Руководство администратора OpenLDAP 2.4
The OpenLDAP Project <http://www.openldap.org/>
19 августа 2013 г.

Содержание

Предисловие

Авторские права

соответственно.

На части программного обеспечения OpenLDAP и этого документа могут налагаться дополнительные ограничения авторскими правами других организаций и/или лиц. Чтобы получить информацию о дополнительных авторских правах, просматривайте отдельные файлы исходного кода, которые собираетесь использовать.

Рамки этого документа

Этот документ содержит руководство по установке программного обеспечения OpenLDAP 2.4 (http://www.openldap.org/software/) на UNIX (и UNIX-подобных) системах. Документ предназначен для опытных системных администраторов, которые имеют базовые понятия о службах каталогов, основанных на LDAP.

Этот документ нужно использовать совместно с другими информационными ресурсами OpenLDAP, предоставляемыми с пакетом программного обеспечения и на сайте проекта (http://www.OpenLDAP.org/) в World Wide Web. Ниже перечислены ресурсы, доступные на сайте.

Ресурсы OpenLDAP

Ресурс	**URL**
Каталог документации	http://www.OpenLDAP.org/doc/
Часто задаваемые вопросы (FAQ)	http://www.OpenLDAP.org/faq/
Система отслеживания проблемных вопросов	http://www.OpenLDAP.org/its/
Списки рассылки	http://www.OpenLDAP.org/lists/
Man-страницы	http://www.OpenLDAP.org/software/man.cgi
Страницы программного обеспечения	http://www.OpenLDAP.org/software/
Страницы поддержки	http://www.OpenLDAP.org/support/

Данный документ не является полным справочником по программному обеспечению OpenLDAP; точной документацией являются man-страницы. Лучше всего использовать man-страницы, установленные в Вашей системе с Вашей версией программного обеспечения OpenLDAP, тогда Вы точно имеете дело с документацией, соответствующей работающим у Вас программам. Хотя сайт OpenLDAP также предоставляет man-станицы для удобства, Вы не можете быть уверены в том, что они соответствуют конкретной версии программ, работающих у Вас.

Благодарности

Проект OpenLDAP - это команда добровольцев. Данного документа не существовало бы, если бы они не вложили в него свои время и энергию.

Проект OpenLDAP также хотел бы поблагодарить Команду LDAP Мичиганского Университета за построение основы программного обеспечения LDAP и информацию, на основании которой было построено программное обеспечение OpenLDAP. Данный документ основан на документе Мичиганского Университета: The SLAPD and SLURPD Administrators Guide.

Поправки

Предложения по улучшению и коррекции этого документа подавайте в Систему отслеживания проблемных вопросов OpenLDAP (http://www.openldap.org/its/).

Об этом документе

Этот документ подготовлен с использованием системы документирования Simple Document Format (SDF) (http://search.cpan.org/src/IANC/sdf-2.001/doc/catalog.html) разработанной *Ian Clatworthy*. Инструменты для SDF можно найти на CPAN (http://search.cpan.org/search?query=SDF&mode=dist).

К содержанию

1. Введение в службы каталогов OpenLDAP

Эта документация рассказывает о том, как собрать, настроить и эксплуатировать программное обеспечение OpenLDAP для организации службы каталогов. В ней детально описаны конфигурирование и запуск автономного демона LDAP (Standalone LDAP Daemon), *slapd*(8). Документация предназначена как для новичков, так и для опытных системных администраторов. В данном разделе дается базовое введение в службы каталогов и, в частности, в службы каталогов, построенные на *slapd*(8). Данное введение не претендует на полноту, а даёт лишь тот минимум, который необходим, чтобы начать изучение LDAP, X.500 и служб каталогов.

1.1. Что такое служба каталогов?

Каталог - это специализированная база данных, предназначенная для поиска и просмотра информации, а также поддерживающая наполнение данными и их обновление в качестве дополнительных функций.

Замечание: Некоторые называют каталог просто базой данных, оптимизированной для запросов на чтение. Это определение, в лучшем случае, чрезмерно упрощено.

Каталоги имеют тенденцию содержать описательную информацию, основанную на атрибутах, и поддерживать сложные способы фильтрации. Каталоги обычно не поддерживают механизмы транзакций и откатов (roll-back), применяемые в СУБД, ориентированных на комплексные обновления большого объема данных. Обновления в каталогах (если они вообще разрешены), обычно происходят по простой схеме: "изменить всё или ничего". Каталоги обычно оптимизируются на скорейшую выдачу результата при поиске среди больших объемов информации. Они также могут иметь возможность репликации информации, то есть создания удалённых копий каталога с целью повышения доступности информации, надёжности её хранения и снижения времени отклика. В процессе репликации, до полного её окончания, допустимо временное рассогласование информации между репликами.

Существует много методов организации службы каталогов. Различия могут касаться типов информации, хранимой в каталоге, выдвижения различных требований к обращению и обновлению этой информации, ссылкам на неё, организации системы разграничения доступа к информации, и т. д. Некоторые службы каталогов могут быть *локальными*, предоставляющими услуги в ограниченном контексте (например, для службы finger на отдельностоящей машине). Другие службы глобальны, предоставляют услуги в гораздо более широком контексте (например, всему Internet). Глобальные службы обычно являются *распределенными*. Это означает, что содержащаяся в них информация распределена между многими компьютерами, взаимодействующими друг с другом для предоставления услуг службы каталогов. Обычно глобальная служба каталогов определяет унифицированное *пространство имён*, чтобы пользователь получал одинаковый результат независимо от того, откуда он производит запрос и к какому серверу службы обращается.

Веб-каталог, такой как *Open Directory Project* <http://dmoz.org>, - хороший пример службы каталогов. Эта служба каталогизирует веб-страницы и специально разработана для просмотра и поиска.

Некоторые приводят Domain Name System (DNS) в качестве примера глобально распределённой службы каталогов, однако DNS не доступен ни для просмотра, ни для поиска в прямом смысле этого слова. Более правильно было бы назвать его глобально распределённой службой ответов на конкретно поставленные вопросы.

1.2. Что такое LDAP?

LDAP - это аббревиатура от Lightweight Directory Access Protocol. Как следует из названия, это облегчённый протокол доступа к службам каталогов, предназначенный для доступа к службам каталогов на основе X.500. LDAP работает поверх TCP/IP или других ориентированных на соединение сетевых протоколов. LDAP стандартизирован в качестве протокола IETF, и его описание можно найти в "Lightweight Directory Access

Protocol (LDAP) Technical Specification Road Map" ("Описание технической спецификации Lightweight Directory Access Protocol (LDAP)") RFC4510.

Данный подраздел дает некоторое представление о LDAP с точки зрения пользователя.

Какого рода информация может храниться в каталоге? Информационная модель LDAP основана на *записях (entry)*. Запись - это коллекция атрибутов (attribute), обладающая *уникальным именем* (Distinguished Name, DN). DN глобально-уникально для всего каталога и служит для однозначного указания на запись. Каждый атрибут записи имеет свой *тип (type)* и одно или несколько *значений (value)*. Обычно типы - это мнемонические строки, в которых отражено назначение атрибута, например "`cn`" - для общепринятого имени (common name), или "`mail`" - для адреса электронной почты. Синтаксис значений зависит от типа атрибута. Например, атрибут `cn` может содержать значение `Babs Jensen`. Атрибут `mail` может содержать значение "`babs@example.com`". Атрибут `jpegPhoto` будет содержать фотографию в бинарном формате JPEG.

Как организовано размещение информации? Записи каталога LDAP выстраиваются в виде иерархической древовидной структуры. Традиционно, эта структура отражает географическое и/или организационное устройство хранимых данных. В вершине дерева располагаются записи, представляющие собой страны. Под ними располагаются записи, представляющие области стран и организации. Еще ниже располагаются записи, отражающие подразделения организаций, людей, принтеры, документы, или просто всё то, что Вы захотите включить в каталог. На рисунке 1.1 показан пример дерева каталога LDAP, использующего традиционное именование записей.

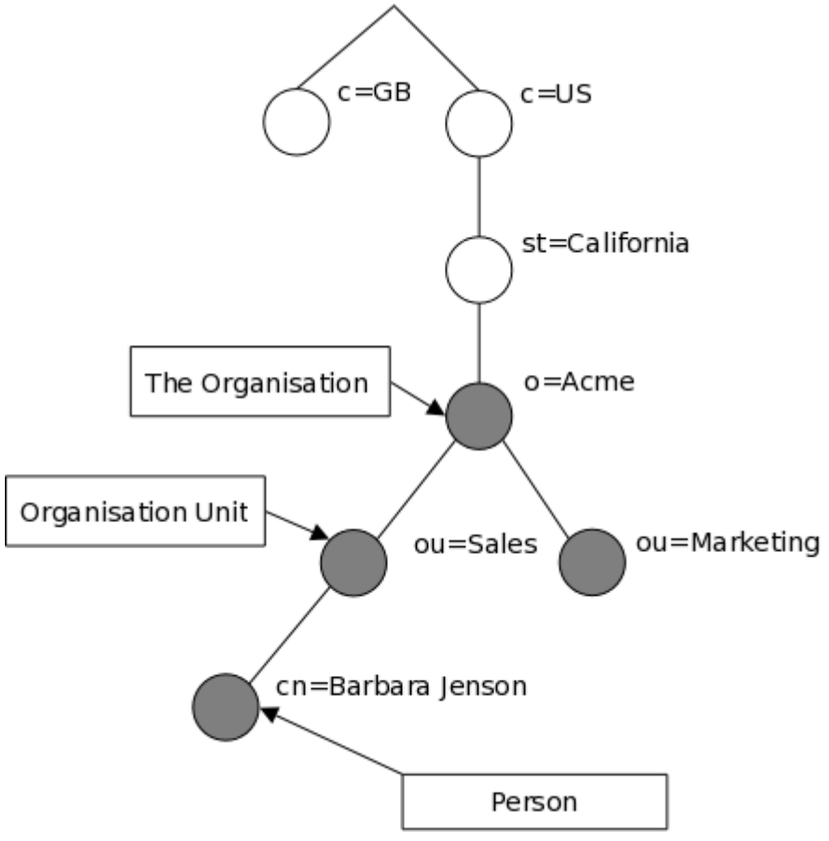

Рисунок 1.1: Дерево каталога LDAP (традиционное именование записей)

Построение дерева может быть также основано на доменных именах Internet. Этот подход к именованию записей становится всё более популярным, поскольку позволяет обращаться к службам каталогов по аналогии с доменами *DNS*. На рисунке 1.2 показан пример дерева каталога LDAP, использующего именование записей на основе доменов.

