем. Поэтому начнем с того, что создадим пользователя `postgres`.

Запустите приложение «Пользователи и пароли» панели управления, чтобы создать пользователя `postgres` как члена группы администраторов. Назначьте пароль обычным образом. Пароль необходим для запуска служб Windows, поэтому советуем установить для него неограниченный срок действия.

Теперь вы можете зарегистрироваться как пользователь `postgres` и запустить в Cygwin сессию `bash`.

Прежде чем запускать программу `postmaster`, необходимо запустить службу, реализующую IPC UNIX-типа. Это делается командой

```
$ /usr/local/bin/ipc-daemon.exe &
```

Теперь можно инициализировать базу данных – так же, как в среде UNIX и Linux:

```
pg$ /usr/local/pgsql/bin/initdb -D /usr/local/pgsql/data
```

Чтобы разрешить удаленный доступ, необходимо перед запуском программы `postmaster` отредактировать файл `pg_hba.conf`, добавив в конце строку:

```
host    all    192.168.0.0    255.255.0.0    trust
```

Это означает, что все компьютеры, IP-адреса которых начинаются с 192.168, могут получить доступ к базе данных.

Запустите серверный процесс той же командой, что и раньше:

```
pg$ /usr/local/pgsql/bin/postmaster -i -D /usr/local/pgsql/data
>logfile 2>&1 &
```

Как и раньше, можно создавать пользователей и базы данных:

```
$ /usr/local/pgsql/bin/createuser neil
$ /usr/local/pgsql/bin/createdb test
```

Используя программу `psql` локально или удаленно, можно проверить, запущена ли база данных. На снимках экрана показано, как PostgreSQL выполняется под Windows 2000, а доступ осуществляется с Linux-машины (рис. 3.13 и 3.14).

То же самое можно сделать в Windows, запустив в Cygwin вторую сессию `bash` и выполнив команду

```
$ /usr/local/pgsql/bin/psql -d test
```

