Руководство администратора Firebird 3.0

Содержание

Введение

Глава 1. Установка, обновление и запуск сервера

Скачать дистрибутивы Firebird можно на официальном сайте https://firebirdsql.org в разделе Downloads.

1.1. Установка на ОС Linux

Firebird под Linux распространяется в двух вариантах через репозитории и в виде tar-архивов. За репозиториями следят либо разработчики дистрибутивов Linux, либо частные лица (PPA). Tar-архивы распространяет организация Firebird Foundation. Для официальных релизов Firebird 3.0 они выложены на странице https://firebirdsql.org/en/firebird-3-0/. Кроме того tar-архивы автоматически собираются для снапшотов, которые вы можете найти по адресу http://web.firebirdsql.org/download/snapshot_builds/linux/fb3.0/.

1.1.1. Установка из репозитория

Репозитории — это сервера в интернете, на которых хранятся файлы пакетов приложений Linux и другая сопутствующая информация.

Практически у каждого дистрибутива Linux есть свой репозиторий, который содержит только совместимые и поддерживаемые конкретным дистрибутивом пакеты, соответственно, при установке приложений из официальных репозиториев Вы всегда устанавливаете только проверенные и стабильные версии программ.

Репозитории бывают как основные, т.е. официально поддерживаемые, так и дополнительные, которые можно подключить в случае возникновения необходимости. Firebird редко включён в основной репозиторий, но часто присутствует в дополнительных репозиториях или в PPA (Personal Package Archive).

За актуальностью пакетов в репозиториях следят либо разработчики дистрибутивов (Linux), либо владельцы репозиториев, а не сообщество Firebird. По этой причине в репозиториях может быть не самая последняя версия Firebird. Если вас надо гарантировано установить именно последнюю версию Firebird, либо вовсе произвести обновление из снапшотов, то рекомендуем вам производить установку Firebird из tar-архива.

Установка Firebird в Ubuntu

В качестве примера рассмотрим установку Firebird 3.0 в Ubuntu 18.04 TLS.

Стабильный пакет (версия 3.0.x) для Ubuntu TLS расположен в репозитории ppa.

Данный репозиторий может быть добавлен следующим способом:

```
sudo add-apt-repository ppa:mapopa/firebird3.0
sudo apt-get update
```

Вы можете посмотреть пакеты связанный с Firebird 3.0 следующим образом:

```
apt-cache search firebird3.0
```

Вы уведите следующий список пакетов:

```
firebird-dev - Development files for Firebird
firebird3.0-common - common files for firebird 3.0 server, client and utilities
firebird3.0-common-doc - copyright, licensing and changelogs of firebird3.0
firebird3.0-doc - Documentation files for firebird database version 3.0
firebird3.0-examples - Examples for Firebird database
firebird3.0-server-core - Firebird engine core
flamerobin - graphical database administration tool for Firebird DBMS
libib-util - Firebird UDF support library
firebird3.0-server - Firebird Server - an RDBMS based on InterBase 6.0 code
firebird3.0-utils - Firebird command line utilities
libfbclient2 - Firebird client library
```

Для установки Firebird 3.0 в архитектуре SuperServer наберите следующую команду:

```
sudo apt-get install firebird3.0-server
```

```
The following extra packages will be installed:
  firebird3.0-common firebird3.0-common-doc firebird3.0-utils libfbclient2 libib-util
Suggested packages:
  firebird3.0-doc
The following NEW packages will be installed:
  firebird3.0-common firebird3.0-common-doc firebird3.0-server firebird3.0-utils
libfbclient2 libib-util
0 upgraded, 6 newly installed, 0 to remove and 0 not upgraded.
Need to get 4,889kB of archives.
After this operation, 13.1MB of additional disk space will be used.
Do you want to continue [Y/n]?
```

В процессе установки у вас спросят о пароле для пользователя SYSDBA.

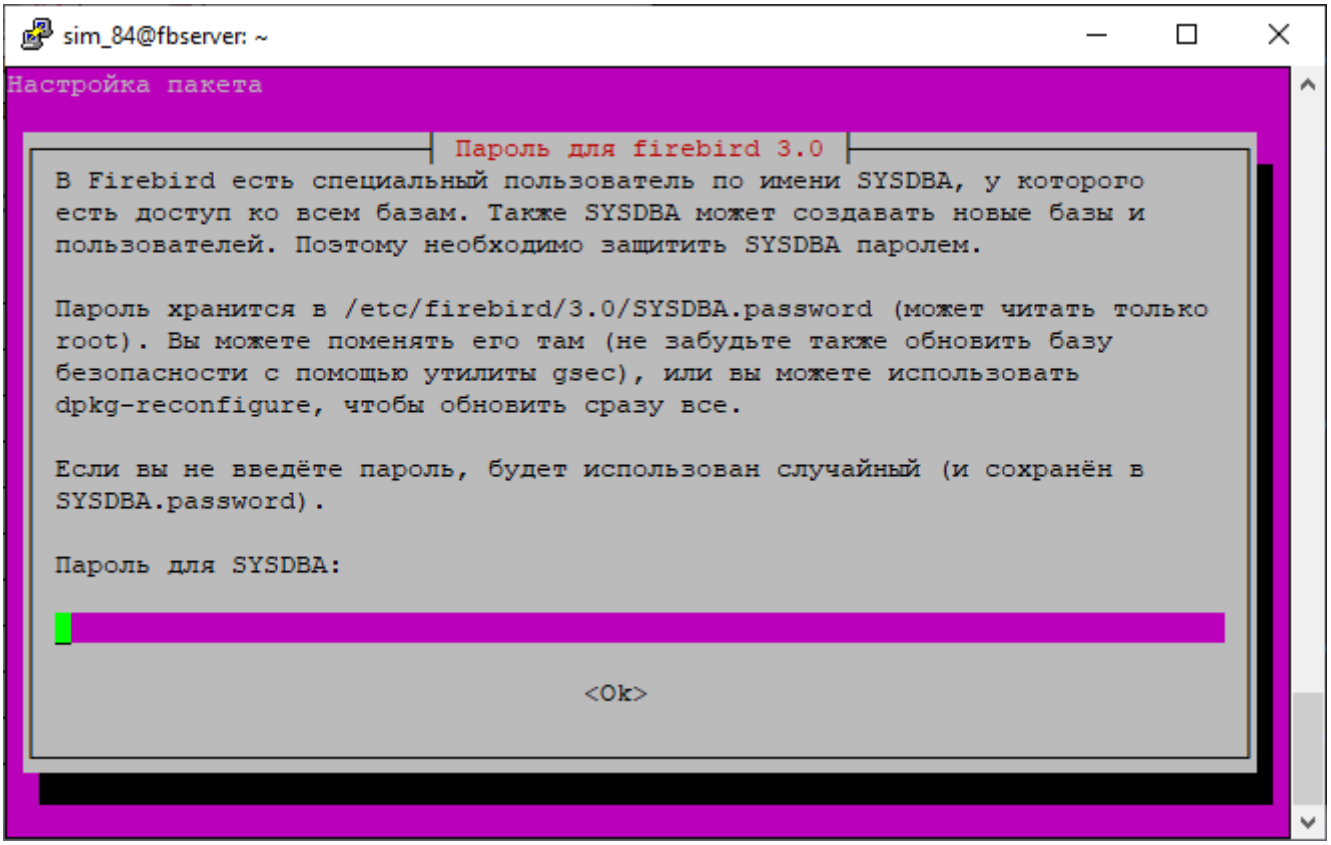

Рисунок 1. Задание пароля SYSDBA

Если вам необходимо изменить конфигурацию после установки воспользуйтесь командой:

```
sudo dpkg-reconfigure firebird3.0-server
```

Для установки примеров и файлов для разработки наберите команду:

```
sudo apt-get install firebird3.0-examples firebird-dev
```

База данных *employee.fdb* архивирована и находится в директории */usr/share/doc/firebird3.0-examples/examples/empbuild/*

```
cd /usr/share/doc/firebird3.0-examples/examples/empbuild/
sudo gunzip employee.fdb.gz
sudo chown firebird.firebird employee.fdb
sudo mv employee.fdb /var/lib/firebird/3.0/data/
```

Это описание заимствовано со следующего ресурса https://help.ubuntu.com/community/Firebird3.0. Когда я пробовал проделать тоже самое, то по какой-то причине у меня не оказалось каталога */usr/share/doc/firebird3.0-examples/examples/empbuild/*

К счастью в папке */usr/share/doc/firebird3.0-examples/examples/* лежал запакованный sql скрипт для создания базы данных employee.

```
cd /usr/share/doc/firebird3.0-examples/examples/
sudo gunzip employee.sql.gz
```

Полученный файл *employee.sql* требуется немного поправить чтобы база данных создавалась в нужном месте.

```
-- This is a generated file
set sql dialect 3;
create database '/var/lib/firebird/3.0/data/employee.fdb';
...
```

После чего можно запустить скрипт на выполнение:

```
isql-fb -i employee.sql
```

Теперь можно попробовать подключится к нашей тестовой базы данных.

```
$ isql-fb
SQL> connect localhost:/var/lib/firebird/3.0/data/employee.fdb user 'SYSDBA' password
password 'Xtwe9k';
```

Если все в порядке, то вы получите сообщение о том, к какой базе данных подключен, какой пользователь и о готовности к использованию.

```
Database: localhost:/var/lib/firebird/3.0/data/employee.fdb, User: SYSDBA
SQL>
```

Теперь проверим версию серверы и таблицы.

```
SQL> show tables;
       COUNTRY                         CUSTOMER
       DEPARTMENT                      EMPLOYEE
       EMPLOYEE_PROJECT                JOB
       PROJECT                         PROJ_DEPT_BUDGET
       SALARY_HISTORY                  SALES

SQL> show version;
ISQL Version: LI-V3.0.5.33100 Firebird 3.0
Server version:
Firebird/Linux/AMD/Intel/x64 (access method), version "LI-V3.0.5.33100 Firebird 3.0"
Firebird/Linux/AMD/Intel/x64 (remote server), version "LI-V3.0.5.33100 Firebird
3.0/tcp (fbserver)/P15:C"
Firebird/Linux/AMD/Intel/x64 (remote interface), version "LI-V3.0.5.33100 Firebird
3.0/tcp (fbserver)/P15:C"
on disk structure version 12.0
```

Если ваш сервер установлен с графическим окружением, то вы также можете установить GUI инструмент администрирования `flamerobin` простой командой.

```
sudo apt-get install flamerobin
```

Установка только клиента Firebird

Если необходимо установить только клиента Firebird без сервера, то процедура немного сокращается.

Первым делом добавьте репозиторий ppa:mapopa/firebird3.0

```
sudo add-apt-repository ppa:mapopa/firebird3.0
sudo apt-get update
```

Теперь устанавливаем сам клиент:

```
sudo apt-get install libfbclient2
```

Установка Firebird в CentOS

В качестве примера рассмотрим установку Firebird 3.0 в CentOS 8.0.

Для установки Firebird 3.0 необходимо подключить репозиторий epel. Это делается следующей командой:

```
sudo dnf install https://dl.fedoraproject.org/pub/epel/epel-release-latest-
8.noarch.rpm
```

Для просмотра информации о версии Firebird доступным для установки введите команду:

```
sudo yum info firebird
```

```
Последняя проверка окончания срока действия метаданных: 0:01:19 назад, Ср 08 янв 2020
11:46:11.
Имеющиеся пакеты
Имя          : firebird
Версия       : 3.0.4.33054
Выпуск       : 1.el8
Архитектура  : x86_64
Размер       : 3.6 M
Источник     : firebird-3.0.4.33054-1.el8.src.rpm
Репозиторий  : epel
Краткое опис : SQL relational database management system
URL          : http://www.firebirdsql.org/
Лицензия     : Interbase
Описание     : Firebird is a relational database offering many ANSI SQL standard
             : features that runs on Linux, Windows, and a variety of Unix
             : platforms. Firebird offers excellent concurrency, high
             : performance, and powerful language support for stored procedures
             : and triggers. It has been used in production systems, under a
             : variety of names, since 1981.
```

Как видим в репозитории доступна довольно свежая версия. Можно приступать к установке.

```
sudo yum install firebird
```

```
Последняя проверка окончания срока действия метаданных: 0:00:43 назад, Ср 08 янв 2020
11:51:23.
Зависимости разрешены.
================================================================================
 Пакет                 Архитектура   Версия                    Репозиторий Размер
================================================================================
Installing:
 firebird              x86_64        3.0.4.33054-1.el8         epel        3.6 M
Установка зависимостей:
 firebird-utils        x86_64        3.0.4.33054-1.el8         epel        1.1 M
 libfbclient2          x86_64        3.0.4.33054-1.el8         epel        592 k
 libib-util            x86_64        3.0.4.33054-1.el8         epel         18 k
 libtommath            x86_64        1.1.0-1.el8               epel         47 k

Результат транзакции
================================================================================
Установка  5 Пакетов

Объем загрузки: 5.4 M
Объем изменений: 23 M
Продолжить? [д/Н]: y
Загрузка пакетов:
(1/5): libfbclient2-3.0.4.33054-1.el8.x86_64.rp 372 kB/s | 592 kB     00:01
(2/5): libib-util-3.0.4.33054-1.el8.x86_64.rpm  107 kB/s |  18 kB     00:00
(3/5): firebird-utils-3.0.4.33054-1.el8.x86_64. 617 kB/s | 1.1 MB     00:01
(4/5): libtommath-1.1.0-1.el8.x86_64.rpm        451 kB/s |  47 kB     00:00
(5/5): firebird-3.0.4.33054-1.el8.x86_64.rpm    1.5 MB/s | 3.6 MB     00:02
--------------------------------------------------------------------------------
Общий размер                                    1.3 MB/s | 5.4 MB     00:04
```

Теперь необходимо задать пароль для пользователя SYSDBA. Я предпочитаю делать это через `isql`.

```
cd /usr/bin
sudo ./isql-fb
```

```
Use CONNECT or CREATE DATABASE to specify a database
SQL> connect security.db user sysdba;
Database: security.db, User: SYSDBA
SQL> create user sysdba password 'xRt3sd6T';
SQL> exit;
```

Обратите внимание, что утилита `isql` переименована в `isql-fb`. Это сделано потому, что в Linux уже есть утилита `isql` в рамках проекта unixODBC.

Осталось включить и запустить сервис firebird-superserver.

```
sudo systemctl enable firebird-superserver
```

```
Created symlink /etc/systemd/system/multi-user.target.wants/firebird-superserver.
service  → /usr/lib/systemd/system/firebird-superserver.service.
```

```
sudo systemctl start firebird-superserver
```

Установка только клиента Firebird

Если необходимо установить только клиента Firebird без сервера, то процедура немного сокращается.

Первым делом добавьте репозиторий EPEL

```
sudo dnf install https://dl.fedoraproject.org/pub/epel/epel-release-latest-
8.noarch.rpm
sudo dnf update
```

Теперь устанавливаем сам клиент:

```
sudo dnf install libfbclient2
```

1.1.2. Ручная установка из tar архива

Tar-архивы распространяет организация Firebird Foundation. Для официальных релизов Firebird 3.0 они выложены на странице https://firebirdsql.org/en/firebird-3-0/. Кроме того tar-архивы автоматически собираются для снапшотов, которые вы можете найти по адресу http://web.firebirdsql.org/download/snapshot_builds/linux/fb3.0/.

Стоит отметить что процедура установки из tar-архивов одинаковая для всех дистрибутивов Linux. Нюансы возникают только при разрешении зависимостей от некоторых библиотек.

При установке из tar-архива сервер будет установлен в директорию */opt/firebird*

В отличие от установки из репозиториев утилита `isql` не переименована и расположена в */opt/firebird/bin*.

Установка Firebird в Ubuntu

В качестве примера рассмотрим установку Firebird 3.0 в Ubuntu 18.04.

Перед установкой Firebird необходимо установить следующие пакеты

```
sudo apt-get install libtommath1
```

Для библиотеки libtommath необходимо также создать символическую ссылку:

```
sudo ln -sf /usr/lib/x86_64-linux-gnu/libtommath.so /usr/lib/x86_64-linux-
gnu/libtommath.so.0
```

Скачиваем и распаковываем tar-архив текущей версии Firebird

```
wget https://github.com/FirebirdSQL/firebird/releases/download/R3_0_5/Firebird-
3.0.5.33220-0.amd64.tar.gz
tar xvf Firebird-3.0.5.33220-0.amd64.tar.gz
cd Firebird-3.0.5.33220-0.amd64
```

Теперь можно запускать скрипт установки

```
sudo ./install.sh
```

В процессе инсталляции у вас спросят пароль для пользователя SYSDBA.