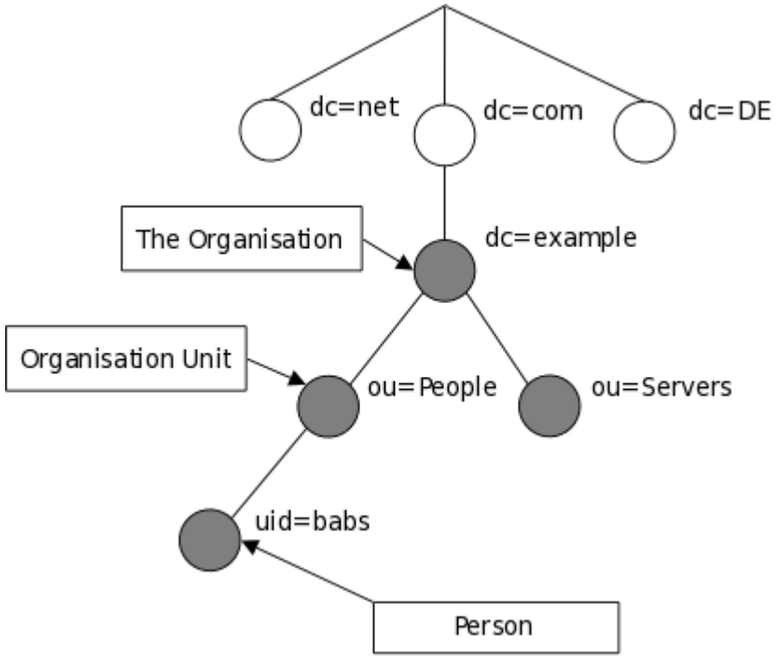

Рисунок 1.2: Дерево каталога LDAP (Internet-именование записей)

Кроме того, LDAP, посредством специального атрибута `objectClass`, позволяет контролировать, какие атрибуты обязательны и какие допустимы в той или иной записи. Значения атрибута `objectClass` определяются правилами *схемы (schema)*, которым должны подчиняться записи.

Как можно обратиться к информации? К записи обращаются по ее уникальному имени, которое состоит из собственно имени записи (так называемое *относительное уникальное имя* (Relative Distinguished Name, RDN) с прибавлением к нему имён записей-предков. Так, запись, описывающая Barbara Jensen в приведенном выше примере с Internet-именованием, имеет RDN `uid=babs`, и DN - `uid=babs,ou=People,dc=example,dc=com`. Полное описание формата DN можно найти RFC4514, "LDAP: String Representation of Distinguished Names" ("LDAP: строковое представление уникальных имен").

Какие манипуляции можно произвести с информацией? В LDAP определены операции для опроса и обновления каталога. К числу последних относятся операции добавления и удаления записи из каталога, изменения существующей записи и изменения названия записи. Однако, большую часть времени LDAP используется для поиска информации в каталоге. Операции поиска LDAP позволяют производить поиск записей в определённой части каталога по различным критериям, заданным поисковыми фильтрами. У каждой записи, найденной в соответствии с критериями, может быть запрошена информация, содержащаяся в её атрибутах.

К примеру, Вам захотелось найти записи о человеке по имени `Barbara Jensen` во всем подкаталоге, начиная с уровня `dc=example,dc=com` и ниже, и получить адрес электронной почты в каждой найденной записи. LDAP позволяет Вам легко это сделать. Или Вам хочется поискать непосредственно на уровне `st=California,c=US` записи организаций, названия которых содержат строку `Acme` и имеющих номер факса. Такой поиск LDAP тоже позволяет сделать. В следующем подразделе более подробно описано, что Вы можете сделать с LDAP и чем он может быть Вам полезен.

Как информация защищена от несанкционированного доступа? Некоторые службы каталогов не предоставляют никакой защиты, позволяя любому просматривать хранящуюся в них информацию. Однако LDAP предоставляет механизмы для аутентификации клиента, либо других способов доказательства его подлинности серверу каталогов, а также богатые возможности контроля доступа к информации, содержащейся на этом сервере. LDAP также обеспечивает защиту информации в каталоге (её целостность и конфиденциальность).

1.3. Для чего можно использовать LDAP?

Очень хороший вопрос. В общем случае, службу каталогов можно использовать, когда Вам требуется надёжное хранение информации с возможностью централизованного управления и доступа к ней, с использованием стандартизированных методов.

Вот ряд (но, конечно, не полный) самых распространённых примеров промышленного использования служб каталогов:

- Идентификация компьютеров
- Аутентификация пользователей
- Группировка пользователей (в том числе системные группы)
- Адресные книги
- Представление штатно-кадровой структуры организации
- Учет закрепления имущества организации за сотрудниками
- Телефонные справочники
- Управление пользовательскими ресурсами
- Справочники адресов электронной почты
- Хранение конфигурации приложений
- Хранение конфигурации АТС
- и т.д. ...

Для организации каталога под столь разные задачи существуют различные, основанные на стандартах файлы наборов схемы, распространяемые с дистрибутивом. Также Вы можете создать свою собственную спецификацию схемы для решения Вашей задачи.

Всегда найдутся новые способы использования каталогов и применения принципов LDAP для решения различных проблем, поэтому не существует простого ответа на вопрос этого подраздела.

Если есть сомнения, присоединяйтесь к общему форуму для некоммерческих обсуждений и информации, относящейся к LDAP по адресу: http://www.umich.edu/~dirsvcs/ldap/mailinglist.html и спрашивайте.

1.4. Для чего LDAP лучше не использовать?

Если Вы чувствуете, что нужно исхитриться, чтобы заставить каталог делать то, что Вам требуется, то, возможно, стоит поискать альтернативные способы решения задачи. Или, возможно, найдутся более подходящие средства, если Вам всего-лишь нужно приложение для манипуляций и использования собственной информации (о противопоставлении LDAP и реляционных СУБД можно почитать в подразделе Непростые взаимоотношения LDAP и реляционных СУБД).

Чаще всего очевидно, в каких случаях LDAP - верный способ решения Вашей задачи.

1.5. Как работает LDAP?

LDAP использует *клиент-серверную модель*. Один или несколько серверов LDAP содержат информацию, образующую *информационное дерево каталога* (directory information tree, DIT). Клиент подключается к серверу и делает запрос. В ответ сервер отправляет результаты обработки запроса и/или указатель на то, где клиент может получить дополнительные сведения (обычно, на другой сервер LDAP). Независимо от того, к какому серверу LDAP подключается клиент, он увидит одинаковое представление каталога; на записи, расположенные на одном сервере LDAP, будут указывать правильные ссылки при обращении к другому серверу LDAP, и наоборот. Это важная особенность глобальной службы каталогов.

1.6. Как насчёт X.500?

Технически, LDAP - это протокол доступа к службе каталогов X.500, то есть службе каталогов OSI. Изначально, клиенты использовали LDAP для получения доступа к шлюзам службы каталогов X.500. Такие шлюзы использовали LDAP для общения с клиентом, а для обращения к серверу X.500 использовали *протокол доступа к каталогам* (Directory Access Protocol, DAP). DAP - весьма тяжеловесный протокол, функционирующий поверх полного стека протоколов OSI и требующий значительного количества вычислительных ресурсов. LDAP разработан для функционирования поверх TCP/IP, и обеспечивает большую часть функциональности DAP по гораздо более низкой цене.

Хотя LDAP по-прежнему используется для доступа к службе каталогов X.500 через шлюзы, сейчас он чаще непосредственно встраивается в программное обеспечение серверов X.500.

Автономный демон LDAP, или *slapd*(8), можно рассматривать как *легковесный* сервер службы каталогов X.500. Он не реализует X.500 DAP и не поддерживает полные информационные модели X.500.

Если Вы уже используете службу X.500 и DAP и планируете продолжать, Вам, скорее всего, можно не читать это руководство дальше, поскольку оно целиком посвящено работе LDAP с использованием *slapd*(8), без запуска X.500 DAP. Если Вы не используете X.500 DAP, собираетесь прекратить его использовать, либо думаете, стоит ли запускать X.500 DAP, читайте дальше.

Существует возможность переноса данных из службы каталогов LDAP в X.500 DAP DSA. Для этого нужен шлюз LDAP/DAP. В состав программного обеспечения OpenLDAP такой шлюз не входит.

1.7. В чём отличие между LDAPv2 и LDAPv3?

LDAPv3 был разработан в конце 90-х годов для замены LDAPv2. LDAPv3 добавил в LDAP следующие возможности:

- Строгая аутентификация и сервисы безопасности данных с помощью SASL
- Аутентификация с использованием сертификатов и сервисы безопасности данных с помощью TLS (SSL)
- Интернационализация посредством использования Unicode
- Поддержка ссылок и продолжений
- Развёртывание в соответствии со схемой данных
- Расширяемость (средствами контроля, дополнительными операциями, и другими возможностями)

Сейчас LDAPv2 является историческим (RFC3494). Поскольку большинство *так называемых* реализаций LDAPv2 (в том числе *slapd*(8)) не соответствуют техническим спецификациям LDAPv2, совместимость между такими реализациями, декларирующими поддержку LDAPv2, ограничена. Так как LDAPv2 существенно отличается от LDAPv3, поддержка работоспособности одновременно и LDAPv2 и LDAPv3 весьма проблематична. Следует избегать использования LDAPv2, по умолчанию он отключен.

1.8. Непростые взаимоотношения LDAP и реляционных СУБД

Этот вопрос поднимался много раз в различных формах. Однако, чаще всего в такой: *Почему бы OpenLDAP не перейти от Berkeley DB к использованию полноценной реляционной системы управления базами данных (СУБД)?* Такой переход мог бы сделать *OpenLDAP* быстрее или дать другие преимущества за счет используемых в СУБД коммерческого класса сложных алгоритмов, и, с другой стороны, дал бы возможность другим приложениям работать с теми же данными непосредственно из БД.

Если ответить коротко, использование встроенной базы данных и простой системы индексирования позволяет OpenLDAP обеспечивать высокую производительность и масштабируемость без потери надёжности. OpenLDAP использует многопользовательскую СУБД Berkeley DB с поддержкой транзакций. То же самое программное обеспечение используется в ведущих коммерческих службах каталогов.

Теперь попробуем ответить на вопрос более развёрнуто. Все мы постоянно сталкиваемся с выбором между реляционными СУБД и каталогами. Это нелёгкий выбор, и простого ответа тут не существует.

Кажется, что использование в качестве хранилища каталога реляционной СУБД решит все подобные проблемы. Но тут кроется подвох. Всё дело в том, что информационные модели каталога и реляционной БД очень разные. Попытка проецирования данных каталога в реляционную БД потребует разделения данных на несколько таблиц.

Возьмём, к примеру, объектный класс person. В нём определены обязательные типы атрибутов objectClass, sn и cn, а также необязательные типы атрибутов userPassword, telephoneNumber, seeAlso и description. Все эти атрибуты могут иметь несколько значений, таким образом нормализация потребует поместить каждый тип атрибута в отдельную таблицу.

Теперь нужно подумать о ключевых полях для этих таблиц. В качестве первичного ключа может быть использован DN, но в большинстве реализаций баз данных это будет весьма неэффективно.

Еще одной большой проблемой может стать обращение к различным областям жесткого диска при поиске данных, представляющих одну и ту же запись каталога. Для одних приложений это не вызовет затруднений, но для большинства из них это обернётся потерей производительности.

В главную таблицу, хранящую сведения о записях каталога, могут быть помещены только данные из обязательных для объектного класса типов атрибутов, которые могут иметь только одно значение. Кроме того, туда можно поместить данные из необязательных атрибутов с одним значением, и записывать в такие поля NULL (или ещё что-нибудь), если атрибут не задан.

Однако, у записи каталога может быть несколько объектных классов, и они могут иерархически наследоваться один от другого. Запись объектного класса organizationalPerson будет иметь атрибуты объектного класса person плюс ряд дополнительных, к тому же некоторые ранее необязательные атрибуты могут стать обязательными.