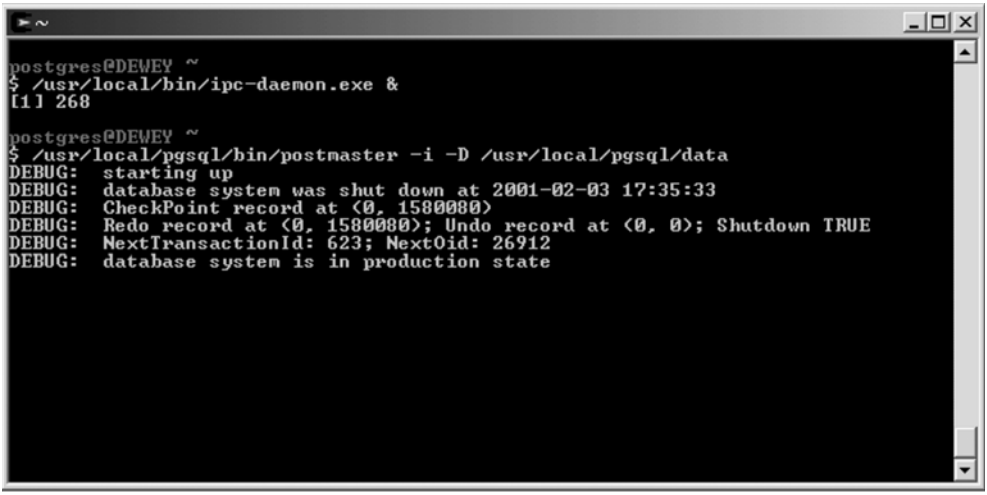

Рис. 3.13. Запуск PostgreSQL в Windows

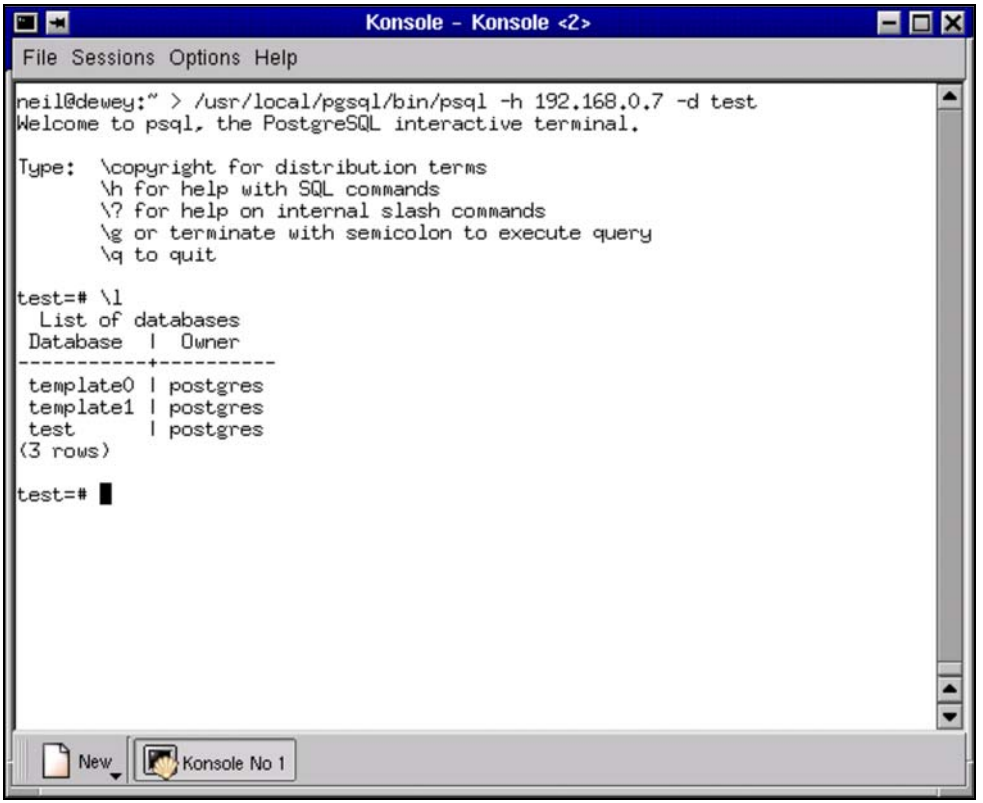

Рис. 3.14. Удаленный доступ к PostgreSQL с Linux-машины

Автоматический запуск PostgreSQL

Вот вы и научились запускать сервер PostgreSQL вручную. Теперь, если имеется компьютер под управлением Windows NT или 2000, выделенный под PostgreSQL, и вы готовы регистрироваться как пользователь `postgres` и запускать два процесса (`ipc-daemon` и `postmaster`) при каждом перезапуске системы, можно все оставить как есть.

Но если вы попробуете выйти из системы и войти как обычный пользователь, то обнаружите, что запущенные вами процессы остановлены.

Однако это можно исправить, заставив Windows автоматически запускать наши процессы. Для этого запустим их как службы Windows. В идеале программы должны быть специальным образом сконструированы, чтобы выполняться как службы, но даже если это не так, можно заставить Windows считать их таковыми.

Возможно, в будущих версиях PostgreSQL появится двоичный модуль, который будет устанавливаться как служба. Пожалуй, стоит следить за последними разработками, появляющимися на сайте.

А пока нам придется воспользоваться двумя небольшими программами: `INSTSRV` и `SRVANY`. Они входят в Windows NT4 Resource Kit и позволяют любую программу запустить в качестве службы. Они работают также и в Windows 2000. Исправленная версия `SRVANY` находится по адресу *ftp://ftp.microsoft.com/bussys/winnt/winnt-public/reskit/nt40/*.

Приведенные ниже инструкции требуют работы с редактором реестра. Редактирование реестра Windows – дело рискованное, т. к. цена ошибки может быть высока. Если вы не уверены в своих действиях, создайте резервную копию своих данных перед внесением изменений.