Обычно после этого служба Firebird стразу же установлена и запущена, вы можете проверить это командой:

```
systemctl status firebird-superserver
```

```
systemctl status firebird-superserver
● firebird-superserver.service - Firebird Database Server ( SuperServer )
   Loaded: loaded (/lib/systemd/system/firebird-superserver.service; enabled; ve
   Active: active (running) since Sun 2020-01-26 15:18:24 UTC; 8min ago
  Process: 6921 ExecStart=/opt/firebird/bin/fbguard -pidfile /var/run/firebird/f
 Main PID: 6923 (firebird)
    Tasks: 4 (limit: 4614)
   CGroup: /system.slice/firebird-superserver.service
           ├──6922 /opt/firebird/bin/fbguard -pidfile /var/run/firebird/firebird.
           └──6923 /opt/firebird/bin/firebird

янв 26 15:18:24 dbserver systemd[1]: Starting Firebird Database Server ( SuperSe
янв 26 15:18:24 dbserver systemd[1]: firebird-superserver.service: Can't open PI
янв 26 15:18:24 dbserver systemd[1]: firebird-superserver.service: Supervising p
янв 26 15:18:24 dbserver systemd[1]: Started Firebird Database Server ( SuperSer
```

Если это не так выполните

```
sudo systemctl enable firebird-superserver
sudo systemctl start firebird-superserver
```

Установка Firebird в CentOS

В качестве примера рассмотрим установку Firebird 3.0 в CentOS 8.0.

Для установки Firebird 3.0 необходимо подключить репозиторий epel. Это делается следующей командой:

```
sudo dnf install https://dl.fedoraproject.org/pub/epel/epel-release-latest-
8.noarch.rpm
```

Перед установкой Firebird необходимо установить следующие пакеты

```
sudo dnf -y install libtommath
sudo dnf -y install libicu-devel
sudo dnf -y install libncurses*
```

Для библиотеки libtommath необходимо также создать символическую ссылку:

```
sudo ln -sf /usr/lib64/libtommath.so.1 /usr/lib64/libtommath.so.0
```

Скачиваем и распаковываем tar-архив текущей версии Firebird

```
wget https://github.com/FirebirdSQL/firebird/releases/download/R3_0_5/Firebird-
3.0.5.33220-0.amd64.tar.gz
tar xvf Firebird-3.0.5.33220-0.amd64.tar.gz
cd Firebird-3.0.5.33220-0.amd64
```

Теперь можно запускать скрипт установки

```
sudo ./install.sh
```

В процессе инсталляции у вас спросят пароль для пользователя SYSDBA.

Обычно после этого служба Firebird стразу же установлена и запущена, вы можете проверить это командой:

```
systemctl status firebird-superserver
```

```
● firebird-superserver.service - Firebird Database Server ( SuperServer )
   Loaded: loaded (/usr/lib/systemd/system/firebird-superserver.service; enable>
   Active: active (running) since Sun 2020-01-26 18:41:14 MSK; 2h 24min left
  Process: 1105 ExecStart=/opt/firebird/bin/fbguard -pidfile /var/run/firebird/>
 Main PID: 1143 (firebird)
    Tasks: 5 (limit: 17728)
   Memory: 18.5M
   CGroup: /system.slice/firebird-superserver.service
           ├──1132 /opt/firebird/bin/fbguard -pidfile /var/run/firebird/firebird>
           └──1143 /opt/firebird/bin/firebird

янв 26 18:41:13 localhost.localdomain systemd[1]: Starting Firebird Database Se>
янв 26 18:41:13 localhost.localdomain systemd[1]: firebird-superserver.service:>
янв 26 18:41:14 localhost.localdomain systemd[1]: firebird-superserver.service:>
янв 26 18:41:14 localhost.localdomain systemd[1]: Started Firebird Database Ser>
```

Если это не так выполните

```
sudo systemctl enable firebird-superserver
sudo systemctl start firebird-superserver
```

Изменение режима работы Firebird

При установке сервера из tar-архива Firebird по умолчанию устанавливается в архитектуре SuperServer. Если вам необходимо изменить режим работы сервера, то сделать это можно при помощи скрипта */opt/firebird/bin/changeServerMode.sh*

```
sudo /opt/firebird/bin/changeServerMode.sh
```

```
Firebird server may run in 2 different modes - super and classic.
Super server provides better performance, classic - better availability.

Which option would you like to choose: (super|classic) [super] classic
Stopping currently running engine...
Starting firebird in classic server mode...
Updated /opt/firebird/firebird.conf
Done.
```

1.1.3. Ручная установка клиента

1.2. Установка на ОС Windows

Firebird под Windows распространяется в двух вариантах: в виде выполняемого файла установщика с расширением exe и в виде zip или 7z архивов. Для 32-разрядного варианта в имени файла содержится суффикс Win32, для 64-разрядного — x64. Отладочный вариант

имеет дополнительный суффикс pdb.

1.2.1. Установка из исполняемого пакета

Скачайте и запустите инсталяционный пакет разрядности, соответствующей разрядности операционной системы (Win32 может работать на 64-разрядной Windows, но это не имеет смысла). Инсталляция СУБД Firebird осуществляется с помощью стандартного мастера установки программ. В ходе установки мастер собирает всю необходимую для установки сервера информацию, производит копирование файлов и регистрацию программных модулей в реестре Windows.

Для установки Firebird необходимы права администратора.

Рекомендуется перед установкой этого пакета ДЕИНСТАЛЛИРОВАТЬ все предыдущие версии Firebird или InterBase. Особенно важно убедиться, что файлы *fbclient.dll* и *gds32.dll* удалены из каталога *system32*.

Если вам необходимо установить Firebird 3.0 совместно с другими версиями, то это необходимо сделать из zip-архива.

Выберите язык установки. Предусмотрена установка на русском, английском и других языках.

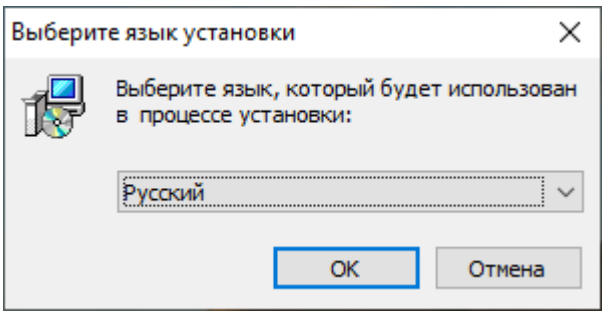

Рисунок 2. Выбор языка установки

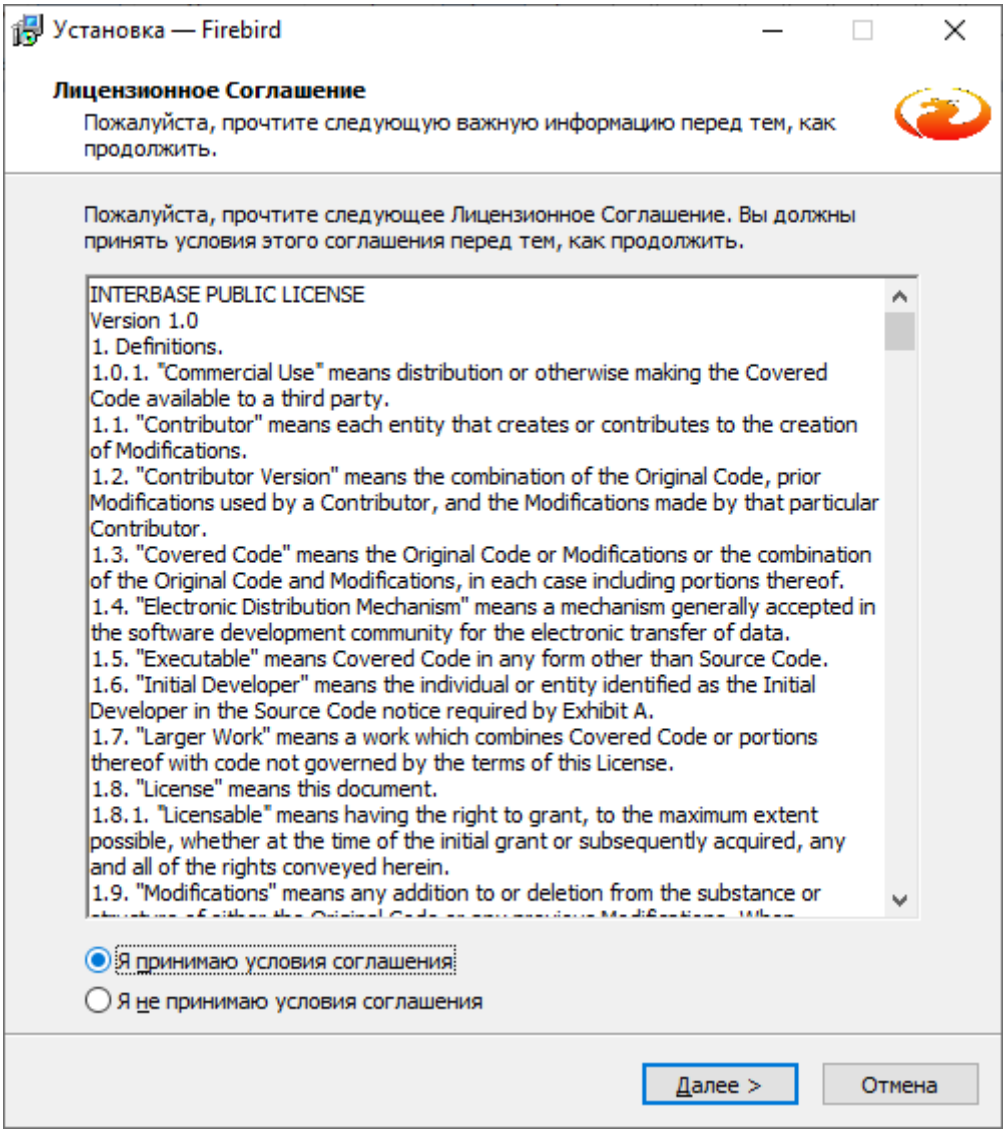

Рисунок 3. Лицензионное соглашение и информация об установке

В процессе инсталляции вам будет предложено выбрать папку для установки, а также выбрать устанавливаемые компоненты.

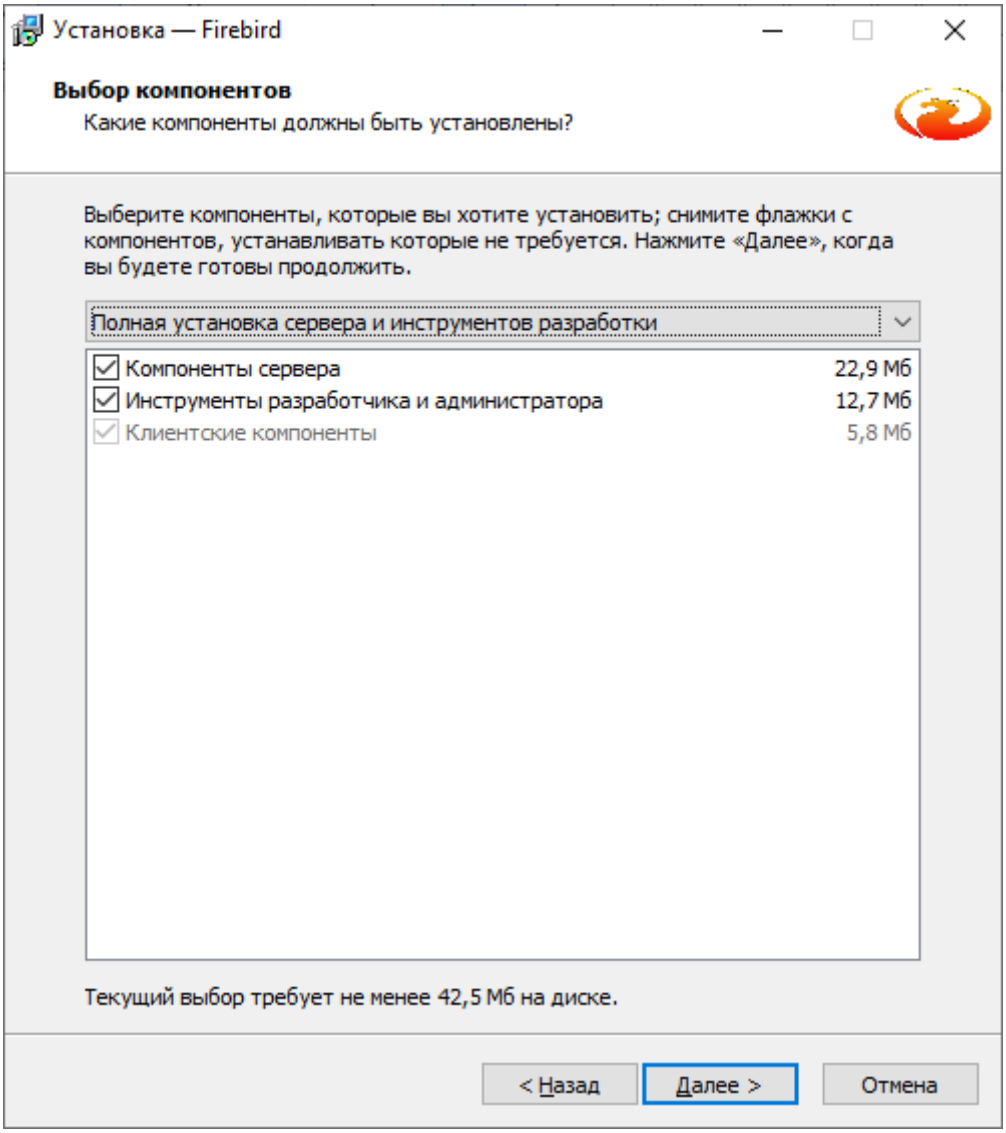

Рисунок 4. Выбор компонентов для установки

Выбор архитектуры сервера

Далее будет предложено выбрать архитектуру сервера и некоторые дополнительные опции. Доступны следующие архитектуры:

- **Classic**:
  - использует отдельный процесс на каждое пользовательское соединение;
  - каждый процесс содержит в себе все что нужно для работы с базой данных: область памяти для метаданных, страничный кэш для минимизации повторных чтений из файла БД; память для сортировок;
  - если происходит сбой, другие соединения остаются работоспособными
  - поддержка мультипроцессорности: в многопроцессорных системах ОС автоматически распределяет процессы по процессорам/ядрам
- **Superserver**:
  - один процесс с общей областью памяти для всех пользовательских соединений (общий страничный кеш и память под сортировки);
  - используется пул потоков ОС для обработки запросов от соединений, таким образом

каждое соединение работает в отдельном потоке управляемом ОС, а неактивные соединения не отъедают ресурсы потоков;

- каждый поток со своим кэшем метаданных;
- поддержка мультипроцессорности: потоки ОС легко распараллеливаются;
- возможный сбой в одном процессе разорвет все подключения.

- **SuperClassic**:
  - единый процесс на всех пользователей с общей памятью под сортировки;
  - используется пул потоков ОС для обработки запросов от соединений, таким образом каждое соединение работает в отдельном потоке управляемом ОС, а неактивные соединения не отъедают ресурсы потоков;
  - каждый поток со своим страничным кэшем и кэшем метаданных;
  - поддержка мультипроцессорности: потоки ОС легко распараллеливаются;
  - возможный сбой в одном процессе разорвет все подключения.

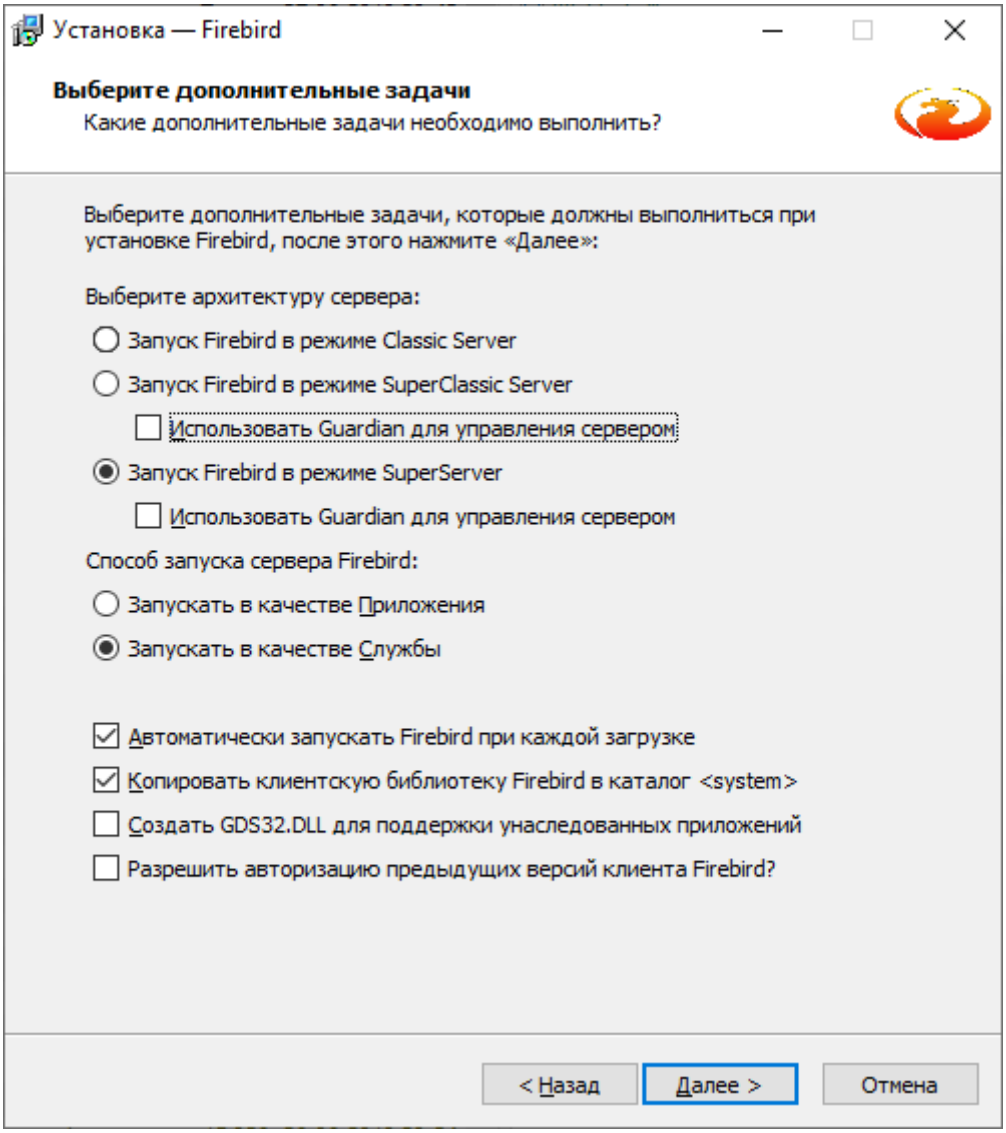

Рисунок 5. Выбор архитектуры

Каждый из режимов стабилен, и нет причин полностью отдавать предпочтение какому-то одному. Конечно, у вас могут быть свои собственные конкретные соображения. Если Вы

сомневаетесь, просто следуйте за установкой по умолчанию. Позже вы можете изменить архитектуру через файл конфигурации *firebird.conf* (параметр ServerMode), что потребует перезагрузки, но не переустановки.