Что же делать? Нужно ли заводить разные таблицы под разные объектные классы? В таком случае запись, описывающая человека, будет иметь одну строку в таблице person, еще одну в organizationalPerson, и т.д. Или не стоит ничего помещать в таблицу person а всё записывать только во вторую таблицу?

А что нам делать с фильтрами типа (cn=*), где cn - тип атрибута, который может использоваться в очень многих объектных классах? Придётся пройтись по всем таблицам, где можно встретить такой атрибут? Не очень привлекательно.

Когда наши рассуждения достигли данной точки, на ум приходят три подхода. Первый из них заключается в полной нормализации, то есть в помещении каждого типа атрибута, вне зависимости от того, какие данные в нём хранятся, в отдельную таблицу. В простейшем случае в качестве первичного ключа можно использовать DN, но это крайне расточительно, поэтому напрашивается присвоение записи уникального числового идентификатора, который в главной таблице будет сопоставляться DN, а потом использоваться в качестве в качестве внешнего ключа в подчиненных таблицах. Такой подход в любом случае будет неэффективным, когда потребуются данные сразу из нескольких атрибутов для одной или ряда записей каталога. Тем не менее, с такой базой данных можно работать из SQL-приложений.

Второй подход заключается в том, чтобы помещать целиком все данные о записи в одном blob-поле таблицы, в которой будут храниться все записи каталога, независимо от их объектных классов, и иметь дополнительные таблицы, содержащие индексы для первой. И это будут не индексы базы данных, а величины, применяемые для оптимизации поиска в конкретной реализации LDAP-сервера. Однако, подобные базы данных становятся непригодными для SQL-запросов. Таким образом, использование полноценной реляционной СУБД не обеспечивает практически никаких преимуществ, и потому бесполезно. Гораздо лучше использовать что-то более легковесное и быстрое, вроде Berkeley DB.

Наконец, совершенно отличный подход заключается в отказе от реализации полноценной модели данных каталога. В этом случае LDAP используется в качестве протокола доступа к данным, которые можно назвать каталогом лишь с ограничениями. К примеру, это может быть каталог в режиме "только для чтения", либо, если разрешены операции обновления, накладываются различные ограничения, такие как присвоение только одного значения атрибутам, которые в полноценной реализации могли бы иметь несколько значений. Либо отсутствие возможности добавить новый объектный класс к существующей записи, или убрать один из тех, которые у неё уже имеются. Диапазон ограничений может варьироваться от вполне безобидных (вроде тех, что налагаются в результате контроля доступа), до прямого нарушения модели данных, но за счет этого можно попытаться организовать LDAP-доступ к уже существующим данным, которые используются другими приложениями. И всё же надо понимать, что такую систему "каталогом" можно назвать с большой натяжкой.

В существующих коммерческих реализациях LDAP-серверов, использующих реляционные базы данных, применяется либо первый, либо третий подход. Всё же хотелось бы отметить, что ни в одной из них применение реляционной СУБД не позволило сделать работу эффективнее, чем при использовании BDB.

Для тех, кто заинтересовался "третьим путём" (представление СУЩЕСТВУЮЩИХ данных, хранящихся в реляционной СУБД, в виде LDAP-дерева, имеющее, с одной стороны, некоторые ограничения по сравнению с классической LDAP-моделью, а с другой стороны позволяющее организовать взаимодействие между LDAP и SQL-приложениями), есть хорошая новость. OpenLDAP включает в себя back-sql - механизм манипуляции данными, который делает это возможным. Он использует ODBC + дополнительную метаинформацию о

трансляции LDAP-запросов в SQL-запросы в схему данных Вашей СУБД, организует различные уровни доступа от "только для чтения" до полного доступа, в зависимости от СУБД, которую Вы используете, и Вашей схемы данных.

За дополнительной информацией о принципах работы и ограничениях обращайтесь к man-странице *slapd-sql*(5) или к разделу Механизмы манипуляции данными. Есть также несколько примеров для разных СУБД в поддиректориях `back-sql/rdbms_depend/*` .

1.9. Что такое slapd и на что он способен?

slapd(8) - это сервер службы каталогов, работающий на очень многих платформах. Вы можете использовать его для организации службы каталогов и настройки её индивидуально под себя. Ваш каталог может хранить информацию практически обо всём, что Вам заблагорассудится. Вы можете подключить его к глобальной службе каталогов, или использовать только в своих интересах. Вот некоторые наиболее интересные возможности и особенности slapd:

LDAPv3: *slapd* реализует версию 3 протокола Lightweight Directory Access Protocol. *slapd* поддерживает работу LDAP поверх как IPv4 так и IPv6, а также Unix IPC.

Simple Authentication and Security Layer: *slapd* поддерживает строгую аутентификацию и безопасность (целостность и конфиденциальность) данных с использованием SASL. Реализация SASL в *slapd* основана на применении программного обеспечения Cyrus SASL с поддержкой ряда механизмов, в том числе DIGEST-MD5, EXTERNAL и GSSAPI.

Transport Layer Security: *slapd* поддерживает аутентификацию на базе сертификатов и безопасность (целостность и конфиденциальность) данных с использованием TLS (или SSL). Реализация TSL в *slapd* может быть основана на применении программного обеспечения OpenSSL, GnuTLS или MozNSS.

Контроль доступа на основе сетевой топологии: *slapd* может быть настроен на запрещение доступа на уровне подключений на основе информации о топологии сети. Данная возможность основана на применении *TCP wrappers*.

Контроль доступа: *slapd* предоставляет богатые и мощные средства контроля доступа к информации в Ваших базах данных. Вы можете контролировать доступ к записям по аутентификационной информации LDAP, IP-адресу, доменному имени и другим критериям. *slapd* поддерживает как *статическую* так и *динамическую* информацию для осуществления контроля доступа.

Интернационализация: *slapd* поддерживает Unicode и языковые теги.

Выбор механизма манипуляции данными: *slapd* поставляется с набором различных механизмов манипуляции на Ваш выбор. Вот некоторые из них: BDB, высокопроизводительный механизм манипуляции с поддержкой транзакций; HDB, иерархичный высокопроизводительный механизм с поддержкой транзакций; *SHELL*, механизм для выполнения произвольных shell-скриптов; PASSWD, простой механизм доступа к файлу *passwd*(5). Механизмы BDB и HDB основаны на применении Oracle Berkeley DB.

Применение нескольких хранилищ данных одновременно: *slapd* может быть настроен для работы с несколькими базами данных одновременно. Это означает, что один сервер *slapd* может обслуживать запросы к нескольким логически различным частям дерева LDAP, с использованием одинаковых или различных механизмов манипуляции данными.

Разнообразные API-модули: Если Вам требуется еще большая гибкость настроек, *slapd* позволяет Вам без труда написать собственные модули. *slapd* состоит из 2-х отдельных частей: интерфейс приёма запросов, обслуживающая общение с клиентами посредством протокола LDAP, и модули, выполняющие специфические задачи, такие как операции с базами данных. Поскольку эти 2 части взаимодействуют друг с другом через чётко определённый C API, Вы можете на его основе писать собственные модули, что может значительно расширить функциональность *slapd*. Также доступен ряд программируемых модулей доступа к базам данных, которые позволяют определить внешние источники данных для *slapd* с использованием популярных языков программирования (Perl, *shell*, and SQL).

Потоки: *slapd* поддерживает разделение на потоки для повышения производительности. Один

многопоточный процесс *slapd* обслуживает все входящие запросы с использованием пула потоков. Это позволяет уменьшить нагрузку на систему, увеличивая тем самым производительность.

Репликация: *slapd* может быть сконфигурирован для выполнения фонового копирования данных каталога. Подобная схема репликации *"один главный/несколько подчиненных серверов"* имеет жизненно важное значение в больших высоко-загруженных системах, где один сервер *slapd* просто не в состоянии обеспечить необходимую доступность и надежность. В экстремально сложных системах с повышенными требованиями к безотказности возможно также использование схемы репликации *"несколько главных серверов"*. В *slapd* включена поддержка *LDAP Sync*-репликации.

Прокси-кэширование: *slapd* может быть сконфигурирован в качестве кэширующего прокси-сервера LDAP.

Настраиваемость: *slapd* может быть очень гибко и разнообразно настраиваться посредством одного единственного конфигурационного файла, который позволяет Вам изменить всё, что Вам только захочется изменить. Опции конфигурации имеют разумные значения по умолчанию, чтобы максимально облегчить Ваш труд. Конфигурация также может быть произведена динамически посредством самого LDAP, что значительно повышает управляемость.

К содержанию

2. Руководство по быстрому развёртыванию и началу работы

Здесь представлено руководство по быстрому развёртыванию и началу работы с программным обеспечением OpenLDAP 2.4, включая Автономный демон LDAP, *slapd*(8).

Данное руководство проведёт Вас по основным этапам, необходимым для установки и настройки программного обеспечения OpenLDAP. Его нужно использовать совместно с другими разделами этого документа, man-страницами и другими материалами, предоставляемыми с дистрибутивом (например, документ INSTALL) или на сайте OpenLDAP (http://www.OpenLDAP.org), в частности с OpenLDAP Software FAQ (http://www.OpenLDAP.org/faq/?file=2).

Если Вы всерьёз намерены запустить ПО OpenLDAP, Вы должны рассмотреть весь этот документ прежде, чем пытаться устанавливать данное программное обеспечение.

Примечание: Данное руководство по быстрому развёртыванию и началу работы не содержит сведений о механизмах строгой аутентификации или сервисах предоставления целостности и конфиденциальности. Эти сервисы описаны в других разделах Руководства администратора OpenLDAP.

---

1. **Получите программное обеспечение**
   Вы можете получить копию программного обеспечения, следуя инструкциям на странице скачиваний ПО OpenLDAP (http://www.openldap.org/software/download/). Рекомендуется, чтобы новые пользователи начинали с самого последнего *релиза*.

2. **Распакуйте дистрибутив**
   Выберите директорию, в которую поместите исходный код, перейдите в неё, и распакуйте дистрибутив следующей командой:
   ```
   gunzip -c openldap-VERSION.tgz | tar xvfB -
   ```
   Затем перейдите в директорию дистрибутива:
   ```
   cd openldap-VERSION
   ```
   Замените VERSION на версию Вашего дистрибутива.

3. **Просмотрите документацию**

Просмотрите идущие с дистрибутивом документы `COPYRIGHT`, `LICENSE`, `README` и `INSTALL`. Документы `COPYRIGHT` и `LICENSE` дают информацию о разрешениях на использование и копирование, а также об ограниченной гарантии пакета OpenLDAP.

Также ознакомьтесь с остальными разделами этого документа. В частности, с разделом Сборка и установка программного обеспечения OpenLDAP, где представлена детальная информация о требованиях, предъявляемых пакетом OpenLDAP к установленному на сервере программному обеспечению, и о процедуре установки.