Скопируйте `instsrv.exe` и `srvany.exe` в любой локальный каталог. Программа `srvany.exe` всегда должна быть запущена во время старта служб PostgreSQL. Далее будем предполагать, что эти программы установлены в каталоге `C:\NTRESKIT`.

Создадим две новые службы в Windows, каждая из которых будет запускать утилиту `srvany.exe`, которая, в свою очередь, и запустит программу, выполняющуюся в качестве службы.

Начнем с более простой программы – демона IPC.

> ***Во время написания этой книги уже велись работы по модификации демона IPC, чтобы он мог запускаться как служба без помощи программ*** `INSTSRV` ***и*** `SRVANY`***. Если вы работаете с версией*** `cygipc` ***более поздней чем 1.09, обратитесь к файлу README за разъяснениями.***

Во-первых, необходимо создать новую службу, запустив утилиту `instsrv.exe`:

```
C:\NTRESKIT> instsrv "IPC Daemon"
c:\ntreskit\srvany.exe
```

```
The service was successfully added!

Make sure that you go into the Control Panel and use
the Services applet to change the Account Name and
Password that this newly installed service will use
for its Security Context.

C:\NTRESKIT>
```

Прежде чем редактировать реестр, добавляя туда параметры новой службы, необходимо настроить ее так, чтобы она запускалась под пользователем `postgres`. Для этого запустите оснастку Службы (Services) из состава средств Консоли управления (MMC).[1] В параметрах регистрации службы укажите имя пользователя `postgres` и пароль, выбранный ранее при создании пользователя (рис. 3.15). Поскольку пользователь зарегистрирован на локальной машине, а не в домене, то имя отображается в виде `.\postgres`, как это показано на рис. 3.15:

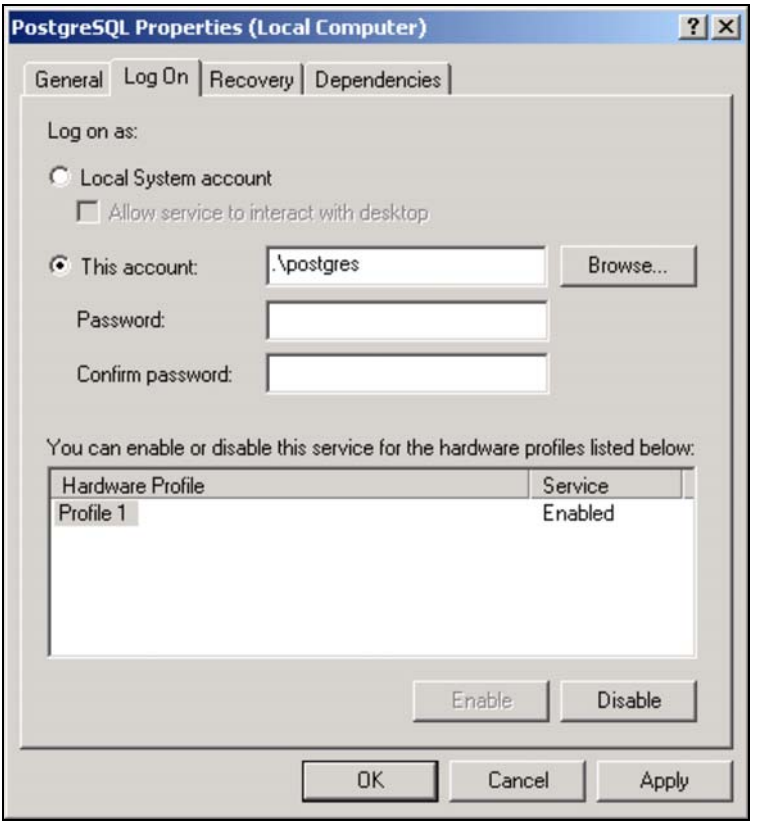

Рис. 3.15. Свойства PostgreSQL

[1] По цепочке Панель управления ▸ Администрирование ▸ Службы и приложения ▸ Службы. – *Примеч. ред.*

Теперь запустим редактор реестра `regedit.exe`, чтобы отредактировать запись о только что созданном демоне IPC. Необходимо указать параметры для утилиты `srvany.exe`, чтобы она запускала программу `ipc-daemon.exe` от нашего имени.