Установка Guardian

Мы собираемся отказаться от использования Firebird Guardian в будущих версиях. Он не работает в случае архитектуры Classic Server и программа установки не предлагает его установку, если выбрана данная архитектура. Для архитектур SuperServer и SuperClassic установка Firebird Guardian возможна, но по умолчанию не выбрана.

Способ запуска сервера Firebird

Сервер может быть запущен в качестве приложения (менее предпочтительный вариант) или в качестве службы. Для запуска в качестве приложения используется следующая команда:

```
firebird -a
```

Установка gds32.dll

По умолчанию данная библиотека не копируется в системные папки Windows в процессе установки. Мы не можем гарантировать, что необходимые библиотеки MS VC runtime будут установлены и доступны в системе. Тем не менее, данная возможность доступна опционально в процессе установки, так же как и копирование *fbclient.dll* в системные папки Windows. Если вы используете данную возможность, то вы должны убедиться, что библиотеки MS VC runtime версии 10.0 установлены в системе.

Авторизация с предыдущих версий клиента Firebird

В Firebird 3.0 по умолчанию используется безопасная парольная аутентификация (SRP). Клиенты Firebird 2.5 и более ранние версии использовали традиционную аутентификацию (Legacy_Auth), которая отключена в Firebird 3.0 по-умолчанию, поскольку не является безопасной.

Вы можете разрешить авторизацию предыдущими версиями клиентов Firebird в процессе инсталяции. Это можно сделать и после установки изменив параметры AuthServer, UserManager и WireCrypt.

Пароль SYSDBA

Изначально в системе существует только один пользователь – администратор сервера SYSDBA (пароль по умолчанию — masterkey). Этот пользователь обладает полными правами на выполнение всех функций по управлению работой сервера и работе с базами данных. В процессе инсталляции Вас попросят изменить пароль данного пользователя в целях безопасности.

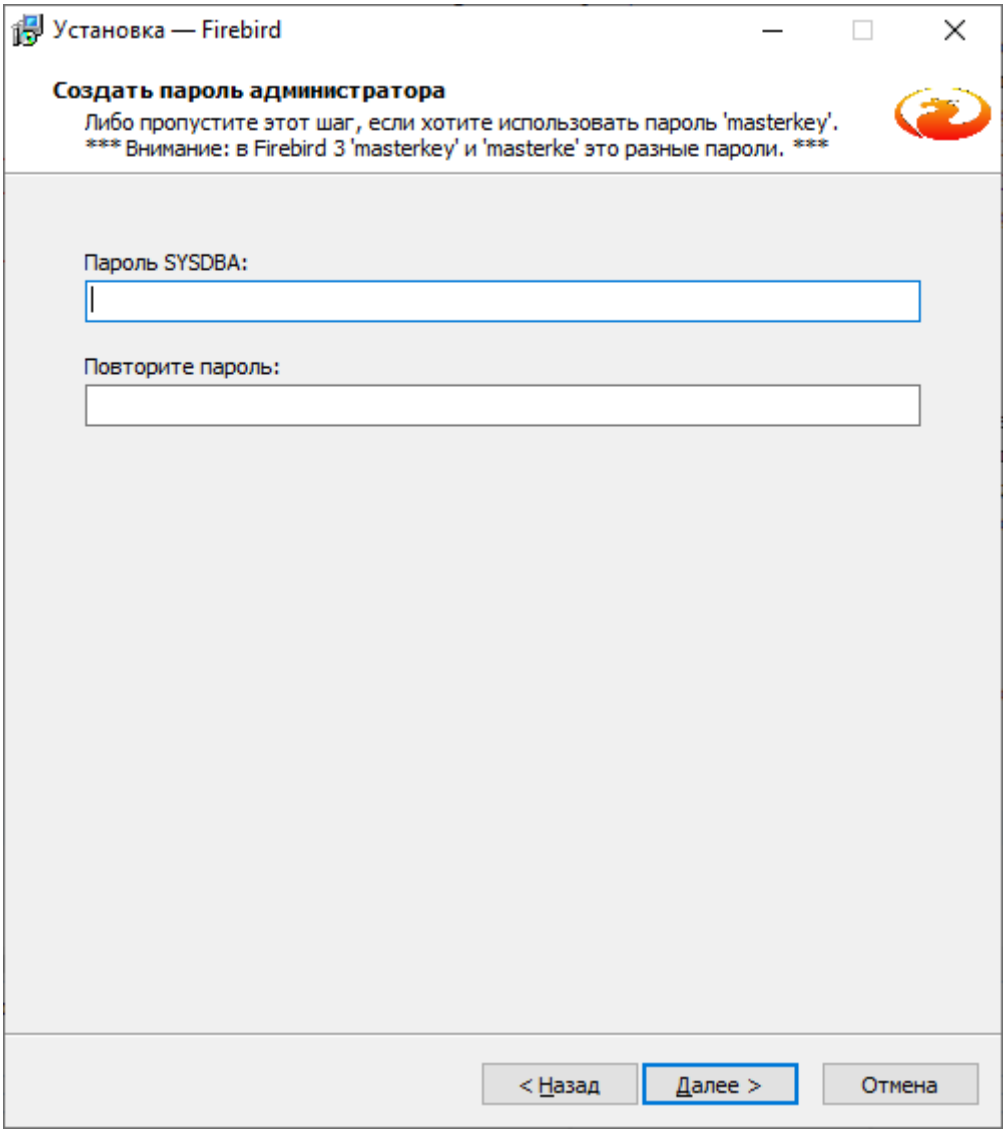

Рисунок 6. Задание пароля SYSDBA

После этого шага будет отображено окно, в котором отображаются опции, выбранные пользователем, если вас всё устраивает можно начать автоматическую установку сервера.

1.2.2. Ручная установка из ZIP архива

Скачайте архив соответствующей разрядности и распакуйте его в директорию, в которой будет размещён сервер Firebird.

Инициализация SYSDBA

Начиная с Firebird 3.0 пользователь SYSDBA не инициализирован по умолчанию (для плагина управления пользователями SRP), поэтому необходимо явно создать пользователя и указать ему пароль. Это можно сделать двумя способами: с использованием консольного инструмента для выполнения интерактивных запросов `isql.exe` и консольного инструмента для управления базой данных безопасности `gsec.exe`.

В зависимости от размещения Firebird эти утилиты могут потребовать запуска с привилегиями администратора.

Инициализация SYSDBA с помощью ISQL

Запустите инструмент для выполнения интерактивных запросов `isql.exe`. Соединитесь с базой данных безопасности в режиме встроенного сервера, указав при этом пользователя SYSDBA без пароля.

Пользователя SYSDBA ещё не существует в базе данных безопасности, но в embedded режиме пользователь и его пароль не проверяется, и Firebird доверяет любому указанному имени пользователя.

Выполите SQL запрос для создания пользователя SYSDBA:

```
CREATE USER SYSDBA PASSWORD '<password>';
```

Пользователь SYSDBA инициализирован, можно выходить из интерактивного режима.

Пример 1. Инициализация SYSDBA через isql

```
c:\Firebird\3.0>isql
Use CONNECT or CREATE DATABASE to specify a database
SQL> connect security.db user SYSDBA;
Database: security.db, User: SYSDBA
SQL> CREATE USER SYSDBA PASSWORD 'm8ku234pp';
SQL> exit;

c:\Firebird\3.0>
```

Инициализация SYSDBA с помощью GSEC

Запустите `gsec.exe` указав пользователя SYSDBA и базу данных *security.db*.

Выполите команду для добавления пользователя SYSDBA:

```
add SYSDBA -pw <password>
```

Пример 2. Инициализация SYSDBA через GSEC

```
c:\Firebird\3.0>gsec -user SYSDBA -database security.db
*** gsec is deprecated, will be removed soon ***

GSEC> add SYSDBA -pw m8ku234pp
GSEC> quit

c:\Firebird\3.0>
```

Инструмент `gsec.exe` является устаревшим, многие возможности доступные через SQL не доступны в нём.

Конфигурация

Режим сервера

По-умолчанию Firebird будет стартовать в режиме SuperServer. Если вы хотите чтобы сервер запускался в другой архитектуре, то необходимо изменить значение параметра `ServerMode` в *firebird.conf*. Разкомментируйте его (удалите решётку) и установите нужный режим: Super, SuperClassic или Classic.

```
ServerMode = Classic
```

Подробнее о режимах севера и параметре `ServerMode` читайте в Выбор архитектуры сервера и ServerMode.

Авторизация с предыдущих версий клиента Firebird

В Firebird 3.0 по умолчанию используется безопасная парольная аутентификация (SRP). Клиенты Firebird 2.5 и более ранние версии использовали традиционную аутентификацию (Legacy_Auth), которая отключена в Firebird 3.0 по-умолчанию, поскольку не является безопасной.

Для поддержки традиционной аутентификации необходимо изменить следующие параметры AuthServer, UserManager и WireCrypt.

Пример 3. Включение авторизации с предыдущих версий клиента Firebird

```
AuthServer = Srp256, Srp, Legacy_Auth
UserManager = Srp, Legacy_UserManager
WireCrypt = Enabled
```

Если вам не нужна поддержка безопасной парольной аутентификации (SRP), удалите из `AuthServer` плагины Srp256 и Srp; из `AuthServer` — Legacy_UserManager, а `WireCrypt` можете изменить на Disabled.

После вышеперечисленных манипуляций у нас будет активно два менеджера пользователей, по умолчанию активный тот что первый в списке UserManager.

Одноименные пользователи в разных менеджерах пользователей — это разные пользователи.

Ранее мы уже создали SYSDBA в менеджере пользователей SRP. В Legacy_UserManager SYSDBA уже существует, причём со стандартным паролем masterkey, который необходимо изменить. Сделаем это с использованием инструмента isql. В операторе `ALTER USER` необходимо

обязательно указать менеджер пользователей Legacy_UserManager.

```
c:\Firebird\3.0>isql
Use CONNECT or CREATE DATABASE to specify a database
SQL> connect security.db user sysdba;
Database: security.db, User: SYSDBA
SQL> ALTER USER SYSDBA SET PASSWORD 'er34gfde' USING PLUGIN Legacy_UserManager;
SQL> exit;

c:\Firebird\3.0>
```

Одновременный запуск нескольких Firebird

Для одновременного запуск нескольких Firebird, необходимо развести их по разным портам tcp (если конечно слушатель запущен в режиме прослушивания TCP/IP). Для этого необходимо изменить в *firebird.conf* параметр `RemoteServicePort`. Например, если у вас уже есть один сервер который слушает порт 3050, то необходимо установить любой другой свободный порт, например 3051. В этом случае в строке подключения необходимо будет указывать новый порт.

```
RemoteServicePort = 3051
```

Если необходимо обеспечить также одновременную работоспособность по локальному протоколу XNET и через именованные каналы WNET, то необходимо так же изменить параметры `IpcName` и `RemotePipeName`. Однако стоит учесть, что эти параметры придётся изменять и на стороне клиента через DPB.

Если вы хотите запускать несколько экземпляров Firebird, то не используйте утилиту `instreg.exe`, которая записывает путь к серверу в реестр Windows.

Запись в реестре

Утилита `instreg.exe` позволяет прописывать в реестр Windows путь к корневому каталогу сервера. Для этого необходимо запустить команду

```
instreg install -z
```

Для удаления записи из реестра необходимо запустит следующую команду

```
instreg remove -z
```

Если вы хотите запускать несколько экземпляров Firebird, то не используйте утилиту `instreg.exe`.

Переключатель -z не является обязательным, он позволяет вывести установленную или удаленную версию Firebird.

Установка и запуск Firebird как службы

Утилита `instsvc.exe` записывает, удаляет или меняет информацию о запуске сервера в базе сервисов операционной системы. Кроме того, она позволяет управлять запуском и остановкой сервиса. Если запустить её без параметров, то будет выведена справка по командам и параметрам.

```
instsvc
```

```
Usage:
  instsvc i[nstall]
                   [ -a[uto]* | -d[emand] ]
                   [ -g[uardian] ]
                   [ -l[ogin] username [password] ]
                   [ -n[ame] instance ]
                   [ -i[nteractive] ]

          sta[rt]  [ -b[oostpriority] ]
                   [ -n[ame] instance ]
          sto[p]   [ -n[ame] instance ]
          q[uery]
          r[emove] [ -n[ame] instance ]


  '*' denotes the default values
  '-z' can be used with any other option, prints version
  'username' refers by default to a local account on this machine.
  Use the format 'domain\username' or 'server\username' if appropriate.
```

Для установки сервиса необходимо ввести команду

```
instsvc install
```

В этом случае Firebird будет установлен в качестве службы с именем “Firebird Server – DefaultInstance”. Эта служба будет запускаться автоматически при старте ОС, под учётной записью LocalSystem, предназначенной для служб.

Если необходимо чтобы было установлено несколько экземпляров Firebird работающих как службы, то необходимо задать им разные имена с помощью переключателя `-n`

```
instsvc install -n fb30
```

Для запуска службы воспользуйтесь командой

```
instsvc start
```

Если служба была установлена с именем отличным от умолчательного, то необходимо воспользоваться переключателем `-n`

```
instsvc start -n fb30
```

Для остановки службы воспользуйтесь командой

```
instsvc stop
```

Если служба была установлена с именем отличным от умолчательного, то необходимо воспользоваться переключателем `-n`

```
instsvc stop -n fb30
```

Для удаления сервиса необходимо ввести команду

```
instsvc remove
```

Если служба была установлена с именем отличным от умолчательного, то необходимо воспользоваться переключателем `-n`

```
instsvc remove -n fb30
```

Для просмотра всех служб Firebird установленных в системе воспользуйтесь командой

```
instsvc query
```

```
Firebird Server - fb30 IS installed.
  Status  : running
  Path    : C:\Firebird\3.0\firebird.exe -s fb30
  Startup : automatic
  Run as  : LocalSystem

Firebird Server - fb40 IS installed.
  Status  : running
  Path    : C:\Firebird\4.0\firebird.exe -s fb40
  Startup : automatic
  Run as  : LocalSystem
```

Использование install_service.bat и uninstall_service.bat

Для упрощения процедуру установки и удаления служб в ZIP архиве в комплекте с Firebird поставляются два BAT файла: [path]fbadmdg-install-service.bat_ и *uninstall_service.bat*.

В этом случае процедура установки Firebird в качестве сервиса выглядит следующим образом

```
install_service.bat
```

Если необходимо задать службе имя отличное от умолчательного, то указываем это имя в качестве аргумента

```
install_service.bat fb30
```

В этом случае процедура удаления службы Firebird выглядит следующим образом

```
uninstall_service.bat
```

Если служба была установлена с именем отличным от умолчательного, то указываем это имя в качестве аргумента

```
uninstall_service.bat fb30
```

Запуск Firebird как приложения

Для запуска Firebird в качестве приложения достаточно выполнить команду

```
firebird -a
```

Запуск Firebird как приложения удобен на компьютере разработчика. Это особенно полезно во время отладки UDF, UDR и различных плагинов Firebird.

Установка клиента

Если речь идёт об установке только о клиентской части, то обязательно требуется файл *fbclient.dll*. Клиент Firebird 3.0 обязательно требует наличие установленного Microsoft Runtime C++ 2010 соответствующей разрядности. Если данная библиотека не установлено, то можно скопировать дополнительные библиотеки, которые поставляются в ZIP архиве под Windows *msvcp100.dll* и *msvcr100.dll*.

Желательно, чтобы рядом с *fbclient.dll* был расположен файл сообщений *firebird.msg*. Большинство сообщений об ошибках уже содержатся в *fbclient.dll*, однако если вы собираетесь пользоваться консольными утилитами файл *firebird.msg* обязательно должен присутствовать.