4. **Выполните команду `configure`**
   Вам нужно запустить предоставляемый скрипт `configure` чтобы *сконфигурировать* дистрибутив для сборки на Вашей системе. Скрипт `configure` принимает множество опций командной строки, которые включают или отключают дополнительные функции программного обеспечения. Обычно у опций есть разумные значения по умолчанию, но вы можете менять их по своему усмотрению. Для получения полного списка опций, которые принимает `configure`, используйте опцию `--help`:
   ```
   ./configure --help
   ```

   Однако, если уж Вы взялись читать ДАННЫЙ раздел руководства, Вы достаточно смелы, чтобы разрешить `configure` самой определить наилучший вариант конфигурации:
   ```
   ./configure
   ```

   В случае, если `configure` не нашла в Вашей системе ничего такого, что ей бы не понравилось, Вы можете переходить к сборке программного обеспечения. Если же `configure` стала жаловаться, что ж, придётся Вам посмотреть раздел *Installation* FAQ (http://www.openldap.org/faq/?file=8) и/или внимательно прочитать раздел Сборка и установка программного обеспечения OpenLDAP этого документа.

5. **Выполните сборку пакета**.
   Следующий шаг - сборка программного обеспечения. Этот шаг состоит из двух частей, сначала мы построим зависимости, а затем скомпилируем пакет:
   ```
   make depend
   make
   ```

   Обе команды make должны завершиться без ошибок.

6. **Протестируйте собранный пакет**.
   Чтобы убедиться в корректности сборки, вы должны запустить набор тестов (это займет несколько минут):
   ```
   make test
   ```

   Будут запущены тесты, применимые к Вашей конфигурации, и они должны успешно пройти. Некоторые тесты, такие, как тест репликации, могут быть пропущены.

7. **Установите программное обеспечение**.
   Теперь всё готово к установке программного обеспечения; обычно установка требует привилегий *администратора*:
   ```
   su root -c 'make install'
   ```

   Пакет будет установлен в директорию `/usr/local` (или ту, которую Вы указали в качестве префикса инсталляции при запуске `configure`).

8. **Отредактируйте конфигурационный файл**.
   С помощью своего любимого текстового редактора отредактируйте поставляемый с дистрибутивом пример конфигурационного файла *slapd.conf*(5) (обычно устанавливается как `/usr/local/etc/openldap/slapd.conf`), чтобы в нём содержалось определение базы данных BDB в таком виде:
   ```
   database bdb
   suffix "dc=<MY-DOMAIN>,dc=<COM>"
   rootdn "cn=Manager,dc=<MY-DOMAIN>,dc=<COM>"
   rootpw secret
   directory /usr/local/var/openldap-data
   ```

   Не забудьте заменить `<MY-DOMAIN>` и `<COM>` на соответствующие компоненты Вашего доменного имени. Например, для `example.com` используйте:

```
database bdb
suffix "dc=example,dc=com"
rootdn "cn=Manager,dc=example,dc=com"
rootpw secret
directory /usr/local/var/openldap-data
```

Если имя Вашего домена содержит дополнительные компоненты, как например `eng.uni.edu.eu`, используйте:

```
database bdb
suffix "dc=eng,dc=uni,dc=edu,dc=eu"
rootdn "cn=Manager,dc=eng,dc=uni,dc=edu,dc=eu"
rootpw secret
directory /usr/local/var/openldap-data
```

Более детальное описание настройки *slapd*(8) можно найти в man-странице *slapd.conf*(5) и в разделе Конфигурационный файл slapd этого документа. Обратите внимание, что указанная директория должна быть создана перед тем, как Вы запустите *slapd*(8).

9. **Запустите SLAPD**.
   Пора запустить Автономный демон LDAP, *slapd*(8), выполнив команду:
   ```
   su root -c /usr/local/libexec/slapd
   ```

   Чтобы проверить, что сервер работает и правильно настроен, по нему можно выполнить поиск с помощью *ldapsearch*(1). По умолчанию, *ldapsearch* установлен как `/usr/local/bin/ldapsearch`:
   ```
   ldapsearch -x -b '' -s base '(objectclass=*)' namingContexts
   ```

   Обратите внимание на заключение параметров команды в одинарные кавычки. Это сделано для предотвращения интерпретации специальных символов оболочкой shell. Итак, мы получим:
   ```
   dn:
   namingContexts: dc=example,dc=com
   ```

   Детали запуска *slapd*(8) можно найти в man-странице *slapd*(8) и в разделе Запуск slapd Этого документа.

10. **Добавьте начальные записи в Ваш каталог**.
    Для добавления записей в Ваш каталог LDAP Вы можете использовать *ldapadd*(1). *ldapadd* работает с данными в формате LDIF. Мы добавим записи в два приёма:
    1. создадим файл LDIF
    2. запустим ldapadd

    С помощью своего любимого текстового редактора создайте файл LDIF следующего содержания:
    ```
    dn: dc=<MY-DOMAIN>,dc=<COM>
    objectclass: dcObject
    objectclass: organization
    o: <MY ORGANIZATION>
    dc: <MY-DOMAIN>

    dn: cn=Manager,dc=<MY-DOMAIN>,dc=<COM>
    objectclass: organizationalRole
    cn: Manager
    ```

    Не забудьте заменить `<MY-DOMAIN>` и `<COM>` на соответствующие компоненты Вашего доменного имени. `<MY ORGANIZATION>` нужно заменить на название Вашей организации. Если вы будете копировать и вставлять, не забудьте обрезать любые начальные и конечные пробелы из данного примера.
    ```
    dn: dc=example,dc=com
    objectclass: dcObject
    objectclass: organization
    o: Example Company
    dc: example

    dn: cn=Manager,dc=example,dc=com
    objectclass: organizationalRole
    ```

```
cn: Manager
```

Теперь можно запустить *ldapadd*(1) для добавления этих записей в Ваш каталог.

```
ldapadd -x -D "cn=Manager,dc=<MY-DOMAIN>,dc=<COM>" -W -f example.ldif
```

Не забудьте заменить `<MY-DOMAIN>` и `<COM>` на соответствующие компоненты Вашего доменного имени. Вам будет предложено ввести пароль, введите "`secret`" (как Вы указали в `slapd.conf`). Например, для `example.com`, выполните:

```
ldapadd -x -D "cn=Manager,dc=example,dc=com" -W -f example.ldif
```

где `example.ldif` - файл, созданный Вами выше.

Дополнительную информацию о создании каталога можно найти в разделе Инструменты создания и обслуживания баз данных этого документа.

11. **Убедитесь, что всё работает**.
Теперь проверим добавленные нами в каталог записи. Чтобы сделать это, можно использовать любой клиент LDAP, но в нашем примере будем использовать утилиту *ldapsearch*(1). Не забудьте заменить `dc=example,dc=com` на корректные для Вашего сайта значения:

```
ldapsearch -x -b 'dc=example,dc=com' '(objectclass=*)'
```

Эта команда найдёт и извлечёт все записи из базы данных.

Теперь Вы можете добавлять другие записи с помощью *ldapadd*(1) или другого клиента LDAP, экспериментировать с различными параметрами настройки, механизмами манипуляции данными, и т.д.

Обратите внимание, что, по умолчанию, к базам данных *slapd*(8) предоставляется *доступ на чтение всем*, за исключением *администратора* (который задан директивой конфигурации `rootdn`). Настоятельно рекомендуется установить контроль доступа и разрешать доступ только авторизованным пользователям. Контроль доступа обсуждается в соответствующем разделе. Также будет полезно прочесть разделы Вопросы безопасности, Использование SASL и Использование TLS.

В следующих разделах представлена более детальная информация по сборке, установке и запуску *slapd*(8).

К содержанию

3. Общая картина - варианты конфигурации

В этом разделе даётся краткий обзор возможных вариантов конфигурации каталога LDAP, а также того, как Ваш Автономный демон LDAP *slapd*(8) может вписаться в остальной мир.

3.1. Локальная служба каталогов

В таком варианте конфигурации Вы запускаете *slapd*(8), который будет предоставлять службу каталога только Вашему локальному домену. Он никогда не будет взаимодействовать с другими серверами службы каталогов. Этот вариант конфигурации показан на рисунке 3.1.

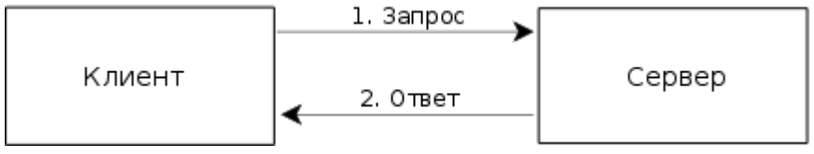

Рисунок 3.1: Локальная служба каталогов.

Данная конфигурация применима, если Вы только начинаете знакомиться со службами каталогов (это как раз то, что у Вас получится после воплощения в жизнь руководства по быстрому развёртыванию и запуску), или если Вам нужна только локальная служба каталогов, а связь с остальным миром Вас не интересует. В любом

случае, довольно просто развернуть на её базе какую-нибудь более весомую конфигурацию, если Вам потом этого захочется.

3.2. Локальная служба каталогов с отсылками

В таком варианте конфигурации Вы запускаете *slapd*(8), который будет предоставлять службу каталога Вашему локальному домену, и настраиваете его так, чтобы он мог возвращать отсылки на другие серверы, способные обрабатывать запросы. Вы можете сами запустить такую службу (или службы), либо воспользоваться какой-нибудь сторонней. Этот вариант конфигурации показан на рисунке 3.2.

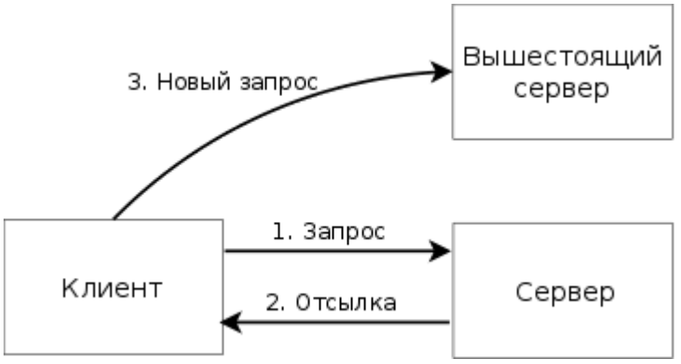

Рисунок 3.2: Локальная служба каталогов с отсылками

Данная конфигурация применима, если Вы хотите организовать локальную службу каталога и принимать участие в Глобальном Каталоге, либо Вы хотите делегировать ответственность за обслуживание *нижестоящих* записей другому серверу.

3.3. Реплицируемая служба каталогов

slapd(8) поддерживает репликацию, основанную на *LDAP Sync*, называемую *syncrepl*, которая может использоваться для поддержания теневых копий информации каталога на нескольких серверах службы каталога. В наиболее общем варианте такой конфигурации присутствует *основной* сервер-поставщик syncrepl и один или несколько *подчинённых* (или *теневых*) серверов-потребителей syncrepl. Пример конфигурации с основным и подчинённым серверами показан на рисунке 3.3. Возможна также конфигурация с несколькими главными серверами.