Запустите `regedit.exe` и перейдите в раздел

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services
```

Вы увидите новый элемент с именем `IPC Daemon`. Внутри него создайте ключ с именем `Parameters`, а в нем – два строковых параметра, `Application` и `AppDirectory`. Укажите в них полные пути программы `ipc-daemon.exe` и каталога `/bin`, используемого Cygwin.

Например, это могут быть `F:\cygwin\usr\local\bin\ipc-daemon.exe` и `F:\cygwin\bin` соответственно.

Результат должен выглядеть приблизительно так (рис. 3.16):

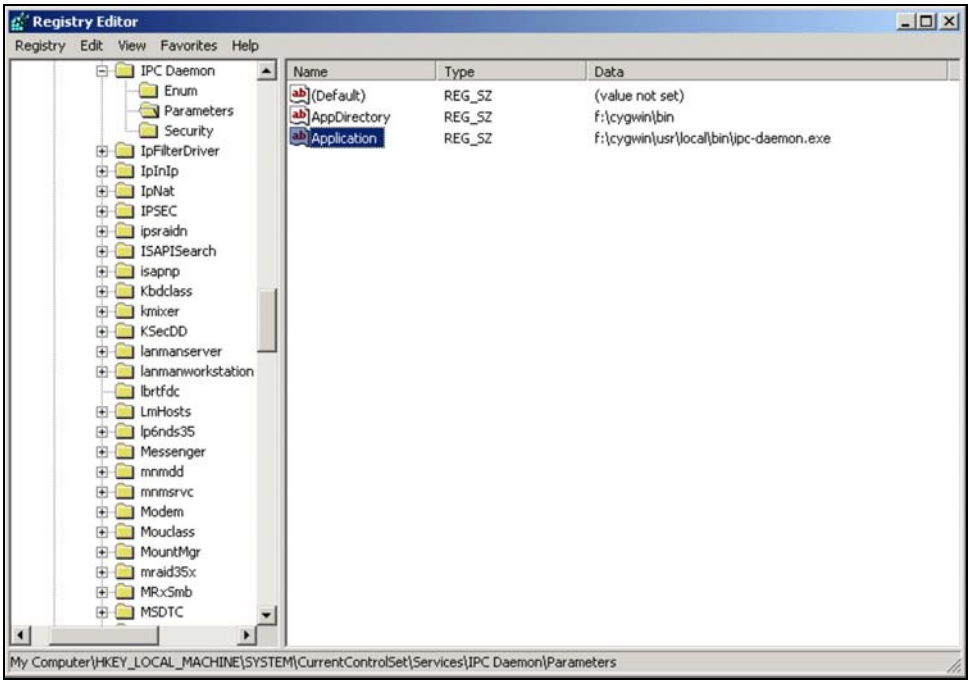

Рис. 3.16. Редактор реестра

Теперь можно запустить оснастку Службы (Services) из группы инструментов Администрирование (Administrative Tools) Панели управления (Control Panel) Windows 2000.

Программа `ipc-daemon.exe` должна стартовать, после чего она будет видна в окне Диспетчер задач Windows (Windows Task Manager) (рис. 3.17):

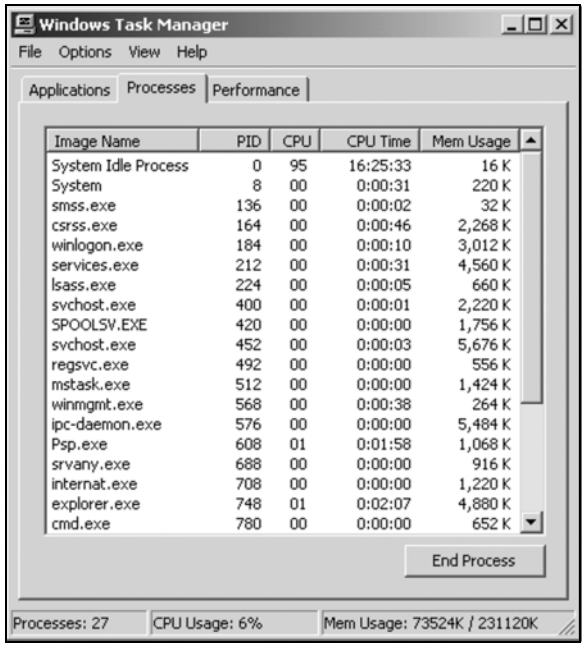

Рис. 3.17. Диспетчер задач Windows

Теперь перейдем к более сложной процедуре запуска процесса `postmaster`. Начнем также с создания службы `SRVANY`.