Если требуется сжатие трафика при работе по TCP/IP, то потребуется так же библиотека *zlib1.dll*.

Для того чтобы клиентское приложение могло загрузить библиотеку *fbclient.dll* она должна располагаться либо рядом с приложением, либо в одной из директорий в которой производится поиск, например добавленной в *PATH* или системной директории для размещения общедоступных библиотек (*system32* или *SysWOW64*).

Размещение клиентской библиотеки в *PATH* может помешать другим приложениям, которым требуется клиентская библиотека другой версии или другого сервера. Поэтому, если предполагается, что приложение должно работать независимо от других приложений с конкретной версией клиента, то файлы клиента требуется разместить в папке приложения, и не прописывать этот путь в *PATH*.

Если у вас ещё нет (и не планируется) установок сервера Firebird или его клиента, то вы можете зарегистрировать путь к корню Firebird в реестре.

```
instreg install
```

Утилита instclient

Утилита `instclient.exe` позволяет:

- установить клиентскую часть одной командой;
- установить клиентскую часть как *fbclient.dll* и/или *gds32.dll*;
- проверить наличие установленной библиотеки *fbclient.dll* и/или *gds32.dll*;
- удалить ранее установленные *fbclient.dll* и/или *gds32.dll*.

Утилита `instclient.exe` должна быть запущена с правами администратора.

Если запустить `instclient.exe` без параметров, то будет выведена краткая справка:

```
instclient
```

```
Usage:
  instclient i[nstall] [ -f[orce] ] library
             q[uery] library
             r[emove] library

  where library is:  f[bclient] | g[ds32]
```

Для развёртывания клиентской библиотеки Firebird в системном каталоге Windows воспользуйтесь командой

```
instclient install fbclient
```

```
FBCLIENT.DLL has been installed to the System directory.
```

При развёртывании библиотеки проверяется версия и инкрементируется счётчик общих библиотек. Для отмены проверки версии вы можете воспользоваться переключателем `-force`.

Некоторые старые приложения требуют для работы клиентскую библиотеку *gds32.dll* (наследие Interbase). Кроме того, многие приложения требуют для работы не просто *gds32.dll*, но и проверяют версию этой библиотеки. Именно поэтому простое переименовывание *fbclient.dll* в *gds32.dll* для таких программ работать не будет — версия библиотеки окажется ниже 6.0 (т.к. соответствует версии Firebird, у которого нумерация версий идет с 1.0). Утилита `instclient.exe` позволяет не просто установить *gds32.dll*, но и изменить её версию на корректную.

```
instclient install gds32
```

```
GDS32.DLL has been installed to the System directory.
```

Утилита `instclient.exe` позволяет также проверить наличие установленной клиентской библиотеки с помощью команды

```
instclient q fbclient
```

```
Installed FBCLIENT.DLL version : 3.0.4.33054 (shared DLL count 1)
```

```
instclient q gds32
```

```
Installed GDS32.DLL version : 6.3.4.33054 (shared DLL count 2)
```

Для удаления воспользуйтесь командой

```
instclient remove fbclient
```

```
The FBCLIENT.DLL has been removed from the System directory.
```

```
instclient remove gds32
```

```
The GDS32.DLL has been removed from the System directory.
```

Обновление

В данном случае речь об обновлении в рамках пойнт-релизов (третья цифра в версии сервера) или так называемых сервис-паков. Обновление мажорной или минорной версии обычно требует гораздо больше действий, описанных в Release Note, и называется “Миграцией”.

Обычно обновление в рамках пойнт-релизов не требует процедуры резервного копирования и восстановления базы данных средством gbak, но бывают исключения. Такие исключения обязательно описаны в Release Note, поэтому обязательно ознакомьтесь с ним перед обновлением.

Примерно те же действия требуются при обновлении на текущий снапшот. Снапшоты — это ежедневные сборки сервера Firebird, которые включают самые последние изменения текущей версии сервера. Это позволяет иметь самую свежую версию сервера с исправленными ошибками и улучшениями. Однако учтите, что снапшоты не проходят полный цикл тестирования и могут содержать новые ошибки или регрессии. Поэтому не обновляйтесь до снапшота без явной необходимости (исправленные в нём ошибки касаются вас непосредственно).

Алгоритм обновления поверх существующего сервера Firebird следующий:

1. Убедится что все пользователи отключены от всех баз данных обслуживаемых обновляемым сервером Firebird. С помощью команды

```
gfix -shut -force n databasename
```

можно закрыть все подключения и запретить последующие.

2. Остановить сервер
3. Сделать резервную копию корневого каталога сервера
4. Распаковать из ZIP архива с пойнт-релизом или снапшотом файлы в корневую директорию сервера
5. Заменить новые файлы конфигурации (*firebird.conf*, *databases.conf* и другие), а также файл *security3.fdb* на прежние
6. Запустить обновленный сервер
7. Перевести базы данных в онлайн-режим

```
gfix -online databasename
```

Встроенный сервер

Начиная с Firebird 3.0 встроенная (embedded) версия не распространяется отдельно. Всё, что необходимо для развертывания Firebird Embedded, находится в том же самом ZIP архиве, что и обычная версия.

Для функционирования Firebird Embedded необходимы следующие файлы:

- клиентская библиотека (*fbclient.dll* или *libfbclient.so*);
- движок для работы с ODS 12.0 (*engine12.dll* или *libEngine12.so*) из папки *plugins*;
- библиотека интернационализации, которая находится в каталоге *intl*;
- файл сообщений *firebird.msg*;
- для изменения некоторых параметров могут потребоваться *firebird.conf* и/или *databases.conf*;
- в Windows обязательно необходимы файлы ICU библиотеки;
- если в Windows не установлен MS VC++ Runtime 2010, то необходимы также файлы (*msvcp100.dll* и *msvcr100.dll*);
- если вам необходимы консольные инструменты администрирования, то скопируйте также *gbak*, *gfix*, *gstat*, *isql*, *nbackup*.

Конфигурация

По умолчанию Firebird работает в режиме Super сервера, в том числе и в embedded варианте. Это не очень удобно, поскольку Super открывает файл базы данных в монопольном режиме, что обозначает что только один процесс может работать с базой данных. Для изменения режима запуска необходимо отредактировать (возможно раскомментировать) значение параметра `ServerMode` на Classic или SuperClassic в файле

firebird.conf.

Поскольку в embedded варианте сетевой доступ не используется вы можете отредактировать также провайдеры `Providers`, удалив от туда лишние провайдеры Remote и Loopback.

```
Providers = Engine12
```

1.3. Файлы конфигурации

databases.conf

В этом текстовом файле можно сопоставить конкретный путь к БД и псевдоним, чтобы затем в прикладных кодах использовать более короткий и удобный псевдоним для обращения к нужной базе данных. Также здесь указываются индивидуальные настройки для каждой конкретной базы данных. Если под управлением сервера Firebird находится более 1 базы данных, то рекомендуем все настройки уровня базы данных делать в этом файле.

fbtrace.conf

Файл с шаблоном настроек *fbtrace.conf* находится в корневом каталоге и содержит список отслеживаемых событий и указывает размещение логов трассировки для каждого события. Это позволяет достаточно гибко настроить параметры аудита различных событий для любой базы данных, при этом логирование будет осуществляться в отдельные файлы.

firebird.conf

Файл содержит параметры конфигурации сервера.

plugins.conf

Файл используется для настройки различных плагинов. Если в файле не указана конфигурация для плагина, то для него будут действовать настройки по умолчанию.

udr_engine.conf

Файл используется для настройки плагина UDR Engine — движок для подключения внешних процедур, функций и триггеров. Файл располагается в директории *plugins*.

fbintl.conf

Файл конфигурации для библиотеки интернационализации. Файл располагается в директории *intl*.

1.4. Инструменты администрирования и сервисы

В Windows исполняемые файлы имеют расширение *.exe*, в других операционных системах могут использоваться другие расширения или вовсе не использоваться.

fb_lock_print

Эта утилита формирует статистические данные файла блокировок, который поддерживается в Firebird для управления последовательностью изменений базы данных несколькими транзакциями. Она может быть полезным инструментом анализа проблем взаимной блокировки.

fbguard

Исполняемый файл приложения Guardian. Он контролирует процесс сервера. Если сервер упал по какой-либо причине, Guardian автоматически перезапускает его.

В современных версиях Windows не требуется. Дело в том, что у служб можно установить свойство автоматического перезапуска в случае падения. Однако данная утилита может быть полезна, если Firebird запускается в качестве приложения.

fbsvcmgr

Утилита предоставляет интерфейс командной строки для Services API, обеспечивая доступ к любой службе, которая реализуется в СУБД.

fbtracemgr

Утилита для работы в интерактивном режиме с трассировкой.

firebird

Исполняемый файл сервера (слушатель).

gbak

Эта утилита предназначена для резервного копирования и восстановления баз данных. Она также обнаруживает разрушения базы данных, освобождает дисковое пространство, появившееся в результате удалений, разрешает незавершенные транзакции, позволяет разделять базы данных на несколько файлов. Она также используется для создания переносимой копии с целью восстановления вашей базы данных на другой аппаратной платформе.

gfix

Это набор общих вспомогательных утилит для изменения свойств баз данных, устранения небольших повреждений базы данных, выполнения различных задач чистки и т. д. Утилита также предоставляет средство администратора для отключения конкретных баз данных до завершения работы сервера. Она может быть использована вместе с утилитой gbak для восстановления некоторых типов нарушений в базе данных.

gpre

Это препроцессор, который конвертирует исходный код, написанный на некоторых языках и содержащий встроенный псевдокод SQL, в корректный отформатированный вызов функций Firebird API.

gsec

Этот инструмент поддержки списка пользователей и их паролей является интерфейсом

командной строки для базы данных security3.fdb; он управляет записями пользователей на сервере.

gsplit

Это фильтр, который позволяет преодолевать ограничения на максимальный размер файла, существующие в некоторых операционных системах, при создании резервной копии очень большой базы данных. Эта утилита поставляется только для ОС Windows и, к сожалению, кажется она не работает. К счастью, gbak позволяет разбивать файлы резервной копии на несколько частей, так что gsplit не требуется. В системах ОС Unix существует подходящие утилиты из операционной системы, которые могут быть использованы вместо gsplit, если это необходимо.

gstat

Этот инструмент получения статистики. Он собирает и отображает статистические сведения по индексам и данным базы данных.

instclient

Назначение утилиты instclient состоит в том, что она:

- позволяет установить клиентскую часть Firebird одной командой;
- позволяет установить клиентскую часть как fbclient.dll, либо как gds32.dll;
- позволяет проверить наличие установленной библиотеки fbclient или gds32;
- позволяет удалить уже установленный в системе fbclient или gds32.

instreg

Эта утилита прописывает необходимую информацию в реестр Windows, указывая умолчательное расположение файлов сервера.

instsvc

Утилита instsvc записывает, удаляет или меняет информацию о запуске сервера в базе сервисов операционной системы Windows.

isql

Интерактивный инструмент, который позволяет выполнять запросы к базе данных.

nbackup

Утилита позволяет создавать резервные копии и восстанавливать из резервных копий также, как gbak, и дополнительно позволяет создавать инкрементные копии и восстанавливать из них БД.

qli

Это Query Language Interpreter (интерпретатор языка запросов), интерактивный клиентский инструмент запросов. Он может обрабатывать операторы DDL и DML из SQL и GDML. Хотя уже есть isql и другие инструменты графического интерфейса сторонних разработчиков, qli все еще имеет значение по причине его способности осуществлять некоторые операции, до сих пор не реализованные в SQL Firebird. В отличие от isql, qli может одновременно соединяться более чем с одной базой данных и может

симулировать обращение к нескольким базам данных в одном запросе.

1.5. Динамические библиотеки

В Windows динамические библиотеки имеют расширение *.dll*, в Linux — *.so*, в Mac OS — *.dylib*.

fbclient

Клиентская библиотека. Предоставляет интерфейс прикладного программирования (API) с функциями для подключения к сервисам, работы с базами данных и их созданию. Библиотека также выполняет роль y-valve для подключения и сопряжения различных плагинов.

В Windows имеет имя *fbclient.dll*, в Unix — *libfbclient.so*.

ib_util

Библиотека утилиты памяти.

В Windows имеет имя *ib_util.dll*, в Unix — *libib_util.so*.

ICU

ICU — International Components for Unicode — библиотеки поддержки юникода.

В Windows используется библиотека поставляемая в комплекте Firebird, в Unix — используется системная библиотека. В Windows в комплекте с Firebird поставляются следующие файлы: *icudt52l.dat*, *icudt52.dll*, *icuin52.dll*, *icuuc52.dll*.

Microsoft VC++ Runtime

Microsoft VC++ Runtime 2010 требуется только в Windows. Поставляется в комплекте с Firebird на случай, если библиотека соответствующей версии не установлена в системе. В комплекте Firebird под Windows входят файлы: *msvcp100.dll*, *msvcr100.dll*.

zlib1.dll

Библиотека *zlib1.dll* используется для сжатия сетевого трафика в Windows, в UNIX используется системная библиотека.

fbintl

Библиотека для поддержки интернационализации, кодировок и порядка сортировок. Располагается в директории *intl* относительно корневой директории сервера.

В Windows имеет имя *fbintl.dll*, в Unix — *fbintl*.

1.6. Плагины

Модули плагинов размещаются в динамических библиотеках. В Firebird для модулей плагинов выделена папка *plugins*, которая находится в корневой директории сервера.

Engine12

Поставщик (ядро) для работы с ODS 12.

В Windows имеет имя *engine12.dll*, в Unix — *libEngine12.so*, в MacOS — *libEngine12.dylib*.

fbtrace

Плагин трассировки.

В Windows имеет имя *fbtrace.dll*, в Unix — *libfbtrace.so*, в MacOS — *libfbtrace.dylib*.

legacy_auth

Плагин для поддержки традиционной аутентификации (использовалась в Firebird 2.5 и в более ранних версиях).

В Windows имеет имя *legacy_auth.dll*, в Unix — *libLegacy_Auth.so*, в MacOS — *libLegacy_Auth.dylib*.

legacy_usermanager

Менеджер пользователей, который используется в традиционной аутентификации.

В Windows имеет имя *legacy_usermanager.dll*, в Unix — *libLegacy_UserManager.so*, в MacOS — *libLegacy_UserManager.dylib*.

srp

Плагин аутентификации методом SRP — Secure remote Password.

В Windows имеет имя *srp.dll*, в Unix — *libSrp.so*, в MacOS — *libSrp.dylib*.

udr_engine

Внешний движок для подключения UDR написанных на компилируемых языках C, C++, Pascal.

В Windows имеет имя *udr_engine.dll*, в Unix — *libudr_engine.so*, в MacOS — *libudr_engine.dylib*.

1.7. Включаемые файлы

В директории *include* расположены заголовочные C файлы с функциями, константами, структурами и интерфейсами API Firebird.

1.8. Примеры

В директории *examples* расположены примеры работы с Firebird API, написания плагинов, UDF и UDR.

1.9. Документация

В директории *doc* расположены файлы документации.

1.10. Другие файлы

security3.fdb

База данных безопасности. В этой базе хранятся параметры пользователей системы, политики доступа, глобальные роли.

firebird.log

Лог-файл сервера.

firebird.msg

Файл с сообщениями сервера (в основном об ошибках).

Глава 2. Описание параметров конфигурации

Для настройки сервера Firebird используется файл *firebird.conf*. Настройки считываются из файла один раз при старте сервера, если архитектура SuperSever или SuperClassic, и при каждом соединении с базой, если архитектура сервера Classic.

По умолчанию все параметры в файле конфигурации закомментированы. Для обозначения комментариев используется символ “#”.

Текст, следующий после символа “#” , до конца строки является комментарием, например:

```
#комментарий
DefaultDbCachePages = 2048 #комментарий
```

Максимальная длина строки в файле конфигурации сервера равна 80 символов.

Первое слово в строке, начинающейся не с символа комментария, считается названием параметра. Справа от имени параметра, после символа “=”, указывается значение параметра.

2.1. Типы параметров

В файле конфигурации присутствуют параметры трех типов:

- целочисленный (Integer);
- строковый (String);
- логический (Boolean).

2.1.1. Целочисленные параметры

Целочисленные как ясно из названия могут содержать только целые числа. Они могут быть записаны со множителями k, m и g.

Пример 4. Целочисленные параметры

```
MaxUnflushedWrites = 100
DefaultDbCachePages = 8K        # 8 * 1024 = 8192
DatabaseGrowthIncrement = 128M  # 128 * 1024 * 1024
```

Значения параметров, определяющих объем памяти, указываются в байтах. В конце таких значений можно ставить сокращения k, m и g, соответствующие килобайтам, мегабайтам и гигабайтам.

2.1.2. Логические параметры

Логическое значение выражается в виде целочисленных значений, где 0 (ноль) означает `false`, а ненулевое значение означает `true`. Для согласованности мы рекомендуем использовать только 0/1. Также строки 'y', 'yes' и 'true' означают `true`, а 'n', 'no' и 'false' — `false`.