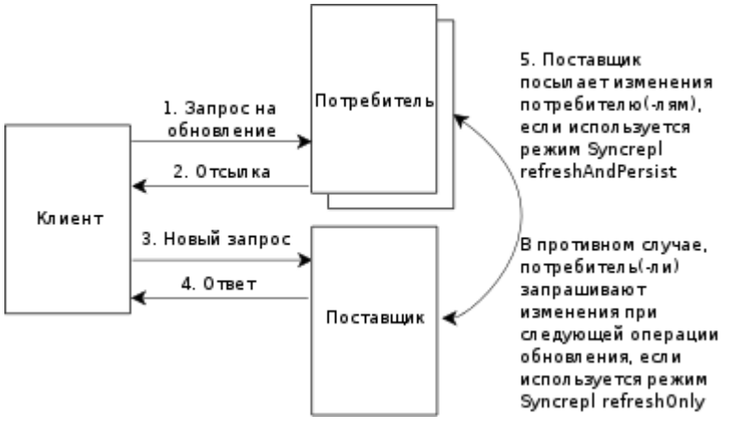

Рисунок 3.3: Реплицируемая служба каталогов

Эта конфигурация может быть использована в сочетании с любой из двух предыдущих в тех ситуациях, когда одиночный сервер slapd (8) не может обеспечить требуемой надежности или доступности.

3.4. Распределённая локальная служба каталогов

В таком варианте конфигурации локальная служба каталогов поделена на более мелкие службы, каждая из которых может быть реплицирована, и эти службы *связаны* между собой с помощью *вышестоящих* и *нижестоящих* отсылок.

К содержанию

4. Сборка и установка программного обеспечения OpenLDAP

В этом разделе подробно рассказывается о том, как собрать и установить пакет программного обеспечения OpenLDAP, включая *slapd*(8), Автономный демон LDAP. Сборка и установка ПО OpenLDAP проходит в несколько этапов: установка ПО, от которого зависит OpenLDAP, конфигурирование самого ПО OpenLDAP, сборка, и, наконец, установка. В следующих подразделах этот процесс описан подробно.

4.1. Получение и распаковка программного обеспечения

Вы можете получить программное обеспечение OpenLDAP со страницы закачек проекта по адресу http://www.openldap.org/software/download/ или прямо с сервиса FTP проекта по адресу ftp://ftp.openldap.org/pub/OpenLDAP/.

Для *общего использования* проект предоставляет два вида пакетов. Проект выпускает *релизы (releases)*, в которых доступны новые функции и исправления ошибок. Хотя проект заботится о том, чтобы повысить стабильность этих релизов, проблемы, как обычно, обнаруживаются только после выпуска *релиза*. *Стабильным (stable)* релизом считается последний *релиз*, который продемонстрировал стабильность во время общего использования.

Пользователи программного обеспечения OpenLDAP могут выбирать по своему желанию между *новыми возможностями* и *продемонстрированной стабильностью*, и устанавливать наиболее подходящий им вид пакета.

После скачивания программного обеспечения OpenLDAP, Вам нужно распаковать дистрибутив из архива и сменить рабочую директорию на корневую директорию дистрибутива:

```
gunzip -c openldap-VERSION.tgz | tar xf -
cd openldap-VERSION
```

Замените VERSION на версию Вашего дистрибутива.

Просмотрите идущие с дистрибутивом документы COPYRIGHT, LICENSE, README и INSTALL. Документы COPYRIGHT и LICENSE дают информацию о разрешениях на использование и копирование, а также об ограниченной гарантии пакета OpenLDAP. В документах README и INSTALL представлена детальная информация о требованиях, предъявляемых пакетом OpenLDAP к установленному на сервере программному обеспечению, и о процедуре установки.

4.2. Программное обеспечение, от которого зависит OpenLDAP

Программное обеспечение OpenLDAP использует для своей работы ряд пакетов, предоставляемых третьими лицами. В зависимости от возможностей, которые Вы собираетесь использовать, Вам может потребоваться скачать и установить дополнительные пакеты программного обеспечения. В этом подразделе даются некоторые детали о ПО третьих лиц, которое обычно требуется установить. Однако, чтобы получить актуальную информацию о зависимостях, прочтите документ README. Обратите внимание, что некоторые из перечисленных здесь пакетов ПО третьих лиц могут, в свою очередь, зависеть от дополнительных пакетов программного

обеспечения. Устанавливайте каждый пакет согласно распространяемых с ним инструкций по установке.

4.2.1. Transport Layer Security

Клиенты и серверы OpenLDAP требуют установки библиотек TLS OpenSSL, GnuTLS или MozNSS для предоставления сервисов Transport Layer Security. Хотя некоторые операционные системы могут предоставлять эти библиотеки как часть основной системы или как дополнительный программный компонент, OpenSSL, GnuTLS, и Mozilla NSS часто требуют отдельной установки.

OpenSSL можно найти на http://www.openssl.org/. GnuTLS можно найти на http://www.gnu.org/software/gnutls/. Mozilla NSS можно найти на http://developer.mozilla.org/en/NSS.

Программное обеспечение OpenLDAP не будет полностью совместимо с LDAPv3, если его скрипт `configure` не определит доступность какой-либо библиотеки TLS.

4.2.2. Simple Authentication and Security Layer

Клиенты и серверы OpenLDAP требуют установки библиотеки Cyrus SASL для предоставления сервисов Simple Authentication and Security Layer. Хотя некоторые операционные системы могут предоставлять эту библиотеку как часть основной системы или как дополнительный программный компонент, Cyrus SASL часто требует отдельной установки.

Cyrus SASL можно найти на http://asg.web.cmu.edu/sasl/sasl-library.html. Cyrus SASL будет использовать библиотеки OpenSSL и Kerberos/GSSAPI, если они были предустановлены.

Программное обеспечение OpenLDAP не будет полностью совместимо с LDAPv3, если его скрипт `configure` не определит доступность установленной Cyrus SASL.

4.2.3. Сервис аутентификации Kerberos

Клиенты и серверы OpenLDAP поддерживают сервисы аутентификации Kerberos. В частности, OpenLDAP поддерживает механизм аутентификации SASL Kerberos V GSS-API, известный как механизм GSSAPI. Чтобы это работало, кроме библиотек Cyrus SASL, требуются библиотеки либо Heimdal, либо MIT Kerberos V.

Heimdal Kerberos можно найти на http://www.pdc.kth.se/heimdal/. MIT Kerberos можно найти на http://web.mit.edu/kerberos/www/.

Настоятельно рекомендуется использовать сервисы строгой аутентификации, такие, как предоставляет Kerberos.

4.2.4. Программное обеспечение баз данных

Основные механизмы манипуляции данными *slapd*(8) BDB и HDB требуют Oracle Corporation Berkeley DB. Если это ПО недоступно во время выполнения `configure`, Вы не сможете собрать *slapd*(8) с поддержкой этих основных механизмов манипуляции данными.

Ваша операционная система может предоставлять поддерживаемую версию Berkeley DB в основной системе или как дополнительный программный компонент. В противном случае, Вам нужно получить и установить её самостоятельно.

Berkeley DB можно найти на странице закачек Oracle Corporation Berkeley DB по адресу http://www.oracle.com/technology/software/products/berkeley-db/index.html.

Там будет доступно несколько версий. Обычно рекомендуется самый последний релиз (с опубликованными патчами). Данный пакет необходим, если Вы собираетесь использовать механизмы манипуляции данными BDB или HDB.

Примечание: Посмотрите также Рекомендуемые версии зависимостей программного обеспечения OpenLDAP.

4.2.5. Потоки

OpenLDAP спроектирован так, чтобы использовать преимущества, предоставляемые потоками. OpenLDAP поддерживает POSIX *pthreads*, Mach *CThreads*, и некоторые другие. Скрипт `configure` пожалуется, если не сможет найти подходящей подсистемы потоков. Если это случилось, посмотрите, пожалуйста, раздел `Software|Installation|Platform Hints` OpenLDAP FAQ http://www.openldap.org/faq/.

4.2.6. TCP Wrappers

slapd(8) поддерживает TCP Wrappers (фильтры контроля доступа уровня IP), если они предустановлены. Рекомендуется использование TCP Wrappers или других фильтров доступа уровня IP (например таких, которые предоставляются IP-фаерволами) на серверах, содержащих информацию ограниченного распространения.

4.3. Запуск configure

Пришло время запустить скрипт `configure` с опцией `--help`. В результате Вы получите список опций, которые можно поменять при сборке OpenLDAP. С помощью этого метода можно включить или отключить многие функции OpenLDAP.

```
./configure --help
```

Скрипт `configure` также обращает внимание на некоторые переменные, задаваемые как в командной строке, так и в окружении. Эти переменные включают в себя:

Таблица 4.1: Переменные

Переменная	**Описание**
`CC`	Указывает альтернативный компилятор C
`CFLAGS`	Указывает дополнительные флаги компилятора
`CPPFLAGS`	Указывает флаги препроцессора C
`LDFLAGS`	Указывает флаги компоновщика
`LIBS`	Указывает дополнительные библиотеки

Теперь запустите скрипт `configure` с любыми желаемыми опциями конфигурации или переменными.

```
./configure [опции] [переменная=значение ...]
```

В качестве примера, предположим, что мы хотим установить OpenLDAP с механизмом манипуляции данными BDB и поддержкой TCP Wrappers. По умолчанию, поддержка BDB включена, а TCP Wrappers - нет. Поэтому нам нужно указать только `--enable-wrappers` , чтобы добавить поддержку TCP Wrappers:

```
./configure --enable-wrappers
```

Однако, выполнение данной команды закончится неудачей, если программное обеспечение, от которого зависит OpenLDAP, не установлено в системных директориях. Например, если заголовочные файлы и библиотеки TCP Wrappers установлены соответственно в `/usr/local/include` и `/usr/local/lib`, скрипт `configure` обычно следует запускать таким образом:

```
./configure --enable-wrappers \
        CPPFLAGS="-I/usr/local/include" \
        LDFLAGS="-L/usr/local/lib -Wl,-rpath,/usr/local/lib"
```

Чаще всего, скрипт `configure` сам определяет соответствующие настройки. Если на этом этапе Вы столкнулись с трудностями, проконсультируйтесь с документацией по Вашей платформе и проверьте Ваши опции `configure`, если Вы их устанавливали.

4.4. Сборка программного обеспечения

При удачном окончании работы скрипта `configure`, последней строкой вывода будет:

```
Please "make depend" to build dependencies
```

Если последняя строка отличается, значит выполнение `configure` окончилось неудачей, и Вам нужно просмотреть вывод скрипта, чтобы определить, что пошло не так. Пока `configure` не завершится удачно, продолжать дальше не следует.

Чтобы построить зависимости, запустите:

```
make depend
```

Теперь соберём программное обеспечение. На этом этапе происходит фактическая компиляция OpenLDAP.

```
make
```

Вам нужно внимательно изучить вывод этой команды, чтобы убедиться, что всё собрано правильно. Обратите внимание, что эта команда, кроме самого *slapd*(8), собирает также библиотеки LDAP и соответствующее клиентское программное обеспечение.

4.5. Тестирование программного обеспечения

Когда программное обеспечение было правильно сконфигурировано и успешно собрано, нужно выполнить набор тестов для проверки сборки.

```
make test
```

Будут запущены тесты, применимые к Вашей конфигурации, и они должны успешно пройти. Некоторые тесты, такие, как тест репликации, могут быть пропущены, если они не поддерживаются Вашей конфигурацией.