```
C:\NTRESKIT> instsrv "PostgreSQL" c:\ntreskit\srvany.exe
```

Прежде чем добавлять в реестр параметры для запуска этой службы, необходимо, как и для демона IPC, настроить ее запуск под пользователем `postgres`. Для этого запустите оснастку Службы и, как и раньше, укажите имя пользователя `.\postgres` и соответствующий пароль.

Для того чтобы запустить процесс `postmaster` в UNIX-подобной среде Cygwin, придется предварительно запустить пару командных процессоров. Используем `srvany.exe` для того, чтобы запустить `cmd.exe` – командную строку Windows. Ей передадим параметр, заставляющий ее запустить другую программу. Этой программой будет командный процессор Cygwin, который, в свою очередь, выполнит сценарий с командами оболочки. Этот-то сценарий и запустит процесс `postmaster`. Все не так уж сложно, если действовать последовательно.

Создайте сценарий для `bash` в каталоге Cygwin `/usr/local/bin` и назовите его `startpg`. Этот текстовый (в формате UNIX) файл должен содержать строки:

```
#!/bin/bash

# Start PostgreSQL

PGDATA=/usr/local/pgsql/data
PATH=/bin:/usr/bin:/usr/local/bin:/usr/local/pgsql/bin
export PGDATA PATH

postmaster -i >/usr/local/pgsql/postmaster.log 2>&1 </dev/null
```

Этот сценарий будет запускать PostgreSQL.

Теперь можно воспользоваться редактором `regedit.exe` для настройки параметров запуска. Как и раньше, перейдем в раздел `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services`. В нем должен появиться элемент `PostgreSQL`.

Создадим новый ключ с именем `Parameters`, а в нем – три строковых параметра со значениями, представленными в табл. 3.4:

Таблица 3.4. Компоненты ключа Parameters

Строка	Параметр
Application	`cmd.exe`
AppDirectory	`f:\cygwin\bin`
AppParameters	`/c "f:\cygwin\bin\bash.exe /usr/local/bin/startpg"`

Теперь можно запускать службу PostgreSQL. Используйте для этого оснастку Службы и диспетчер задач, чтобы убедиться, что обе службы запущены. Вы увидите, что задача `postmaster` в действительности называется `postgres.exe`, т. к. именно эту программу запускает сценарий `postmaster`.

В качестве завершающего теста перезагрузите машину: вы должны увидеть в диспетчере задач, что обе службы запущены и база данных PostgreSQL доступна для пользователя, отличного от `postgres`.

Пожалуй, это все.

Резюме

В данной главе рассмотрено несколько вариантов установки PostgreSQL. Самый простой способ – установка из предварительно откомпилированных пакетов. А поскольку PostgreSQL – продукт с открытым исходным кодом, то при необходимости он может быть откомпилирован пользователем в любой UNIX-совместимой системе.

Приведена последовательность шагов по компиляции, установке и проверке работоспособности с использованием пакетов в Linux, исходных текстов в UNIX и даже в Windows NT/2000 с Cygwin.