Пример 5. Логические параметры

```
RemoteAccess = true       # true
RemoteFileOpenAbility = 0 # false
```

2.1.3. Строковые параметры

Строковые как ясно из названия представляют собой строки.

Пример 6. Строковые параметры

```
RemoteServiceName = gds_db
RemotePipeName = pipe47
```

2.2. Область действия

Параметры конфигурации могут быть применены к экземпляру сервера (Per-server), отдельной базе данных (Per-database) или отдельному соединению (Per-connection).

Параметры, которые могут быть применены к отдельной базе данных можно указывать в файле *databases.conf*. В этом случае они перекроют значение одноимённого параметра указанного в *firebird.conf*.

Параметры, которые могут быть применены к отдельному соединению, предназначены для использования на стороне клиента. Они устанавливаются с помощью `isc_dpb_config` параметра в DPB (Database parameters buffer) (`isc_spb_config` для служб).

Параметры уровня базы данных могут быть установлены используя DPB при первом соединении с базой данных, если используется встроенный режим сервера.

Параметры, которые могут быть применены к отдельному соединению, могут быть прочитаны из файлов *firebird.conf* и *databases.conf* расположенных в том же каталоге, что и клиентская библиотека fbclient. Эти параметры будут перекрыты установками через DPB.

2.3. Макроподстановки

Существует ряд предопределенных макроподстановок, которые могут быть использованы в

файлах конфигурации, где требуется имя каталога. Они используют следующий синтаксис:

```
$(<name>)
```

Полный список таких макроподстановок представлен ниже:

root

корневой каталог экземпляра сервера;

install

директория в которую установлен Firebird. Изначально `$(root)` и `$(install)` одинаковые. `$(root)` может быть переопределена установкой или изменением переменной окружения FIREBIRD. В таком случае эта подстановка отлична от $(install);

this

каталог, в котором находится текущий файл конфигурации;

dir_conf

каталог, в котором расположены файлы конфигурации *firebird.conf* и *databases.conf*;

dir_secDb

директория, в которой расположена база данных безопасности по умолчанию;

dir_plugins

директория, в которой расположены плагины;

dir_udf

директория, предназначенная для размещения UDF функций по-умолчанию;

dir_sample

каталог с примерами (*examples*);

dir_sampleDb

директория, где лежит пример базы данных (*examples/empbuild*);

dir_intl

директория, в которой расположены библиотеки интернационализации;

dir_msg

каталог, в котором расположен файл с сообщениями сервера *firebird.msg*. Обычно он совпадает с `$(root)`, но может быть переопределён переменной окружения `FIREBIRD_MSG`.

Внутренние макроподстановки не чувствительны к регистру. Большие буквы используются исключительно для повышения читабильности.

Пример 7. Пример использования макроподстановки в файле databases.conf

```
employee = $(dir_sampleDb)/employee.fdb
```

2.4. Включение других файлов

Один файл конфигурации может включать другой с помощью директивы `include`:

```
include some_file.conf
```

Относительный путь представляет собой путь по отношению к текущему файлу конфигурации. Так, в примере выше файл */opt/config/master.conf* ссылается на файл расположенный по пути */opt/config/some_file.conf*.

Директива include поддерживает групповые символы “*” и “?”. Все совпадающие с шаблоном файлы будут подключены, порядок включения не определён.

Пример 8. Использование групповых символов с директивой include

```
include $(dir_plugins)/config/*.conf
```

2.5. Параметры уровня базы данных

Настройка конфигурации на уровне базы данных осуществляется с помощью формальных записей в файле *databases.conf*. Такие параметры помечены как “Per-database”.

2.5.1. Формат конфигурационных записей

Если вы не добавляете каких либо специфичных для базы данных директив конфигурации для псевдонима, то формат будет такой же, как он был в *aliases.conf* (Firebird 2.5 и ранее):

```
emp = c:\Program Files\examples\empbuild\employee.fdb
# или
emp = /opt/firebird/examples/empbuild/employee.fdb
# или
emp = $(dir_sampleDb)/employee.fdb
```

Несколько более сложный формат используется для случаев, когда определенные не глобальные параметры должны быть нацелены на отдельные базы данных. Запись для базы данных определяется объявлением псевдонима, как и ранее. Директивы, относящиеся к базе данных, перечислены ниже в фигурных скобках.

```
# Directives for MYBIGDB
MYBIGDB = opt/databases/mybigdb.fdb
{
    LockMemSize = 32M           # We know that MYBIGDB needs a lot of locks
    LockHashSlots = 19927       # and a hash table large enough for them
}
```

2.5.2. Доступные параметры

Следующие параметры можно скопировать/вставить в файл *database.conf* и использовать в качестве переопределений для конкретных баз данных.

Таблица 1. Параметры доступные в databases.conf

Связанные с ядром сервера		
DatabaseGrowthIncrement	DeadlockTimeout	DefaultDbCachePages
EventMemSize	FileSystemCacheThreshold	ExternalFileAccess
GCPolicy	LockAcquireSpins	LockHashSlots
LockMemSize	MaxUnflushedWrites	MaxUnflushedWriteTime
SecurityDatabase	UserManager	
WireCompression	WireCrypt	WireCryptPlugin
Связанные с клиентом	Некоторые параметры могут быть настроены в клиентском соединении через DPB/SPB, в качестве альтернативы их настройке в *database.conf*. Подробнее смотри в Область действия.	
AuthClient	Providers	
Следующие параметры могут быть настроены на стороне клиента только через DPB/SPB.		
ConnectionTimeout	DummyPacketInterval	IpcName
RemoteAuxPort	RemotePipeName	RemoteServiceName
RemoteServicePort	TCPNoNagle	

2.6. Общие настройки

2.6.1. DatabaseAccess

Область действия:

Per-server.

Тип параметра:

String (специальный формат).

Синтаксис

```
DatabaseAccess = None | Full | { Restrict <dir_1>[;<dir_2>[...;<dir_N>] }
```

Параметр `DatabaseAccess` позволяет обеспечить управление безопасностью при доступе к файлам базы данных. Доступ к файлам базы данных на сервере может быть полным (`Full`), ограниченным (`Restrict`) или запрещённым (`None`).

Параметр `DatabaseAccess` имеет строковый тип; по умолчанию значение параметра равно `Full` — полный доступ. Для ограничения доступа используется параметр `Restrict`. В этом случае после слова `Restrict` указываются директории, в которых могут быть сохранены файлы базы данных.

При указании списка каталогов могут быть использованы как абсолютные, так и относительные пути. Относительные пути считаются от корневого каталога сервера Firebird. В качестве разделителя каталогов используется символ “;”.

Если параметр `DatabaseAccess` установлен в значение `None`, то позволено соединяться только с базами данных, перечисленными в файле *databases.conf*.

Пример 9. Значения параметра `DatabaseAccess`

```
DatabaseAccess = None
DatabaseAccess = Restrict C:\DataBase
DatabaseAccess = Restrict C:\DataBase;D:\Mirror
DatabaseAccess = Restrict /db
DatabaseAccess = Restrict /db;/mnt/mirrordb
DatabaseAccess = Full
```

Неконтролируемый доступ к файлам баз данных может поставить под угрозу безопасность вашей системы. Поэтому настоятельно рекомендуем ограничивать директории для размещения баз данных.

2.6.2. RemoteAccess

Область действия:

Per-database.

Тип параметра:

Boolean.

Параметр предоставляет или отменяет удалённый доступ к базам данных.

```
RemoteAccess = true
```

По-умолчанию `RemoteAccess` включен для всех баз данных, за исключением базы данных

безопасности. Если вы намереваетесь использовать больше одной специализированной базы безопасности, то рекомендуем отключить удалённый доступ к ним в файле *databases.conf*.

Для повышения безопасности можно отключить `RemoteAccess` глобально в файле *firebird.conf* и включить его для отдельных баз данных в *databases.conf*.

Параметр имеет тип Boolean и может принимать значения `true`/`false`, `1`/`0` или `yes`/`no`.

2.6.3. ExternalFileAccess

Область действия:

Per-database.

Тип параметра:

String (специальный формат).

Синтаксис

```
ExternalFileAccess = None | Full | { Restrict <dir_1>[;<dir_2>[...;<dir_N>] }
```

Параметр `ExternalFileAccess` позволяет управлять правами размещения файлов внешних таблиц. Разрешение на доступ к внешним файлам может быть полным (`Full`), ограниченным (`Restrict`) или запрещённым (`None`).

Параметр `ExternalFileAccess` имеет строковый тип; значение по умолчанию равно None — запрет на создание внешних таблиц. Для того, чтобы разрешить создание и доступ к внешним файлам, следует выставить значение параметра равным Full. Для ограничения доступа используется значение Restrict. В этом случае после слова Restrict указываются директории, в которых могут быть сохранены файлы внешних таблиц. При указании каталогов могут быть использованы как абсолютные, так и относительные пути. Относительные пути берутся от корневого каталога Firebird. В качестве разделителя директорий используется символ “;”.

Пример 10. Примеры задания значения параметра `ExternalFileAccess`

```
ExternalFileAccess = None
ExternalFileAccess = Restrict C:\DataBase
ExternalFileAccess = Restrict C:\DataBase;D:\Mirror
ExternalFileAccess = Restrict /db;/mnt/mirrordb
ExternalFileAccess = Full
```

Неконтролируемая возможность использования внешних таблиц может поставить под угрозу безопасность вашего сервера. Поэтому настоятельно рекомендуется использовать этот параметр для ограничения директорий размещения внешних таблиц.

2.6.4. UdfAccess

Область действия:

Per-server.

Тип параметра:

String (специальный формат).

Синтаксис

```
UdfAccess = None | Full | { Restrict <dir_1>[;<dir_2>[...;<dir_N>] }
```

Параметр `UdfAccess` предназначен для определения директорий, в которых могут быть сохранены библиотеки UDF. Разрешение на доступ к библиотекам внешних функций может быть полным (Full), ограниченным (Restrict) или запрещённым (None).

Параметр `UdfAccess` имеет строковый тип; значение по умолчанию равно Restrict UDF — udf-библиотеки ищутся только в корневом каталоге сервера в папке *udf*. Для того, чтобы запретить использование udf, нужно выставить значение параметра равным None.

При указании каталогов могут быть использованы как абсолютные, так и относительные пути. Относительные пути берутся от корневого каталога сервера Firebird. В качестве разделителя директорий используется символ “;”.

```
UdfAccess = Restrict UDF
```

Неконтролируемая возможность использования внешних функций может быть использована для того, чтобы поставить под угрозу безопасность как баз данных, так и всей системы. Поэтому настоятельно рекомендуется использовать данный параметр для ограничения директорий размещения udf-библиотек.

Этот параметр никак не влияет на правила размещения нового типа внешних библиотек UDR.

2.6.5. TempDirectories

Область действия:

Per-server.

Тип параметра:

String (специальный формат).

С помощью параметра `TempDirectories` можно задать временный каталог сервера Firebird. Временный каталог необходим для выгрузки данных во время сортировки (и в некоторых других случаях), когда исчерпывается выделенная оперативная память.

Параметр `TempDirectories` имеет строковый тип; значение по умолчанию равно пустой строке. Если параметр `TempDirectories` не активен, то путь к временному каталогу определяется исходя из значения переменных окружения `FIREBIRD_TMP`, `TEMP`, `TMP`. Если никакая из вышеперечисленных переменных не задана, то значением для POSIX будет каталог */tmp*, а для Windows — *C:\TEMP*.

В качестве значения параметра может быть задан путь к одному или нескольким каталогам. В этом случае выгрузка временных данных при сортировке будет осуществляться в указанные каталоги. Для папок допускаются как абсолютные, так и относительные пути. Относительные пути берутся от корневого каталога инсталляции сервера Firebird. Если требуется определить несколько временных каталогов, то в качестве разделителя используется символ “;”.

Если указана одна или несколько директорий, то выгрузка временных данных при сортировке будет осуществляться в указанные каталоги по очереди (если в текущей временной директории не осталось места, то временные файлы будут сохраняться в следующую по списку).

Пример 11. Примеры задания значения параметра TempDirectories

```
TempDirectories = c:\temp
TempDirectories = c:\temp;d:\temp
```

Количество ресурсов, выделяемых под `TempDirectories`, следует увеличивать, если планируется восстановление базы (restore), имеющей большие индексы. Иначе могут возникнуть проблемы с нехваткой места для *fb_sort** файлов, создаваемых СУБД в процессе построения индексов (СУБД не будет запрашивать у операционной системы новое место в этом случае).

Замечания

- Данные GTT (глобальных временных таблиц) хранятся в файлах с маской *fb_table**, создаваемых в каталогах, заданных переменными `FIREBIRD_TMP` (затем — `TMP`; при отсутствии их обоих GTT-данные будут сохраняться — для POSIX — в каталоге */tmp*). СУБД при определении места, где надо хранить GTT-данные, не учитывает значение параметра `TempDirectories`. Поэтому правильным методом является назначение переменной `FIREBIRD_TMP` того же каталога, что указан в `TempDirectories`.
- В POSIX временные файлы, создаваемые СУБД (*fb_table**, *fb_sort**, *fb_recbuf**, *fb_blob**, *fb_merge**), сразу маркируются как “deleted” и не видны при обычном просмотре каталога. Для получения их списка воспользуйтесь, например, пакетом `lsof` с указанием в качестве аргумента PID процесса Firebird (если он работает как SuperServer или SuperClassic).
- В следующем примере процесс firebird имеет PID = 9447, и для получения открытых им временных файлов следует ввести команду:

```
#lsof -p 9447 | grep
"/fb_table\|/fb_sort\|/fb_recbuf\|/fb_blob\|/fb_merge"
```

Пример вывода:

```
firebird 9447 firebird  296u REG  253,0    147456  2490547
/tmp/fb_table_5aNN2s
  (deleted)
firebird 9447 firebird 1503u REG  253,0    147456  2490573
/tmp/fb_table_0Jwjo3
  (deleted)
firebird 9447 firebird  623u REG   0,16  75497472 25189649
/tmp/fb_sort_kv21Av
  (deleted)
firebird 9447 firebird  296u REG  253,0    147456  2490547
/tmp/fb_table_5aNN2s
  (deleted)
```

- В Windows временные файлы видны, но их равен нулю до тех пор, пока остается свободная память, заданные параметром `TempCacheLimit`.

2.6.6. AuditTraceConfigFile

Область действия:

Per-server.

Тип параметра:

String.

Параметр `AuditTraceConfigFile` в файле конфигурации *firebird.conf* задает имя и расположение файла с настройками системного аудита. Этот параметр имеет строковый тип и по умолчанию имеет пустое значение. Пустое значение параметра означает, что системный аудит выключен.

Указанный файл конфигурации должен иметь ту же структуру, что и *fbtrace.conf*.

Пример 12. Примеры задания значения параметра `AuditTraceConfigFile`

```
AuditTraceConfigFile = fbtrace.conf
```

2.6.7. MaxUserTraceLogSize

Область действия:

Per-server.

Тип параметра:

Integer.

Задает максимальный суммарный размер (в мегабайтах) временных файлов, создаваемых сессией пользовательской трассировки. После прочтения временного файла приложением он автоматически удаляется. Параметр имеет целочисленный тип. Единица измерения — мегабайты. По умолчанию максимальный размер файла вывода ограничен 10 МБ. Если значения ограничения `MaxUserTraceLogSize` достигнуто, то сервер автоматически приостанавливает сессию слежения.

Пример 13. Примеры задания значения параметра `MaxUserTraceLogSize`

```
MaxUserTraceLogSize = 10
```

Количество пользовательских сессий трассировки, которые можно запускать на одном сервере, ничем не ограничено. Указанный параметр лимитирует размер файлов для каждой отдельной сессии трассировки, а не для всех сеансов. Поэтому общий размер временных файлов может превышать 10 Мб.

2.6.8. DefaultDbCachePages

Область действия:

Per-database.

Тип параметра:

Integer.

Параметр `DefaultDbCachePages` используется для настройки количества страниц одной базы данных, находящихся в кеш-памяти одновременно. SuperServer использует единый страничный кеш (2048 страниц по умолчанию) для всех подключений. Classic и SuperClassic создает отдельный страничный кеш (по умолчанию 256 страниц) для каждого соединения.

Параметр имеет целочисленный тип. Допустимо использовать множители (k, m, g). Максимальное значение 2147483647 страниц.

Этот параметр может быть установлен для каждой базы данных индивидуально в файле *databases.conf*. Это особенно удобно если на вашем сервере находятся несколько разных баз данных под управлением одного экземпляра Firebird.

Размер страничного кеша может быть установлен на странице заголовков файла базы данных с помощью утилиты gfix.

```
gfix -b[uffers] <page-count> <database>
```

В этом случае размер страничного кеша устанавливается для каждой базы одновременно. На наш взгляд предпочтительно делать это в файле *databases.conf*, где вы можете установить и другие параметры индивидуальные для вашей базы данных.

Обратите внимание, размер страничного кеша установленный на уровне заголовка базы данных перекрывает значение установленное в *firebird.conf* и *databases.conf*.