4.6. Установка программного обеспечения

После того, как Вы успешно протестировали программное обеспечение, пора его установить. Для этого Вам понадобятся права на запись в директории установки, которые Вы указали при запуске `configure`. По умолчанию, программное обеспечение OpenLDAP устанавливается в `/usr/local`. Если Вы поменяли эту настройку опцией `--prefix` при запуске `configure`, ПО будет установлено в то место, которое Вы указали.

Обычно, установка требует прав *администратора*. Находясь в корне каталога с исходным кодом OpenLDAP, выполните:

```
su root -c 'make install'
```

и введите соответствующий пароль при запросе.

Вам нужно внимательно изучить вывод этой команды, чтобы убедиться, что всё установлено правильно. По умолчанию, конфигурационные файлы *slapd*(8) находятся в `/usr/local/etc/openldap`. Дополнительную информацию смотрите в разделе Настройка slapd.

К содержанию

5. Настройка slapd

После того как программное обеспечение было собрано и установлено, можно приступать к настройке *slapd*(8) для работы Вашего каталога.

В OpenLDAP 2.3 и более новых версиях осуществлён переход к использованию механизма динамической конфигурации времени исполнения, *slapd-config*(5). *slapd-config*(5):

- полностью поддерживает LDAP;
- управляется с помощью стандартных операций LDAP;
- хранит свои конфигурационные данные в базе данных LDIF, обычно в директории `/usr/local/etc/openldap/slapd.d`;
- позволяет менять все настройки slapd на лету, обычно не требуя перезагрузки демона для вступления изменений в силу.

В этом разделе описывается общий формат системы конфигурации *slapd-config*(5), а затем дается детальное описание часто используемых конфигурационных настроек.

Конфигурационный файл в старом стиле *slapd.conf*(5) до сих пор поддерживается, но его использование не рекомендуется. В будущих версиях OpenLDAP его поддержка будет прекращена. Настройка *slapd*(8) с помощью *slapd.conf*(5) описана в следующем разделе.

За информацией о том, как с помощью slapd автоматически преобразовать файл *slapd.conf*(5) в формат *slapd-config*(5), обратитесь к man-странице *slapd*(8).

Примечание: Несмотря на то, что система *slapd-config*(5) хранит свою конфигурацию в LDIF (текстовых) файлах, Вам *ни в коем случае* не следует напрямую редактировать какие-либо LDIF-файлы. Изменения конфигурации должны выполняться с помощью операций LDAP, таких как *ldapadd*(1), *ldapdelete*(1) или *ldapmodify*(1).

Примечание: Необходимо продолжать использование старой системы конфигурации *slapd.conf*(5) в том случае, когда для работы Вашего каталога OpenLDAP требуется один или несколько механизмов манипуляции данными или наложений, которые еще не адаптированы для использования с системой *slapd-config*(5). В OpenLDAP версии 2.4.33 адаптированы все официальные механизмы манипуляции данными. Однако могут быть неадаптированы некоторые дополнительно распространяемые или экспериментальные наложения.

5.1. Макет конфигурации

Конфигурация slapd хранится в специальном каталоге LDAP с предопределенными схемой и DIT. Для хранения глобальных конфигурационных опций, определений схем, механизмов манипуляции данными, баз данных и различных других значений используются специфические объектные классы. Примерное дерево конфигурации показано на рисунке 5.1.

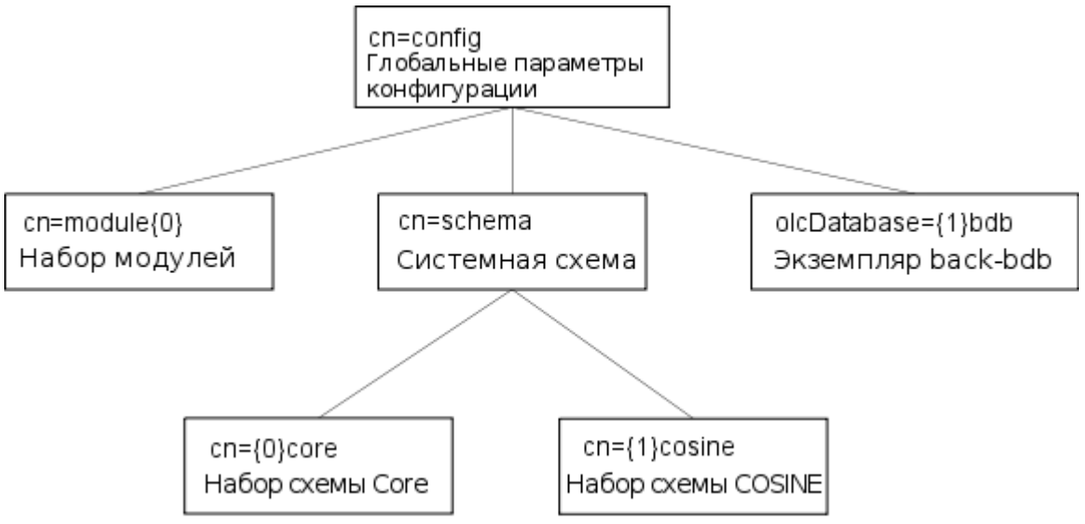

Рисунок 5.1: Примерное дерево конфигурации.

В конфигурации могут быть и другие объекты, но для ясности иллюстрации они были исключены.

У дерева конфигурации *slapd-config* очень специфичная структура. Корневая запись называется `cn=config` и содержит глобальные установки конфигурации. Дополнительные установки содержатся в отдельных дочерних записях:

- Модули, загружаемые динамически
  Возможно использование, только если при сборке программного обеспечения была указана опция `--enable-modules`.
- Определения схемы
  Запись `cn=schema,cn=config` содержит описания системной схемы (тех наборов, которые были скомпилированы в slapd).
  Дочерние записи `cn=schema,cn=config` содержат пользовательские наборы схемы, которые загружаются из конфигурационных файлов или добавляются на лету.
- Настройки конкретных механизмов манипуляции данными
- Настройки конкретных баз данных
  Настройки наложений определяются в дочерних записях записи настроек базы данных.
  Записи настроек баз данных и наложений могут также иметь различные другие дочерние записи.

К конфигурационной информации применимы обычные правила для LDIF-файлов. Строки комментариев, начинающиеся с символа '`#`', игнорируются. Если строка начинается с единичного пробела, то она считается продолжением предыдущей строки (даже если предыдущие строки является комментарием) и этот пробел будет удален. Записи разделяются пустыми строками.

Основные разделы конфигурационного LDIF выглядят следующим образом:

```
# глобальные настройки конфигурации
dn: cn=config
objectClass: olcGlobal
cn: config
<глобальные настройки>

# определения схемы
dn: cn=schema,cn=config
objectClass: olcSchemaConfig
cn: schema
<системная схема>

dn: cn={X}core,cn=schema,cn=config
objectClass: olcSchemaConfig
cn: {X}core
<набор схемы core>

# дополнительные определённые пользователем наборы схемы
...

# определения механизма манипуляции данными
dn: olcBackend=<typeA>,cn=config
objectClass: olcBackendConfig
olcBackend: <typeA>
<настройки, специфичные для механизма манипуляции данными>

# определения базы данных
dn: olcDatabase={X}<typeA>,cn=config
objectClass: olcDatabaseConfig
olcDatabase: {X}<typeA>
<настройки, специфичные для базы данных>

# последующие определения и настройки
...
```

В названиях некоторых из перечисленных выше записей есть числовой индекс "`{X}`". Дело в том, что базы данных LDAP изначально неупорядочены, а большинство настроек конфигурации имеют зависимость от порядка их применения, то есть применение одной настройки возможно только до (или только после) применения другой. Числовые индексы как раз используются для обеспечения последовательного упорядочения в базе данных конфигурации, таким образом соблюдается зависимость от порядка применения настроек. В большинстве случаев не требуется явного указания индекса; он будет сгенерирован автоматически на основании порядка занесения записей.

Директивы конфигурации задаются как значения отдельных атрибутов. Большинство атрибутов и объектных классов, используемых в конфигурации slapd, содержат в своих именах префикс "`olc`" (OpenLDAP Configuration). Практически, поддерживается однозначное соответствие между атрибутами в новом стиле конфигурации и ключевыми словами файла `slapd.conf` в старом стиле: названием атрибута служит ключевое слово файла с

добавлением префикса "olc".

Директивы конфигурации могут иметь аргументы. В таком случае аргументы разделяются пробелами. Если в самом аргументе есть пробел, этот аргумент должен быть заключён в двойные кавычки `"вот так"`. В дальнейшем тексте описания аргументов, которые требуется заменить на актуальный текст, заключены в угловые скобки <>.

В дистрибутиве есть пример конфигурационного файла, который будет устанавливаться в директорию `/usr/local/etc/openldap`. Несколько файлов, содержащих определение схемы (типов атрибутов и объектных классов) также помещаются в директорию `/usr/local/etc/openldap/schema`.

5.2. Директивы конфигурации

В этом подразделе детально описаны часто используемые директивы конфигурации, полный список директив можно найти в man-странице *slapd-config*(5). Конфигурационные директивы в этом подразделе рассматриваются в порядке "сверху-вниз", начиная от глобальных директив в записи `cn=config`. Для каждой директивы дается её описание, значение по умолчанию (если есть), и пример использования.

5.2.1. cn=config

Директивы, содержащиеся в этой записи, как правило, касаются сервера в целом. Большинство из них являются системными или ориентированными на подключение, и не связаны с базами данных. Эта запись должна иметь объектный класс `olcGlobal`.

5.2.1.1. olcIdleTimeout: <целое число>

Определяет количество секунд до того, как принудительно закрыть простаивающее клиентское соединение. При установке в 0 (значение по умолчанию), эта функция отключается.

5.2.1.2. olcLogLevel: <уровень>

Эта директива определяет уровень отладочной информации и статистики работы slapd, которая должна быть журналирована (в настоящее время журналируется в канал `LOG_LOCAL4` *syslogd*(8)). Чтобы это работало (за исключением двух уровней статистики, которые всегда включены), при конфигурации перед сборкой OpenLDAP нужно указать `--enable-debug` (по умолчанию включено). Уровни журналирования могут быть указаны целым числом или ключевым словом. Можно задать несколько уровней журналирования, либо указать несколько уровней как сумму соответствующих целых чисел. Чтобы узнать соответствие числового значения типу отладочной информации, запустите slapd с ключом `-d?` или посмотрите таблицу ниже. Возможные значения <уровня>:

Таблица 5.1: Уровни отладки

Уровень	Ключевое слово	Описание
-1	any	включить всю отладочную информацию
0		отключить отладку
1	(0x1 trace)	отслеживать вызовы функций
2	(0x2 packets)	отладка обработки пакетов
4	(0x4 args)	усиленная отладка
8	(0x8 conns)	управление соединениями
16	(0x10 BER)	вывод посылки и приёма пакетов
32	(0x20 filter)	обработка поисковых фильтров
64	(0x40 config)	обработка конфигурации
128	(0x80 ACL)	обработка списков контроля доступа
256	(0x100 stats)	статистика соединений/операций/результатов

512	(0x200 stats2)	статистика посылки записей журнала
1024	(0x400 shell)	вывод взаимодействия с механизмами манипуляции данными shell
2048	(0x800 parse)	вывод отладочной информации разбора записей
16384	(0x4000 sync)	обслуживание потребителей syncrepl
32768	(0x8000 none)	только сообщения, выводимые независимо от заданного уровня журналирования (то есть высокоприоритетные)

Требуемый уровень журналирования может быть задан одним целым числом (десятичным или шестнадцатеричным), которое является комбинацией нужных уровней журналирования (соединяемых через логическое ИЛИ). Также он может быть задан списком целых чисел (опять же соединяемых через логическое ИЛИ), либо списком ключевых слов, указанных в скобках в соответствующем столбце таблицы выше. Таким образом, директивы

```
olcLogLevel 129
olcLogLevel 0x81
olcLogLevel 128 1
olcLogLevel 0x80 0x1
olcLogLevel acl trace
```

эквивалентны.

Примеры:

```
olcLogLevel -1
```

Это вызывает журналирование большого количества отладочной информации.

```
olcLogLevel conns filter
```

Журналирование только информации о соединениях и обработке поисковых фильтров.

```
olcLogLevel none
```

Журналирование только тех сообщений, которые выводятся вне зависимости от настроенного уровня журналирования. Это отличается от установки уровня журналирования в 0, когда журналирование отсутствует. Для журналирования высокоприоритетных сообщений требуется как минимум уровень `None`.

Значение по умолчанию:

```
olcLogLevel stats
```

По умолчанию настраивается журналирование основных статистических сведений. Однако, если не определено никакого olcLogLevel, журналирования не происходит (как в уровне 0).

5.2.1.3. olcReferral <URI>

Эта директива определяет отсылку, которая возвращается клиенту когда slapd не может найти подходящую локальную базу данных для обработки запроса.

Пример:

```
olcReferral: ldap://root.openldap.org
```

С помощью этой директивы в проекте OpenLDAP нелокальные запросы перенаправляются на глобальный LDAP-сервер корневого уровня. Продвинутые клиенты LDAP могут сделать повторный запрос на указанный сервер, но имейте в виду, что большинство клиентов способны обрабатывать только простые LDAP URL, состоящие из адреса хоста и, возможно, из уникального имени записи.

5.2.1.4. Пример записи

```
dn: cn=config
objectClass: olcGlobal
cn: config
olcIdleTimeout: 30
```

```
olcLogLevel: Stats
olcReferral: ldap://root.openldap.org
```

5.2.2. cn=module

Если при сборке slapd была включена поддержка динамически загружаемых модулей, можно использовать записи `cn=module` для указания набора модулей, которые требуется загрузить. Записи module должны иметь объектный класс `olcModuleList`.

5.2.2.1. olcModuleLoad: <имя файла>

Указывает имя динамически-загружаемого модуля, который требуется загрузить. Параметр <имя файла> может быть определён либо как абсолютный путь к файлу, либо просто именем файла. Если для файла не задан абсолютный путь, то slapd попытается найти его в директориях, указанных в директиве `olcModulePath`.

5.2.2.2. olcModulePath: <список путей>

Указывает список директорий, в которых будет осуществляться поиск загружаемых модулей. Как правило, пути в списке разделяются двоеточиями, но это зависит от операционной системы.

5.2.2.3. Примеры записей

```
dn: cn=module{0},cn=config
objectClass: olcModuleList
cn: module{0}
olcModuleLoad: /usr/local/lib/smbk5pwd.la

dn: cn=module{1},cn=config
objectClass: olcModuleList
cn: module{1}
olcModulePath: /usr/local/lib:/usr/local/lib/slapd
olcModuleLoad: accesslog.la
olcModuleLoad: pcache.la
```

5.2.3. cn=schema

Запись cn=schema содержит все определения системной схемы данных, которые жёстко вкомпилированы в slapd. В связи с этим, значения этой записи генерируются самим slapd и не требуется указывать никаких значений системной схемы в конфигурационном файле. Тем не менее, эта запись должна быть указана, поскольку она служит базовой записью для добавления пользовательских наборов схемы. Записи schema должны иметь объектный класс `olcSchemaConfig`.

5.2.3.1. olcAttributeTypes: <описание типа атрибута согласно RFC4512>

Эта директива определяет тип атрибута. О том, как использовать эту директиву, читайте в разделе Спецификация схемы.

5.2.3.2. olcObjectClasses: <описание объектного класса согласно RFC4512>

Эта директива определяет объектный класс. О том, как использовать эту директиву, читайте в разделе Спецификация схемы.

5.2.3.3. Примеры записей

```
dn: cn=schema,cn=config
objectClass: olcSchemaConfig
cn: schema

dn: cn=test,cn=schema,cn=config
objectClass: olcSchemaConfig
cn: test
olcAttributeTypes: ( 1.1.1
  NAME 'testAttr'
  EQUALITY integerMatch
```

```
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 )
olcAttributeTypes: ( 1.1.2 NAME 'testTwo' EQUALITY caseIgnoreMatch
  SUBSTR caseIgnoreSubstringsMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.44 )
olcObjectClasses: ( 1.1.3 NAME 'testObject'
  MAY ( testAttr $ testTwo ) AUXILIARY )
```

5.2.4. Директивы, специфичные для механизмов манипуляции данными

Настройки из директив backend применяются ко всем экземплярам баз данных того же типа и, в зависимости от директивы, могут переопределяться директивами конкретной базы данных. Записи backend должны иметь объектный класс `olcBackendConfig`.

5.2.4.1. olcBackend: <тип>

Эта директива обозначает запись, в которой задаётся конфигурация, специфичная для конкретного механизма манипуляции данными. `<Тип>` должен быть одним из поддерживаемых типов механизмов, перечисленных в таблице 5.2.

Таблица 5.2: Механизмы манипуляции данными

Тип	**Описание**
`bdb`	Механизм Berkeley DB с поддержкой транзакций
`config`	Механизм конфигурации slapd
`dnssrv`	Механизм DNS SRV
`hdb`	Иерархический вариант механизма bdb
`ldap`	Механизм Lightweight Directory Access Protocol (прокси)
`ldif`	Механизм Lightweight Data Interchange Format (LDIF)
`meta`	Механизм мета-каталога
`monitor`	Механизм мониторинга
`passwd`	Предоставление доступа только для чтения к *passwd*(5)
`perl`	Программируемый механизм Perl
`shell`	Механизм запуска внешних программ Shell
`sql`	Программируемый механизм SQL

Пример:

```
olcBackend: bdb
```

В этой записи не определено других директив. Предусмотрена возможность указывать дополнительные атрибуты для задания частных свойств какого-либо конкретного механизма манипуляции данными, но до сих пор не было определено таких атрибутов. Поэтому эти директивы, как правило, не появляются в каких-либо фактических конфигурациях.

5.2.4.2. Пример записи

```
dn: olcBackend=bdb,cn=config
objectClass: olcBackendConfig
olcBackend: bdb
```

5.2.5. Директивы, специфичные для базы данных

Директивы в этом подразделе поддерживаются всеми типами баз данных. Записи database должны иметь объектный класс `olcDatabaseConfig`.

5.2.5.1. olcDatabase: [{<индекс>}]<тип>

Эта директива обозначает запись для конкретного экземпляра базы данных. Цифровой {<индекс>} может быть указан для того, чтобы различать несколко баз данных одного и того же типа. Обычно этот индекс можно

опустить, и slapd сгенерирует его автоматически. `<Тип>` должен быть одним из поддерживаемых типов механизмов манипуляции данными, перечисленных в таблице 5.2, или тип `frontend`.

Специальная база данных `frontend` используется для указания настроек уровня базы данных, которые будут применяться ко всем остальным базам данных. Определения в записях конкретных баз данных могут переопределять некоторые настройки базы данных frontend.

Также существует специальная база данных `config`; обе базы данных `config` и `frontend` всегда неявно создаются, даже если они не были сконфигурированы явно, и они создаются перед всеми остальными базами данных.

Пример:

```
olcDatabase: bdb
```

Здесь обозначено начало определения настроек нового экземпляра базы данных BDB.

5.2.5.2. olcAccess: to <what> [by <who> [<accesslevel>] [<control>]]+

Эта директива предоставляет доступ (заданный аргументом <accesslevel>) к набору записей и/или атрибутов (заданный аргументом <what>) одному или нескольким пользователям (заданным аргументом <who>). Смотрите раздел Контроль доступа данного руководства для получения подробных сведений о настройке.

Замечание: Если не задано ни одной директивы `olcAccess`, применяется политика контроля доступа по умолчанию `access to * by * read`, предоставляющая всем пользователям (как авторизованным, так и анонимным) доступ на чтение.

Замечание: Настройки контроля доступа, определённые в базе данных frontend, добавляются к настройкам контроля доступа всех остальных баз данных.

5.2.5.3. olcReadonly { TRUE | FALSE }

Эта директива переводит базу данных в режим "только для чтения". На все попытки внесения в базу данных изменений возвращается ошибка "unwilling to perform" ("не желают выполнять").

Значение по умолчанию:

```
olcReadonly: FALSE
```

5.2.5.4. olcRootDN: <DN>

Эта директива определяет DN, на который не будут накладываться контроль доступа и административные ограничения при выполнении операций с этой базой данных. Данный DN не обязательно должен указывать на запись из этой базы данных, и даже на запись из этого каталога. Данный DN может ссылаться на идентификационную сущность SASL.

Пример с записью:

```
olcRootDN: "cn=Manager,dc=example,dc=com"
```

Пример с SASL:

```
olcRootDN: "uid=root,cn=example.com,cn=digest-md5,cn=auth"
```

Информацию об аутентификационных идентификационных сущностях SASL можно посмотреть в разделе Аутентификация SASL.

5.2.5.5. olcRootPW: <password>

Эта директива может использоваться для назначения пароля для DN из rootdn (когда в качестве rootdn назначен DN из этой базы данных).

Пример:

```
olcRootPW: secret
```

Также можно назначить хэш пароля в форме RFC2307. Для генерации хэша пароля можно использовать *slappasswd*(8).

Пример:

```
olcRootPW: {SSHA}ZKKuqbEKJfKSXhUbHG3fG8MDn9j1v4QN
```

Данный хэш был сгенерирован командой `slappasswd -s secret`.

5.2.5.6. olcSizeLimit: <целое число>

Эта директива определяет максимальное количество записей, возвращаемых поисковым запросом.

Значение по умолчанию:

```
olcSizeLimit: 500
```

Для получения дополнительной информации смотрите раздел Ограничения этого руководства и slapd-config(5).

5.2.5.7. olcSuffix: <dn суффикса>

Эта директива определяет DN суффикса запросов, которые будут направляться в эту базу данных. Возможно определение нескольких суффиксов. Обычно как минимум один суффикс должен быть определён для каждой базы данных. (Некоторые типы механизмов манипуляции данными, такие как `frontend` и `monitor` используют жестко-прописанный суффикс, который невозможно переопределить в конфигурации).

Пример:

```
olcSuffix: "dc=example,dc=com"
```

Запросы с DN, оканчивающимся на "dc=example,dc=com" будут направлены в эту базу данных.

Замечание: Когда выбран механизм манипуляции данными для обработки запроса, slapd ищет строку (строки) определения суффикса в каждом определении базы данных в порядке их указания при конфигурации. Поэтому, если один суффикс базы данных является префиксом к другому, то более общий суффикс должен указываться при конфигурации позже.