Чтобы проверить установлен ли размер страничного кеша на уровне заголовка базы данных вы можете воспользоваться утилитой gstat

```
d:\fb\fb30>gstat -u sysdba -p masterkey -h testdb
```

```
Database "d:\fb\fb30\data\testdb.fdb"
Gstat execution time Wed Jul 17 12:59:26 2019

Database header page information:
        Flags                   0
        Generation              2180
        System Change Number    0
        Page size               16384
        ODS version             12.0
        Oldest transaction      2037
        Oldest active           2038
        Oldest snapshot         2038
        Next transaction        2038
        Sequence number         0
        Next attachment ID      391
        Implementation          HW=AMD/Intel/x64 little-endian
OS=Windows CC=MSV
C
        Shadow count            0
        Page buffers            0
        Next header page        0
        Database dialect        1
        Creation date           Jun 28, 2016 18:04:35
        Attributes              force write

    Variable header data:
        Sweep interval:         20000
        *END*
Gstat completion time Wed Jul 17 12:59:26 2019
```

Page buffers — размер кэша базы данных. Если это значение равно 0, то будет использоваться значение параметра `DefaultDbCachePages` в *firebird.conf* или *databases.conf*.

При изменении данного параметра стоит учитывать архитектуру сервера, максимальное количество соединений с базой данных (Classic, SuperClassic), количество доступной оперативной памяти, а также некоторые другие настройки сервера.

В архитектуре Classic количество страниц может управляться отдельным подключением (то есть оно само может сказать, сколько ему сейчас нужно страниц). Размер страничного кеша может быть увеличен только после рестарта Firebird (для архитектуры SuperServer) или переподключения (для Classic).

В архитектуре SuperServer (предоставляющей для всех подключений общий страничный кеш) рекомендуется назначать страничный кеш как можно большим, но также есть рекомендация не делать его размер более, чем половина от физического размера ОЗУ. Для очень больших БД достаточно выделить такой размер кеша, чтобы в него помещалась активная часть БД. Достаточность кеша обычно проверяют соотношением read/fetch. Хорошим соотношением для FB является `read/fetch < 0.005`.

Пример 14. Установка параметра `DefaultDbCachePages` *в firebird.conf*

```
DefaultDbCachePages = 2048
```

Пример 15. Установка параметра DefaultDbCachePages в databases.conf для базы данныхtestdb

```
testdb = d:\fb\fb30\data\testdb.fdb
{
    DefaultDbCachePages = 8K
}
```

Кроме того, следует обращать внимание на значение параметра `FileSystemCacheThreshold`. Если размер страничного кеша превышает значение параметра `FileSystemCacheThreshold`, то системный файловый кеш будет отключен.

Смотри также:

FileSystemCacheThreshold

2.6.9. DatabaseGrowthIncrement

Область действия:

Per-database.

Тип параметра:

Integer.

Параметр позволяет указать объем дискового пространства, которое может быть предварительно выделено под базу данных. Дисковое пространство резервируется в системе, что позволяет в дальнейшем снизить физическую фрагментацию файла(-ов) базы данных и дает возможность продолжить работу в условиях недостатка места на диске. Если режим резервирования включен, то сервер резервирует 1/16 часть от уже используемого дискового пространства для одного соединения, но не меньше 128 KB и не больше, чем значение, заданное параметром `DatabaseGrowthIncrement` (по умолчанию 128 MB).

Для отключения резервирования дискового пространства необходимо выставить значение `DatabaseGrowthIncrement` равным 0.

Пример 16. Установка параметра `DatabaseGrowthIncrement`

```
DatabaseGrowthIncrement = 128M
```

Пространство под теневые копии баз данных не резервируется.

2.6.10. FileSystemCacheThreshold

Область действия:

Per-database.

Тип параметра:

Integer.

Параметр `FileSystemCacheThreshold` устанавливает порог использования системного файлового кэша сервером Firebird. Системный файловый кэш используется, если размер страничного кэша (установленного явно в заголовке базы данных или через параметр конфигурации `DefaultDbCachePages`) меньше чем значение `FileSystemCacheThreshold`.

Параметр имеет целочисленный тип. Единица измерения – страница базы данных. По умолчанию параметр имеет значение — 65536 страниц. Максимально допустимое значение параметра — 2147483647. Минимальное значение параметра — 0. Если значение параметра `FileSystemCacheThreshold` равно 0, то сервер не будет использовать системный файловый кэш.

Пример 17. Установка параметра `FileSystemCacheThreshold`

```
FileSystemCacheThreshold = 128K
```

Системный файловый кэш имеет смысл отключать только при достаточно большом размере страничного кэша базы данных. Это позволяет избегать двойного кэширования и снизить потребление оперативной памяти. Это может быть полезным для баз, работающих в режиме read-only на архитектуре SuperServer. Для баз данных, работающих в режиме OLTP-нагрузки, отказ от кеширования файловой системой почти всегда приводит к крайне сильной просадке производительности. Одна из причин — отсутствие в СУБД Firebird возможности упреждающего чтения и отложенной записи, которые есть в операционной системе. Для таких баз всегда должно соблюдаться правило: `DefaultDbCachePages` < `FileSystemCacheThreshold`.

Смотри также:

DefaultDbCachePages

2.6.11. FileSystemCacheSize

Область действия:

Per-server.

Тип параметра:

Integer, измеряется в процентах от объёма доступной оперативной памяти.

Параметр `FileSystemCacheSize` устанавливает максимальный размер оперативной памяти, используемый системным файловым кешем 64-битными Windows XP или Windows Server 2003 с Service Pack 1 или выше. Этот параметр не оказывает никакого эффекта в Unix-подобных операционных системах.

Параметр содержит целое число, представляющее собой количество (в процентах) оперативной памяти, которое может быть использовано под файловый кеш. Значение может быть от 10 до 95%.

Значение считывается при старте сервера и не может быть изменено. Требуется перезагрузка операционной системы для вступления изменений в силу.

Если задать значение 0, операционная система сама будет определять размер файлового кеша (обычно это 30%). Это и есть значение по умолчанию.

Пример 18. Установка параметра `FileSystemCacheSize`

```
FileSystemCacheSize = 25
```

Windows требует обладания привилегией SeIncreaseQuotaPrivilege для управления настройками файлового кеша. Эта привилегия доступна по умолчанию администраторам и службам, а также выдается учетной записи Firebird при установке из дистрибутива Windows Installer.

Если Firebird запущен как приложение или в режиме Embedded или установлен не из официального дистрибутива, учетная запись может не иметь данной привилегии. Процесс не выдаст ошибку при запуске, а просто запишет соответствующее сообщение в файл *firebird.log* и будет работать с настройками операционной системы.

2.6.12. RemoteFileOpenAbility

Область действия:

Per-server.

Тип параметра:

Boolean.

Параметр RemoteFileOpenAbility отключает защиту от открытия баз данных на смонтированных томах NFS в Linux/Unix и SMB/CIFS в Windows.

Эта опция удаляет важную функцию безопасности Firebird и может привести к неисправимому повреждению базы данных. Не используйте эту опцию, если вы не понимаете риски и не готовы принять потерю содержимого вашей базы данных.

Параметр имеет логический тип. По умолчанию его значение равно 0. В этом случае Firebird может открыть базу данных, только если она хранится на диске, физически подключенном к локальному компьютеру, на котором установлена эта копия Firebird. Запросы на подключение к базам данных, хранящимся на смонтированных дисках NFS, перенаправляются на сервер Firebird, работающий на компьютере, который “владеет” диском.

Пример 19. Установка параметра RemoteFileOpenAbility

```
RemoteFileOpenAbility = 0
```

Это ограничение не позволяет двум разным копиям Firebird открывать одну и ту же базу данных без координации их действий. Несогласованный доступ нескольких копий Firebird приведет к повреждению базы данных. В локальной системе блокировка файлов на уровне системы предотвращает несогласованный доступ к файлу базы данных.

NFS не обеспечивает надежный способ обнаружения доступа нескольких пользователей к файлу на смонтированном диске NFS. Если вторая копия Firebird подключится к базе данных на смонтированном диске NFS, она повредит базу данных. В некоторых случаях запуск сервера Firebird на компьютере, который владеет томами, смонтированными в NFS, неудобен или невозможен. Приложения, которые используют embedded вариант Firebird и никогда не разделяющие доступ к базе данных, могут использовать эту опцию, чтобы разрешить прямой доступ к базам данных на смонтированных томах NFS.

Ситуация для SMB / CIFS очень похожа на NFS, поскольку не все конфигурации предоставляют механизмы блокировки файлов, необходимые для безопасной работы. Использование механизма SuperServer с базой данных на файловом сервере NT может считаться относительно безопасным, поскольку блокировка файлов защищает базу данных от использования несколькими механизмами. Сетевой стек по-прежнему может изменять порядок записи, поэтому вы можете получить поврежденную базу данных в случае сетевых ошибок или отключения питания.

Относительно полезный и безопасный случай — это работа с общей базой данных, помеченной только для чтения.

2.6.13. TempBlockSize

Область действия:

Per-server.

Тип параметра:

Integer.

Параметр `TempBlockSize` используется для управления временным пространством. Временное хранилище используется большими сортировками, или для промежуточного хранения набора данных. Параметр `TempBlockSize` определяет размер блока, выделяемого для временного хранилища. Это значение отражает гранулярность выделения пространства.

Параметр имеет целочисленный тип. Единица измерения — байты. По умолчанию параметр имеет значение 1 Мбайт. Максимально допустимое значение 2 Гбайт. Минимальное значение параметра — 0.

Пример 20. Установка параметра `TempBlockSize`

```
TempBlockSize = 2M
```

Оптимальное значение этого параметра (хотя и трудно определяемое) — средний размер памяти, отводимой на операцию сортировки.

Предельный размер всей области оперативной памяти, выделяемой для временных файлов, задается параметром `TempCacheLimit`. Блоки, которые ранее выделялись, хранится в нём как линейный список, поиск в котором выполняется методом половинного деления (бисекции). Если `TempCacheLimit` задан слишком большого размера, например, 16 Гб, то даже при его заполненности на 50% (т.е. 8Гб) для поиска блока размером 1 Мб потребуется 13 шагов. В этом случае лучше увеличить размер `TempBlockSize` до 3-4M, что приведет к уменьшению числа шагов бисекции до 6-8. В то же время, рекомендуется ставить `TempBlockSize` в значение, НЕ превышающее 5% от `TempCacheLimit`, а также оно должно быть НЕ меньше 1 Мб.

Смотри также:

TempCacheLimit

2.6.14. TempCacheLimit

Область действия:

Per-server.

Тип параметра:

Int64.

Параметр `TempCacheLimit` определяет максимальный объём оперативной памяти, который используется для кэширования временного пространства.

Параметр имеет целочисленный тип. Единица измерения — байты. Минимальное значение равно 0, максимальное — (2^{64} - 1) байт. Значение по умолчанию для классического сервера

равно 8 Мбайт, для архитектур SuperClassic и SuperServer — 64 Мбайт.

Для классического сервера это ограничение распространяется для каждого соединения с базой данных, для SuperServer и SuperClassic — для каждого запущенного экземпляра сервера Firebird.

Пример 21. Установка параметра `TempCacheLimit`

```
TempCacheLimit = 256M
```

- Firebird 3.0 и выше позволяет указывать значения этого параметра больше, чем 4 Гб.
- Когда какому-то соединению память перестает быть нужной, она возвращается операционной системе. Другими словами, задание `TempCacheLimit = 4096M` не означает, что 4 Гб памяти будут сразу и безвозвратно зарезервированы процессом Firebird.
- Увеличение `TempCacheLimit` даёт лучший эффект чем назначение для временных файлов каталога на tmpfs (см. параметр `TempDirectories`), так как при этом не возникает дополнительных расходов на обмен данными с ядром операционной системы.
- Значение `TempCacheLimit` гарантирует, что данные этого размера будут находиться в оперативной памяти. Когда память, отведенная для временного хранения данных, будет исчерпана, очередной запрос от Firebird на получение блока размером `TempBlockSize` байт приведет к тому, что новая память может быть уже выделена на диске в swap-области.
- В Firebird 3.0 в области `TempCacheLimit` размещаются данные при выполнении hash-соединений.

Будьте осторожны с назначением этого параметра при работе СУБД в режиме Classic Server. Общий лимит памяти, которую Firebird будет запрашивать у операционной системы для временных файлов, определяется при этом как результат умножения `TempCacheLimit` на количество подключений.

Смотри также:

TempBlockSize

2.6.15. AuthServer и AuthClient

Область действия:

Per-database.

Тип параметра:

String.

Параметр `AuthServer` — набор методов аутентификации, разрешенных на сервере (определяется в файле конфигурации сервера).

Параметр `AuthClient` — набор методов аутентификации, поддерживаемых клиентом (определяется в файле конфигурации на клиенте).

Включенные методы перечислены в виде строковых символов, разделенных запятыми, точками с запятой или пробелами. Если проверить подлинность с помощью первого метода не удалось, то сервер переходит к следующему и т.д. Если ни один метод не подтвердил подлинность, то пользователь получает сообщение об ошибке.

Firebird поддерживает следующие методы аутентификации:

- Безопасная парольная аутентификация (Srp). Протокол SRP (Secure Remote Password) позволяет пользователю, не передавая своего пароля, подтвердить серверу тот факт, что он знает свой пароль;
- Традиционная (Legacy_Auth) аутентификация, использовалась в качестве основного метода аутентификации в Firebird версии 2.5 и ниже;
- Доверительная (Win_Sspi) аутентификация для ОС Windows.

По умолчанию на стороне сервера используется метод Secure remote passwords (Srp256), представленный соответствующей ОС плагином (*libSrp.so* | *Srp.dll* | *Srp.dylib*).

```
AuthServer = Srp256
```

Первоначально Firebird 3.0 использовал по умолчанию плагин Srp, который использовал SHA-1 для генерации ключа шифрования, начиная с Firebird 3.0.4 введён усовершенствованный метод аутентификации Srp256, который использует SHA-256.

Список AuthServer по умолчанию содержит только один элемент (Srp). Это значит, что к такому серверу возможно подключение только тех приложений, которые используют клиента версии 3 и старше. Если ваш сервер должен принимать подключения от клиентов версий до 2.5 включительно, необходимо добавить в этот список элемент Legacy_Auth.

Традиционная (Legacy_Auth) не умеет работать с новым менеджером пользователей Srp. Поэтому если вы хотите соединяться с Firebird 3.0 клиентами fbclient версией 2.5 и ниже, то необходимо также включить в список плагинов управления пользователями (параметр UserManager) плагин Legacy_UserManager.

Если вы хотите использовать плагины аутентификации, которые не предоставляют ключа шифрования (Win_Sspi, Legacy_Auth), то следует отключить обязательное (Required) шифрование каналов передачи данных (параметр `WireCrypt`), кроме случаев, когда вы

работаете с протоколом XNET, который никогда не использует шифрование.

На клиентской стороне по умолчанию используется следующий список методов аутентификации:

```
AuthClient = Srp256, Srp, Win_Sspi, Legacy_Auth  # для Windows клиентов
AuthClient = Srp256, Srp, Legacy_Auth            # для не Windows клиентов
```

Под клиентом следует понимать как обычное подключение от пользователя, так и запрос от сервера-1, к которому ранее уже было подключение, к серверу-2, через механизм external datasource. В этом случае клиентом выступает сервер-1.

Плагин Legacy_Auth присутствует в списке AuthClient по умолчанию для того, чтобы клиенты версии 3.0 и выше могли подключаться к серверам более старых версий.

Чтобы отключить какой-нибудь из методов, раскомментируйте строку и удалите нежелательный метод из списка.

Поскольку плагины аутентификации пробуются в порядке их перечисления, то для наилучшей производительности переместите наиболее часто используемый метод аутентификации на первое место.

Оба параметра могут быть использованы в *databases.conf*. Они могут использоваться как в DPB, так и в SPB для конкретных настроек соединения.

Смотри также:

UserManager, WireCrypt.

2.6.16. UserManager

Область действия:

Per-database.

Тип параметра:

String.

Устанавливает плагин, который будет работать в базе данных безопасности. Это может быть список с пробелами, запятыми или точками с запятой в качестве разделителей. По-умолчанию используется первый подключаемый плагин из списка. Firebird поддерживает следующие плагины управления пользователями:

- Srp;
- Legacy_UserManager.

Для поддержки старой базы данных безопасности и управления пользователями в ней, следует установить значение параметра Legacy_UserManager.

Установка параметра UserManager

```
UserManager = Srp
```

В SQL операторах управления пользователями можно явно указать какой плагин будет использоваться с помощью ключевого слова USING.

Одноименные пользователи, созданные с помощью разных плагинов управления пользователями — это разные пользователи.

Параметр `UserManager` можно использовать в *database.conf* для переопределения в конкретной базе данных.

2.6.17. TracePlugin

Область действия:

Per-server.

Тип параметра:

String.

Задает плагин, используемый функцией трассировки Firebird для отправки данных трассировки в приложение клиента или данных аудита в лог файл.

Установка параметра TracePlugin

```
TracePlugin = fbtrace
```

2.6.18. WireCryptPlugin

Область действия:

Per-connection.

Тип параметра:

String.

Плагин поточного шифра используется для шифрования и дешифрования данных, передаваемых по сети.

По-умолчанию устанавливается значение параметра Arc4, что означает использование плагина потокового шифра Alleged RC4. Сконфигурированный плагин, который требует ключ, сгенерированный настроенным подключаемым модулем аутентификации, может быть переопределен в API для конкретного соединения через DPB или SPB.

Установка параметра WireCryptPlugin

```
WireCryptPlugin = Arc4
```

Смотри также:

WireCrypt.

2.6.19. KeyHolderPlugin

Область действия:

Per-server.

Тип параметра:

String.

Этот параметр представляет собой некоторую форму временного хранилища для ключей шифрования базы данных.

Реализованного плагина по-умолчанию нет, но пример для Linux под названием *libCryptKeyHolder_example.so* можно найти в папке */plugins/*.

2.6.20. AllowEncryptedSecurityDatabase

Область действия:

Per-database.

Тип параметра:

Boolean.

Задает возможность использования зашифрованной базы данных с учетными записями пользователей и паролями.

Передача по зашифрованному сетевому каналу с ключем шифрования, который создается плагином аутентификации (Srp) для передачи далее по сети ключа шифрования БД, может рассматриваться как замкнутый круг. Чтобы послать ключ шифрования БД по сети безопасным способом, передаваемые части ключа уже должны быть зашифрованными. Но это требует наличия ключа шифрования от плагина аутентификации, которому нужно открыть базу безопасности для валидации хеша от пароля, а это, в свою очередь, требует ключа шифрования БД.

К счастью, в большинстве случаев нет необходимости шифровать базу данных безопасности — она достаточно хорошо защищает себя, если вы используете пароли надлежащего качества. Но в некоторых случаях желательно иметь зашифрованную базу безопасности, например, если нужно, чтобы зашифрованная база с пользовательскими данными, также служила базой безопасности. В этом случае особое внимание следует уделить шифрованию ключа перед передачей его на сервер с помощью обратного вызова.

Убедитесь, что ваши ключи хорошо зашифрованы перед тем, как разрешать шифрование базы безопасности. Примите по внимание, что при AllowEncryptedSecurityDatabase=TRUE возможна передача ключа, который не зашифрован протоколом Firebird, даже при не зашифрованной базе безопасности.

Указанное свойство не поддерживается унаследованным (Legacy) плагином проверки пользователей. Если вам нужна безопасность, пожалуйста, никогда не используйте унаследованную аутентификацию.

Убедитесь в том, что понимаете свои действия, прежде чем менять этот параметр на True.

2.6.21. Providers

Область действия:

Per-database и per-connection.

Тип параметра:

String.

Провайдеры — это практически то, что мы подразумеваем под способами, используемыми для соединения клиента с сервером, т.е. через интернет; на том же компьютере через 'localhost'; или через прямое соединение в локальной сети (старый *libfbembed.so* для POSIX сейчас реализован как библиотека *libEngine12.so*; для Windows — *engine12.dll*; для MacOSX — *engine12.dylib*).

В *firebird.conf* доступны по-умолчанию следующие провайдеры:

```
Providers = Remote,Engine12,Loopback
```

В *databases.conf* один или несколько провайдеров могут быть заблокированы, если вставить и раскомментировать строку из *firebird.conf* и удалить нежелательные провайдеры.

Архитектура провайдеров (известная как Open Systems Relational Interface, OSRI) очень эффективна для поддержки сочетания старых и новых форматов базы данных (с разными ODS) на одном сервере, имеющих смешанные подключения к локальным и удаленным базам данных.

Провайдеры реализованные в Firebird, позволяют поддерживать все эти режимы (удаленные соединения, базы данных с разными ODS), а также цепочки провайдеров. Цепочка — это термин для ситуации, когда провайдер использует обратный вызов стандартного API для выполнения операции над базой данных.

Главным элементом архитектуры провайдеров является **y-valve**. На начальном этапе соединения с базой данных или её создания **y-valve** просматривает список известных провайдеров и вызывает их по одному, пока один из них не завершит запрошенную операцию успешно. Для соединения, которое уже установлено, соответствующий провайдер вызывается сразу с почти нулевыми накладными расходами.

Рассмотрим пример работы **y-valve**, когда он выбирает подходящего провайдера при подключении к базе данных. Конфигурация по-умолчанию содержит три провайдера:

- **Remote** (используется для сетевого соединения);
- **Engine12** (ядро для работы с ODS 12);
- **Loopback** (принудительное соединение с локальным сервером, когда задано только имя базы данных без явного указания протокола).

Типичная конфигурация клиента работает таким образом: при подключении к базе данных с именем `RemoteHost:dbname` (синтаксис TCP/IP) или `\\RemoteHost\dbname` (Named pipes) или в URI-подобном синтаксисе с протоколами inet и wnet, провайдер Remote обнаруживает явный синтаксис сетевого протокола и перенаправляет вызов RemoteHost.

Когда `<database name>` не содержит сетевого протокола, а только имя базы данных, провайдер Remote отклоняет его, а провайдер Engine12 выходит на первый план и пытается открыть файл с именованной базой данных. Если это проходит успешно, создается подключение к базе данных.

Но что происходит, если СУБД возвращает ошибку при попытке подключения к базе данных?

- Если файл базы данных, к которому нужно подключиться, не существует, то это не интересно всем.
- Встроенное соединение может не работать, если пользователь, подключившийся к нему, не имеет достаточных прав для открытия файла базы данных. Это было бы обычной ситуацией, если бы база данных не была создана этим пользователем во встроенном режиме или если ему явно не были предоставлены права ОС на встроенный доступ к базам данных.

Это также может происходить, если выбран режим SuperServer и уже существует соединение с базой данных, поскольку SuperServer требует эксклюзивного доступа к файлу базы данных.

- После отказа провайдера Engine12 в получении доступа к базе данных, пытается подключиться провайдер Loopback. Он не очень отличается от Remote, за исключением того, что он пытается получить доступ к именованной базе данных `<dbname>` на сервере с сетевым интерфейсом “внутренней петли” (loopback) в сетевом протоколе TCP/IP.

  В Windows XNET пробуется первым, затем TCP/IP loopback (`localhost:<dbname>`), затем Named Pipes (NetBEUI) loopback. Сервер может быть запущен с отключенным XNET (или любым другим протоколом), поэтому перебираются все варианты. В POSIX поддерживается только TCP/IP протокол, остальные варианты не доступны.

Если вы хотите добиться поведения, аналогичного Firebird 2.5 и ниже, когда при наличии запущенного сервера и указании в строке соединения только алиаса базы данных или пути до неё, устанавливалось соединение по локальному проколу, то необходимо поменять провайдеры Engine12 и Loopback местами.

Для подключения по локальному протоколу рекомендуем использовать URI-подобную строку подключения с явным указанием префикса протокола *xnet://*.

Провайдеры не ограничены тремя вышеперечисленными. Версия 3.0 не поддерживает pre-ODS 12 провайдер. Тем не менее, архитектура провайдеров делает возможным доступ к старым базам данных при переходе на более высокую версию Firebird.

Firebird 4.0 для работы с ODS 13 реализован провайдер Engine13, тем не менее вы можете работать с базой данных ODS 12, через провайдер Engine12 из Firebird 3.0.

2.6.22. DeadlockTimeout

Область действия:

Per-database.

Тип параметра:

Integer.

Параметр `DeadlockTimeout` определяет количество секунд, в течение которых диспетчер блокировок будет ожидать после возникновения конфликта, прежде чем очистить блокировки от мертвых процессов и выполнить дополнительный цикл сканирования взаимоблокировок. Firebird обнаруживает взаимные блокировки мгновенно во всех обычных случаях, поэтому это значение влияет на ситуации, когда что-то идет не так.

Параметр имеет целочисленный тип. Единица измерения — секунды. Значение по умолчанию равно 10 секунд. Минимально допустимое значение параметра равно 0. Максимально допустимое значение равно 2147483647.

Установка параметра DeadlockTimeout

```
DeadlockTimeout = 10
```

Слишком низкое значение может снизить производительность системы.

2.6.23. MaxUnflushedWrites

Область действия:

Per-database.

Тип параметра:

Integer.

Параметр `MaxUnflushedWrites` определяет, как часто страницы из кэш памяти будут выгружаться на жесткий диск (активен только при значении параметра ForcedWrites=Off).

Значение параметра `MaxUnflushedWrites` определяет максимальное количество накопленных страниц не сброшенных на диск, ожидающих сброса при подтверждения транзакции.

Параметр имеет целочисленный тип и измеряется в страницах. Значение по умолчанию равно 100 страниц. Для не Win32 систем значение по умолчанию является -1(Отключено). Максимально допустимое значение равно 2147483647.

Установка параметра MaxUnflushedWrites

```
MaxUnflushedWrites = 100
```

Чем больше значение параметра, тем выше вероятность потери данных при возникновении аппаратного сбоя в системе.

Смотри также:

MaxUnflushedWriteTime

2.6.24. MaxUnflushedWriteTime

Область действия:

Per-database.

Тип параметра:

Integer.

Параметр `MaxUnflushedWriteTime` определяет, как часто страницы из кэш памяти будут выгружаться на жесткий диск (активен только при значении параметра ForcedWrites=Off).

Значение параметра `MaxUnflushedWriteTime` определяет время, по истечении которого страницы данных, ожидающие сброса на диск при подтверждении транзакции, будут выгружены на диск.

Параметр имеет целочисленный тип и измеряется в секундах. Значение по умолчанию равно 5 секунд. Для не Win32 систем значение по умолчанию является -1(Отключено). Максимально допустимое значение равно 2147483647.

Установка параметра MaxUnflushedWriteTime

```
MaxUnflushedWriteTime = 5
```

Чем больше значение параметра, тем выше вероятность потери данных при возникновении аппаратного сбоя в системе.

Смотри также:

MaxUnflushedWrites

2.6.25. BugcheckAbort

Область действия:

Per-server.

Тип параметра:

Boolean.

Опция `BugcheckAbort` определяет, прерывать ли работу сервера при возникновении внутренней ошибки или снимать дамп ядра для последующего анализа. Если опция отключена, то ядро пытается минимизировать ущерб и продолжить работу.

Параметр имеет логический тип. Возможные значения 0 и 1. Значение по умолчанию равно 0, в этом случае механизм снятия дампов отключен. Для отладочных сборок (DEV_BUILD) значение по умолчанию равно 1.

Установка параметра BugcheckAbort

```
BugcheckAbort = 0
```

Обратите внимание, что установка этой опции в 1 заставляет движок производить трассируемые дампы, когда внутри UDF происходит что-то неприятное, например SIGSEGV. В Windows включение этой опции заставляет ядро вызывать средства отладки JIT при возникновении ошибок.

В POSIX при `BugCheckBort = 1` сервер Firebird будет устанавливать soft-лимит для размера файлов равным hard-лимиту, и делать запись в каталог, доступный для записи, как правило это */tmp*.

2.6.26. RelaxedAliasChecking

Область действия:

Per-server.

Тип параметра:

Boolean.

Параметр `RelaxedAliasChecking` позволяет снять ограничение на обязательное использование псевдонимов имен таблиц в запросах. Если опция включена, то Firebird позволяет выполнять подобные запросы:

```
SELECT MY_TABLE.X FROM MY_TABLE A
```

Параметр имеет логический тип. Значение по умолчанию равно 0. Если значение параметра равно 1, то ограничение на обязательное использование псевдонимов таблиц в запросах снимается.

Установка параметра RelaxedAliasChecking

```
RelaxedAliasChecking = 0
```

Не рекомендуется включать этот параметр. Его следует рассматривать как временный обходной путь для портирования неаккуратного устаревшего кода до тех пор, пока не удастся пересмотреть такой код.

Нет никаких гарантий что данная установка будет доступна в следующих версиях Firebird.

Установка параметра RelaxedAliasChecking

```
RelaxedAliasChecking = 0
```

2.6.27. ConnectionTimeout

Область действия:

Per-connection.

Тип параметра:

Integer.

С помощью параметра `ConnectionTimeout` устанавливается ограничение на время ожидания соединения. После того как порог, установленный значением параметра, будет превышен, попытка соединения будет признана неудачной.

Параметр `ConnectionTimeout` имеет целочисленный тип и измеряется в секундах. Значение по умолчанию равно 180 секунд. Минимальное значение равно 0. Максимально допустимое значение равно 2147483647.

Установка параметра ConnectionTimeout

```
ConnectionTimeout = 180
```

2.6.28. WireCrypt

Область действия:

Per-connection.

Тип параметра:

String (предопределённые значения).

Параметр устанавливает, следует ли шифровать сетевое соединение. Он может принимать три возможных значения: `Required`, `Enabled`, `Disabled`. По-умолчанию установлено, что шифрование является обязательным (`Required`) для подключений, поступающих на сервер и включенным (`Enabled`) для подключений, исходящих с клиента.

Установка параметра WireCrypt

```
WireCrypt = Enabled # для клиента
```

```
WireCrypt = Required # для сервера
```

Чтобы получить доступ к серверу с использованием традиционной (`Legacy_Auth`) аутентификации, параметр `WireCrypt` в файле конфигурации сервера должен быть включен (`Enabled`) или выключен (`Disabled`).

Правила очень просты: если на одной стороне стоит значение `WireCrypt = Required`, а на другой установлено значение `Disabled`, то первая сторона отклоняет соединение и оно не устанавливается. Если на одной стороне стоит значение `WireCrypt = Enabled`, то на другой шифрования может и не быть вовсе.

Отсутствующий подключаемый модуль `WireCryptPlugin` или ключ шифрования в случаях, когда канал должен быть зашифрован, также препятствует соединению.

Во всех остальных случаях соединение устанавливается без шифрования, если хотя бы одна сторона имеет `WireCrypt = Disabled`. В других случаях устанавливается шифрованное соединение.

Таблица 2. Совместимость параметров WireCrypt на клиенте и на сервере

	Disabled	Enabled	Required
Disabled	Шифрование отключено	Шифрование отключено	Ошибка соединения

Enabled	Шифрование отключено	Шифрование включено, если плагин аутентификации предоставляет ключ шифрования. Иначе шифрования нет.	Шифрование включено, если плагин аутентификации предоставляет ключ шифрования. Иначе ошибка подключения.
Required	Ошибка соединения	Шифрование включено, если плагин аутентификации предоставляет ключ шифрования. Иначе ошибка подключения.	Шифрование включено, если плагин аутентификации предоставляет ключ шифрования. Иначе ошибка подключения.

Смотри также:

AuthServer и AuthClient, WireCryptPlugin.

2.6.29. WireCompression

Область действия:

Per-connection.

Тип параметра:

Boolean.

Параметр может быть задействован и в *firebird.conf* и в *databases.conf*; он включает или отключает сжатие данных, передающихся по сети.

По-умолчанию сжатие отключено.

Установка параметра WireCompression

```
WireCompression = false
```

Для правильной работы параметра требуется корректная настройка как на сервере, так и на клиенте:

- Чтобы включить `WireCompression` на стороне сервера, поставьте параметр в значение `true` в файле *firebird.conf* или *database.conf*.
- Для того чтобы активировать `WireCompression` на стороне клиента, передайте соответствующий тег в вызов DPB и SPB:

```
isc_dbp_config <string-length> "WireCompression=true"
isc_sbp_config <string-length> "WireCompression=true"
```

- Клиент и сервер должны использовать протокол >= 13 (Firebird 3.0 и старше).

2.6.30. DummyPacketInterval

Область действия:

Per-connection.

Тип параметра:

Integer.

Параметр `DummyPacketInterval` используется для того, чтобы установить число секунд ожидания в “тихом” режиме, прежде чем сервер начнет посылать пустые пакеты для подтверждения соединения.

Параметр имеет целочисленный тип и измеряется в секундах. Значение по умолчанию равно 0 секунд. Максимально допустимое значение равно 2147483647 секунд.

Эта опция может привести к зависанию или сбоям Windows NT4 или Windows 2000 pre SP3 на стороне клиента, как описано в 296265, или не может предотвратить возможное отключение неактивного клиента для других ОС.

Обычно Firebird использует опцию сокета SO_KEEPALIVE, чтобы следить за активными подключениями по TCP/IP протоколу. Если вас не устраивает заданное по умолчанию 2-часовое время ожидания (keepalive), то следует изменить параметры настройки своей операционной системы соответственно:

- в Unix-подобных ОС необходимо изменить содержимое */proc/sys/net/ipv4/tcp_keepalive*_*;
- в Windows необходимо вносить изменения в ветку реестра *HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters* ключи (KeepAliveTime, KeepAliveInterva, TcpMaxDataRetransmissions).

2.6.31. RemoteServicePort или RemoteServiceName

Область действия:

Per-connection.

Тип параметра:

Integer или String.

Параметры RemoteServiceName и RemoteServicePort используются для установки номера порта или имени сервиса, которые будут использоваться для клиентских соединений с базами данных.

Параметр RemoteServiceName имеет строковый тип. Значение по умолчанию равно gds_db.

Параметр RemoteServicePort имеет целочисленный тип. Значение по умолчанию равно 3050.

Установка параметра RemoteServiceName

```
RemoteServiceName = gds_db
```

Установка параметра RemoteServicePort

```
RemoteServicePort = 3050
```

Изменять следует только один из этих параметров, не оба сразу. Сервер ищет номер порта для клиентских соединений в следующем порядке — сначала RemoteServiceName (соответствующая значению параметра запись ищется в файле *services*), затем RemoteServicePort.

Обычно один из этих параметров (чаще всего RemoteServicePort) меняют только на стороне севера в *firebird.conf*, поскольку на клиентской стороне имя сервиса или номер порта можно указать в строке подключения.

2.6.32. RemoteAuxPort

Область действия:

Per-connection.

Тип параметра:

Integer.

Параметр RemoteAuxPort определяет номер TCP-порта, который будет использоваться для передачи уведомлений о событиях сервера.

Параметр RemoteAuxPort имеет целочисленный тип. Значение по умолчанию равно 0. В этом случае номер порта будет выбираться случайно.

Установка параметра RemoteAuxPort

```
RemoteAuxPort = 0
```

2.6.33. TcpRemoteBufferSize

Область действия:

Per-connection.

Тип параметра:

Integer.

Параметр `TcpRemoteBufferSize` определяет размер TCP/IP буфера для обмена сообщениями между сервером и клиентом. Firebird может делать упреждающее чтение клиентом и может отправлять несколько строк за один сетевой пакет. Чем больше размер пакета, тем больше данных будет передаваться за одну передачу.

Параметр имеет целочисленный тип и измеряется в байтах. Значение по умолчанию равно 8192. Минимально допустимое значение равно 1448. Максимальное значение равно 32767.

Установка параметра TcpRemoteBufferSize

```
TcpRemoteBufferSize = 8192
```

2.6.34. TcpNoNagle

Область действия:

Per-connection.

Тип параметра:

Boolean.

Параметр `TcpNoNagle` включает или отключает использование Nagle алгоритма (опция TCP_NODELAY для сокета) в TCP/IP соединениях.

В Linux по умолчанию библиотека сокетов минимизирует количество физических записей путем буферизации записей перед фактической передачей данных. Для этого используется встроенный алгоритм, известный как Nagle's Algorithm. Он был разработан, для того, чтобы избежать проблем с маленькими пакетами в медленных сетях.

Параметр имеет логический тип. По умолчанию значение параметра равно 1 (истина). В этом случае буферизация не используется. На медленных сетях в Linux это позволяет увеличить скорость передачи.

Установка параметра TcpNoNagle

```
TcpNoNagle = 1
```

2.6.35. TcpLoopbackFastPath

Тип параметра:

Boolean.

Параметр `TcpLoopbackFastPath` включает или отключает использование функции "TCP

Loopback Fast Path” (SIO_LOOPBACK_FAST_PATH).

Применим только в Windows (версия 8/2012 и выше). Параметр доступен начиная с Firebird 3.0.5.

Параметр имеет логический тип. По умолчанию значение параметра равно 1 (истина).

Установка параметра TcpLoopbackFastPath

```
TcpLoopbackFastPath = 1
```

2.6.36. IPv6V6Only

Область действия:

Per-server.

Тип параметра:

Boolean.

Этот параметр можно устанавливать только в *firebird.conf*. Firebird поддерживает **IPv6** подключение на стороне сервера и клиента. Параметр может принимать значения `true`/`false`, `1`/`0` или `Yes`/`No`. Значение по умолчанию равно `false`.

В Windows эта опция поддерживается, начиная с Windows Vista, в более ранних версиях значение параметра всегда равно `true`.

Установка параметра IPv6V6Only

```
IPv6V6Only = 0
```

Сервер

По-умолчанию, сервер прослушивает пустой **IPv6** адрес (`::`) и принимает все входящие подключения, будь то **IPv4** или **IPv6** (`IPv6V6Only = false`). Если параметр установлен в `true`, сервер, прослушивая явно или неявно пустой IPv6 адрес, принимает только IPv6 подключения.

Клиент

Адреса **IPv6** отображаются как восемь четырёхзначных шестнадцатеричных чисел (то есть групп по четыре символа), разделённых двоеточием. В строке подключения необходимо заключать **IPv6** адрес в квадратные скобки, чтобы разрешить неоднозначность с использованием двоеточия в качестве разделителя между IP адресом хоста и путем к базе данных. К примеру:

```
connect '[2014:1234::5]:test';
connect '[2014:1234::5]/3049:/srv/firebird/test.fdb';
```

2.6.37. RemoteBindAddress

Область действия:

Per-server.

Тип параметра:

String.

Параметр `RemoteBindAddress` позволяет привязать входящие соединения к IP адресу определенной сетевой карты. При этом все входящие соединения через другие сетевые интерфейсы будут запрещены. По умолчанию подключения из любого доступного сетевого интерфейса разрешены. Если вы используете классический сервер, этот параметр предназначен только для Windows. В Linux, BSD или Mac OS X с сервером Classic используйте конфигурационный файл xinetd или launchd (параметр `bind`).

Параметр имеет строковый тип. По умолчанию его значение равно пустой строке (разрешены соединения с любого IP адреса).

Установка параметра RemoteBindAddress

```
RemoteBindAddress =
```

2.6.38. LockMemSize

Область действия:

Per-database.

Тип параметра:

Integer.

Значение параметра `LockMemSize` определяет объем памяти, которая будет выделена менеджеру блокировок. В архитектуре Classic и SuperClassic данный параметр используется для начального распределения, далее таблица расширяется динамически до предела памяти. В архитектуре Super значение параметра определяет начальное распределение и предел выделяемой памяти.

Параметр имеет целочисленный тип. Единица измерения — байты. Значение по умолчанию равно 1 Мбайт. Минимальное значение равно 0. максимальное — 2 Гбайта.

Установка параметра LockMemSize

```
LockMemSize = 1M
```

На размер таблицы блокировок влияют:

1. В архитектурах Classic и SuperClassic размер страничного кэша (Super сервер для блокировки страниц использует лёгковесные защёлки). Страница, помещенная в кэш,

блокируется, как минимум, один раз, страницы, которые читаются несколькими клиентами, могут блокироваться несколько раз.

2. Число одновременных транзакций. Каждая транзакция имеет блокировку. Блокировка используется для синхронизации транзакций.
3. События. Механизм оповещения о событиях использует блокировки. Число событий и число клиентов, ожидающих эти события, влияют на размер таблицы блокировок.

2.6.39. LockAcquireSpins

Область действия:

Per-database.

Тип параметра:

Integer.

В архитектуре сервера Classic только одно клиентское соединение может обратиться к таблице блокировки в одно и то же время. Доступ к таблице блокировки управляется с помощью mutex(a). Mutex может быть затребован в условном, либо безусловном режиме. Если mutex затребован в условном режиме, то ожидание является отказом, и запрос должен повториться. В безусловном режиме mutex будет ожидаться до тех пор, пока не будет получен.

Параметр `LockAcquireSpins` имеет целочисленный тип. Его значение устанавливает количество попыток, которые будут сделаны в условном режиме. По умолчанию значение параметра равно 0, в этом случае будет использоваться безусловный режим.

Параметр имеет эффект только на SMP (симметричных мультипроцессорных) системах.

Установка параметра LockAcquireSpins

```
LockAcquireSpins = 2
```

В залоговке лок-таблицы этот параметр отображается как “Spin count” в строке вида:

```
Acquires:   2150, Acquire blocks:     14, Spin count:   0
```

2.6.40. LockHashSlots

Область действия:

Per-database.

Тип параметра:

Integer.

Параметр `LockHashSlots` используется для настройки числа слотов хэш таблицы блокировок. Чем больше слотов используется, тем короче получаются хэш цепочки, что увеличивает производительность при повышенной нагрузке.

Параметр имеет целочисленный тип. Максимально допустимое значение этого параметра равно 65521 (значения больше этого порога игнорируются и используется всё равно именно этот порог: 65521).

Значение по умолчанию (8191) можно считать подходящим для работы 100 пользователей и умалчиваемом размере страничного кеша (256).

В качестве значения рекомендуется указывать простое число, чтобы хэш-алгоритм производил хорошее распределение.

Установка параметра LockHashSlots

```
LockHashSlots = 30011
```

Увеличение значения данного параметра необходимо только при высокой загрузке (одновременно с ним следует увеличить и параметр `LockMemSize` на тот же процент). Он вычисляется с использованием утилиты Lock Print по следующему принципу.

Запускаем утилиту

```
fb_lock_print -d <database> | <alias>
```

В группе заголовка блока (**LOCK_HEADER BLOCK**), которая описывает основную конфигурацию и состояние таблицы блокировок, смотрим значение элемента **Hash lengths** (длина хэш цепочки). Этот элемент сообщает минимальную, среднюю и максимальную длину цепочки слотов. Чем длиннее будут цепочки, тем медленнее будет работать менеджер блокировок. Если среднее значение больше 3 или максимальное больше 10, то это означает, что слотов недостаточно. Поэтому следует увеличить параметр `LockHashSlots` в 2-3 раза (при этом взять простое число).

Для применения параметра необходимо, чтобы сервер пересоздал таблицу блокировок (при этом в системе не должно остаться подключений и старой таблицы блокировок).

2.6.41. EventMemSize

Область действия:

Per-database.

Тип параметра:

Integer.

Значение параметра `EventMemSize` определяет объем разделяемой памяти, которая будет выделена менеджеру событий.

Параметр `EventMemSize` имеет целочисленный тип. Единица измерения — байты. Значение по умолчанию равно 64 Кбайта. Минимально допустимое значение равно 0. Максимальное значение равно 2 Гбайта.

Установка параметра EventMemSize

```
EventMemSize = 64K
```

2.7. Настройки ядра

2.7.1. CpuAffinityMask

Область действия:

Per-server.

Тип параметра:

Integer.

Параметр `CpuAffinityMask` позволяет указать, какие процессоры (ядра) будут использоваться сервером (только для ОС Windows в архитектуре Super сервер).

Параметр имеет эффект только в SMP (симметричных мультипроцессорных) системах.

Параметр имеет целочисленный тип. Значение параметра соответствует элементам битового массива, в котором каждый бит представляет центральный процессор. Таким образом, чтобы использовать только первый процессор, значение параметра должно быть равно 1. Чтобы использовать и центральный процессор 1, и центральный процессор 2 — 3. Чтобы использовать центральный процессор 2, и центральный процессор 3 — 6. Значение по умолчанию равно 0 (могут быть использованы все доступные процессоры).

Установка параметра CpuAffinityMask

```
CpuAffinityMask = 64K
```

2.7.2. GCPolicy

Область действия:

Per-database.

Тип параметра:

String.

Параметр `GCPolicy` используется для управления работой “сборщика мусора”. Параметр имеет строковый тип. Возможные значения параметра:

- `background` — сборщик мусора работает как фоновый, собирая мусор в отдельном потоке;
- `cooperative` — сборщик мусора работает в оперативном режиме, собирая мусор немедленно при чтении “мусорных” версий;
- `combined` — сборщик мусора работает в оперативном режиме, но если мусор собрать не удается, то о “замусоренных” страницах сигнализируется фоновому сборщику мусора.

По умолчанию в архитектуре Super сервера “сборщик мусора” работает в комбинированном режиме. В архитектурах Classic и SuperClassic этот параметр игнорируется, а “сборщик мусора” всегда работает в оперативном режиме.

Установка параметра GCPolicy

```
GCPolicy = combined
```

2.7.3. SecurityDatabase

Область действия:

Per-database.

Тип параметра:

String (путь к базе данных или алиас).

Определяет имя и расположение базы данных безопасности, в которой хранятся имена пользователей и пароли, используемые сервером для проверки удаленных подключений.

По-умолчанию в *firebird.conf*:

```
SecurityDatabase = $(dir_secDb)/security3.fdb
```

Параметр может быть переопределен для определенной базы данных в файле *databases.conf*. Любая база данных может быть базой данных безопасности, в том числе и для самой себя.

2.8. Настройки для Windows систем

2.8.1. GuardianOption

Область действия:

Per-server.

Тип параметра:

Boolean.

Параметр определяет должен ли сторож (Guardian) запускать сервер после того, как его работа была завершена некорректно.

- 0 — сервер стартует единожды;
- 1 — сервер стартует каждый раз после некорректного завершения.

Параметр имеет логический тип. Значение по умолчанию равно 1 (истина).

Установка параметра GuardianOption

```
GuardianOption = 1
```

В современных версиях Windows службы могут перезапускаться при аварийном завершении, если указаны соответствующие настройки. Это делает сторож (Guardian) не нужным при установке Firebird в качестве службы. Однако он всё ещё может быть полезным, если Firebird стартует в режиме приложения.

2.8.2. ProcessPriorityLevel

Область действия:

Per-server.

Тип параметра:

Integer.

Параметр определяет уровень приоритетов процессов сервера Firebird. Параметр имеет целочисленный тип и может принимать значения:

- 0 — нормальный приоритет (значение по умолчанию);
- положительное значение — повышенный приоритет (тоже самое что опция -B в командной строке);
- отрицательное значение — пониженный приоритет.

Все изменения данного параметра должны быть тщательно проверены, чтобы гарантировать, что сервер продолжает обрабатывать запросы.

Установка параметра ProcessPriorityLevel

```
ProcessPriorityLevel = 0
```

2.8.3. IpcName

Область действия:

Per-connection.

Тип параметра:

String.

Параметр `IpcName` определяет имя области разделяемой памяти используемой в качестве транспортного канала в локальном протоколе. Параметр имеет строковый тип. Значение по умолчанию равно `FIREBIRD`.

Локальный протокол не совместим с Firebird версия которого меньше 2.0.

Установка параметра IpcName

```
IpcName = FIREBIRD
```

- Сервер может регистрировать объекты в пространстве имен Global, только если он выполняется под учетной записью с привилегией SE_CREATE_GLOBAL_NAME. Это означает, что, если вы работаете под ограниченной учетной записью в Vista, XP SP2 или 2000 SP4, возможность использования локального протокола для других сеансов будет недоступна.
- Если на одном сервере установлено несколько экземпляров Firebird, и все они содержат одно и то же значение параметра IpcName (явно заданное или умалчиваемое), то firebird.log всех запускаемых экземпляров, начиная со второго, будет при каждом (ре-)старте пополняться сообщением вида:

```
XNET error: XNET server initialization failed. Probably another
instance
of server is already running.
operating system directive CreateMutex failed
Невозможно создать файл, так как он уже существует.
```

Для каждого подключения к СУБД может быть назначено индивидуальное значение этого параметра.

2.8.4. RemotePipeName

Область действия:

Per-connection.

Тип параметра:

String.

Параметр `RemotePipeName` определяет название канала (Pipe), используемого как

транспортный канал в протоколе NetBEUI. Название канала в протоколе NetBEUI имеет то же самое значение, что и номер порта для протокола TCP/IP.

Параметр имеет строковый тип. Значение по умолчанию равно `interbas` и совместимо с InterBase/Firebird 1.0.

Установка параметра RemotePipeName

```
RemotePipeName = interbas
```

2.9. Настройки для Unix/Linux систем

2.9.1. Redirection

Область действия:

Per-server.

Тип параметра:

Boolean.

Параметр `Redirection` используется для отключения защиты от переадресации запросов на другие сервера. Возможность переадресации запросов на другие серверы изначально присутствовала в InterBase. Но она была исключена корпорацией Borland в InterBase 6.0 после доработки добавившей SQL-диалекты. Возможность перенаправления запросов была восстановлена в Firebird 2.0.

На сегодняшний день использование этой возможности (прокси сервер) представляет угрозу безопасности. Например, вы используете защищенный сервер Firebird, доступ к которому осуществляется из глобальной сети. В этом случае, если у сервера есть доступ к локальной сети, то он будет исполнять роль шлюза для входящих запросов типа:

```
firebird.your.domain.com:internal_server:/private/database.fdb
```

При этом злоумышленнику достаточно знать имя или IP-адрес хоста вашей локальной сети, потому что для соединения не требуется знать логин и пароль на внешнем сервере. Такой шлюз позволяет обойти систему сетевой защиты, установленную в вашей локальной сети.

Параметр имеет логический тип. Значение по умолчанию равно 0 (`false`). В этом случае возможность перенаправления запросов отключена. Для включения этой опции следует значение параметра выставить равным 1 (`true`).

Установка параметра Redirection

```
Redirection = 0
```

Не включайте эту опцию если вы не уверены, что именно она делает.

2.10. Настройки архитектуры

2.10.1. ServerMode

Область действия:

Per-server.

Тип параметра:

String.

Параметр `ServerMode` определяет архитектуру сервера. Существует 3 варианта архитектуры:

- `Super` (или `ThreadedDedicated`) — база данных открывается эксклюзивно одним серверным процессом, им же обслуживаются все соединения с базой данных. Подключения обрабатываются потоками из общего пула; используется общий страничный кэш на каждую базу данных для всех соединений и общий кэш для временного пространства (сортировок).
- `SuperClassic` (или `ThreadedShared`) — базы данных открываются одним серверным процессом, но доступ не исключительный — embedded процессы могут открыть одновременно одну ту же базу. Подключения обрабатываются потоками из общего пула; используется собственный страничный кэш для каждого соединения и общий кэш для временного пространства (сортировок).
- `Classic` (или `MultiProcess`) — создаётся отдельный процесс на каждое соединение с базой данных. Каждая база данных может быть открыта несколькими процессами (включая локальные для embedded доступа); используется отдельный кеш страниц на каждое соединение и отдельный кэш для временного пространства (сортировок).