5.2.5.8. olcSyncrepl

```
olcSyncrepl: rid=<replica ID>
        provider=ldap[s]://<имя хоста>[:порт]
        [type=refreshOnly|refreshAndPersist]
        [interval=dd:hh:mm:ss]
        [retry=[<интервал между повторами> <количество повторов>]+]
        searchbase=<base DN>
        [filter=<строка фильтра>]
        [scope=sub|one|base]
        [attrs=<список атрибутов>]
        [attrsonly]
        [sizelimit=<limit>]
        [timelimit=<limit>]
        [schemachecking=on|off]
        [bindmethod=simple|sasl]
        [binddn=<DN>]
        [saslmech=<mech>]
        [authcid=<identity>]
        [authzid=<identity>]
        [credentials=<passwd>]
```

```
[realm=<realm>]
[secprops=<properties>]
[starttls=yes|critical]
[tls_cert=<file>]
[tls_key=<file>]
[tls_cacert=<file>]
[tls_cacertdir=<path>]
[tls_reqcert=never|allow|try|demand]
[tls_ciphersuite=<ciphers>]
[tls_crlcheck=none|peer|all]
[logbase=<base DN>]
[logfilter=<filter str>]
[syncdata=default|accesslog|changelog]
```

Эта директива определяет, что текущая база данных будет репликой содержимого каталога основного сервера, и текущий *slapd*(8) будет работать в роли сервера-потребителя репликации путём запуска механизма syncrepl. Основная база данных располагается на сервере-поставщике репликации, указанном в параметре `provider`. База данных реплики синхронизируется с содержимым каталога основного сервера с использованием LDAP Content Synchronization protocol (протокол синхронизации содержимого LDAP). Описание протокола можно найти в RFC4533 (рус.) (RFC4533 (ориг.)).

Параметр `rid` используется для идентификации текущей директивы `syncrepl` в пределах сервера-потребителя репликации. `<ID реплики>` — уникальный идентификатор спецификации syncrepl, описываемой текущей директивой `syncrepl`. `<ID реплики>` должен быть положительным числом длиной не более трёх десятичных цифр.

Параметр `provider` определяет адрес поставщика репликации как LDAP URI. Параметр `provider` указывается схемой (ldap или ldaps), именем хоста и, опционально, номером порта, на котором будет слушать slapd поставщика. В качестве <имени хоста> может быть как доменное имя, так и IP-адрес. Например, `ldap://provider.example.com:389` или `ldaps://192.168.1.1:636`. Если <порт> не задан, используются стандартные номера портов LDAP (389 или 636). Обратите внимание, что механизм syncrepl использует протокол, инициируемый со стороны потребителя, поэтому директивы его настройки располагаются на сервере потребителя, в то время как директивы `replica` располагаются на стороне поставщика. Директивы `syncrepl` и `replica` определяют две независимые части механизма репликации. Они не устанавливают никаких взаимных потоков репликации друг к другу.

Содержимое реплики syncrepl формируется как результат поискового запроса. slapd потребителя посылает поисковые запросы к slapd поставщика согласно данной ему спецификации поискового запроса. Спецификация поискового запроса включает в себя параметры `searchbase`, `scope`, `filter`, `attrs`, `attrsonly`, `sizelimit` и `timelimit`, как и в случае обычного поискового запроса. У параметра `searchbase` нет значения по умолчанию и он должен быть задан обязательно. Значения по умолчанию остальных параметров: `scope` — `sub`, `filter` — `(objectclass=*)`, `attrs` — `"*,+"` (то есть репликация и пользовательских, и операционных атрибутов), `attrsonly` по умолчанию не задан. Параметры `sizelimit` и `timelimit` по умолчанию установлены в "unlimited", и могут задаваться либо положительным целым числом, либо "unlimited".

У протокола LDAP Content Synchronization два типа операций: `refreshOnly` и `refreshAndPersist`. Тип операций задаётся параметром `type`. При типе операций `refreshOnly` время выполнения следующей операции синхронизационного поиска периодически переназначается на заданный временной интервал после окончания предыдущей операции синхронизации. Интервал указывается параметром `interval`, по умолчанию заданным в 1 день. При типе операций `refreshAndPersist` синхронизационный поиск постоянно выполняется в экземпляре *slapd* поставщика. При дальнейшем обновлении каталога поставщика будет генерироваться `searchResultEntry` и отправляться slapd потребителя как ответ на постоянный синхронизационный поиск на стороне поставщика.

При возникновении ошибок в процессе репликации, потребитель будет пытаться сделать повторное подключение в соответствии с указанным в параметре retry списком пар значений <интервал между повторами> и <количество повторов>. Например, retry="60 10 300 3" позволяет получателю повторять запрос через каждые 60 секунд первые 10 раз, а затем через каждые 300 секунд еще 3 раза перед тем, как прекратить попытки. Если в <количестве повторов> указан +, то повторные попытки будут осуществляться до получения положительного результата (неограниченное количество раз).

С помощью параметра `schemachecking` можно включить принудительную проверку получаемых данных на соответствие схеме данных на стороне потребителя LDAP Sync. Если он установлен в on, каждая реплицируемая запись будет проверяться, не нарушит ли она целостности схемы данных при помещении её в базу данных реплики. Каждая запись в реплике должна содержать те атрибуты, которые помечены как обязательные в определении схемы. Если этот параметр установлен в off, записи будут сохраняться без

проверки на соответствие схеме. Значение по умолчанию — off.

Параметр `binddn` указывает DN, под которым будет происходить соединение с slapd поставщика для выполнения поисковых запросов syncrepl. Этот DN должен иметь права на чтение к реплицируемому содержимому каталога на главном сервере.

Параметр `bindmethod` может быть либо `simple`, либо `sasl`, в зависимости от метода аутентификации (простая парольная или SASL), используемого при соединении с *slapd* поставщика.

Лучше не использовать простую аутентификацию, если не предприняты адекватные меры защиты целостности и конфиденциальности данных (такие как TLS или IPsec). При использовании простой аутентификации требуется задать параметры `binddn` и `credentials`.

В общем случае рекомендуется использование аутентификации SASL. Аутентификация SASL требует указания механизма с использованием параметра `saslmech`. В зависимости от механизма, можно задать аутентификационную идентификационную сущность и/или учётные данные с помощью соответствующих параметров `authcid` и `credentials`. Параметр `authzid` может быть использован для определения авторизационной идентификационной сущности.

Параметр `realm` указывает realm, с которым некоторые механизмы проводят аутентификацию идентификационной сущности. В параметре `secprops` указываются настройки безопасности Cyrus SASL.

Параметр `starttls` указывает использование расширенной операции StartTLS для установки сессии TLS до начала аутентификации при подключении к серверу-поставщику. Если установлен аргумент `critical`, сессия будет прервана в случае неудачного завершения запроса StartTLS. Если этот аргумент не был установлен, сессия syncrepl будет продолжена без TLS. Обратите внимание, что основные настройки TLS для slapd не используются механизмом syncrepl; по умолчанию будут использоваться параметры TLS из конфигурационного файла *ldap.conf*(5). Если же указать настройки TLS в данной секции конфигурационного файла *slapd.conf*(5), то любые настройки из *ldap.conf*(5) будут полностью проигнорированы.

Вместо того, чтобы реплицировать записи целиком, потребитель может запрашивать журналы изменений данных. Этот режим работы называется *дельта-syncrepl*. В дополнение к вышеперечисленным параметрам, нужно установить параметры `logbase` и `logfilter` в соответствии с типом журнала, который будет использоваться. Параметр `syncdata` должен быть задан либо в `"accesslog"` если журнал соответствует формату *slapo-accesslog*(5), либо в `"changelog"` если журнал соответствует устаревшему формату *changelog*. Если параметр `syncdata` пропущен или установлен в `"default"`, то параметры журналирования игнорируются.

Механизм репликации *syncrepl* поддерживается механизмами манипуляции данными *bdb* и *hdb*.

Дополнительная информация по использованию этой директивы в разделе Репликация на основе LDAP Sync этого руководства.

5.2.5.9. olcTimeLimit: <целое число>

Эта директива указывает максимальное количество секунд, которое slapd может потратить, чтобы ответить на поисковый запрос. Если запрос не был обработан за это время, клиенту возвращается уведомление о превышении времени обработки.

Значение по умолчанию:

```
olcTimeLimit: 3600
```

Для получения дополнительной информации смотрите раздел Ограничения этого руководства и slapd-config(5).

5.2.5.10. olcUpdateref: <URL>

Эта директива применяется только в случае работы slapd в режиме подчинённого сервера. Она указывает URL, возвращаемый клиентам при попытке выполнить на реплике запрос на обновление. Если директива указывается несколько раз, клиенту возвращаются все URL.

Пример:

```
olcUpdateref:   ldap://master.example.net
```

5.2.5.11. Примеры записей

```
dn: olcDatabase=frontend,cn=config
objectClass: olcDatabaseConfig
objectClass: olcFrontendConfig
olcDatabase: frontend
olcReadOnly: FALSE

dn: olcDatabase=config,cn=config
objectClass: olcDatabaseConfig
olcDatabase: config
olcRootDN: cn=Manager,dc=example,dc=com
```

5.2.6. Директивы баз данных BDB и HDB

Директивы этой категории применимы как к базам данных BDB, так и HDB. Они используются в записи olcDatabase в дополнение к общим директивам баз данных, описанным выше. Полное руководство по конфигурационным директивам BDB/HDB смотрите в *slapd-bdb*(5). В дополнение к объектному классу `olcDatabaseConfig`, записи баз данных BDB и HDB должны иметь объектные классы `olcBdbConfig` и `olcHdbConfig` соответственно.

5.2.6.1. olcDbDirectory: <директория>

Эта директива указывает директорию, где будут находиться файлы BDB, содержащие базу данных, и соответствующие им индексы.

Значение по умолчанию:

```
olcDbDirectory: /usr/local/var/openldap-data
```

5.2.6.2. olcDbCachesize: <целое число>

Эта директива указывает размер (в количестве записей) кэша в оперативной памяти, создаваемого экземпляром базы данных механизма манипуляции данными BDB.

Значение по умолчанию:

```
olcDbCachesize: 1000
```

5.2.6.3. olcDbCheckpoint: <Кбайты> <минуты>

Эта директива устанавливает контрольные точки для сброса журнала транзакций BDB. При срабатывании контрольной точки буферы базы данных записываются на диск и в журнал помещается запись о прохождении контрольной точки. Срабатывание контрольной точки наступает либо когда определённое количество <Кбайт> было записано, либо когда с момента последней контрольной точки прошло определённое количество <минут>. По умолчанию оба аргумента установлены в ноль, в этом случае они игнорируются. Когда аргумент <минуты> не нулевой, периодический процесс будет запускаться через указанное количество <минут> для выполнения контрольной точки. Дополнительные детали можно найти в документации Berkeley DB.

Пример:

```
olcDbCheckpoint: 1024 10
```

5.2.6.4. olcDbConfig: <настройки DB_CONFIG>

Этот атрибут указывает директиву конфигурации, которая будет помещена в файл `DB_CONFIG` в директории, где находятся файлы базы данных. Если во время запуска сервера такого файла не существует, он будет создан, и в него будут записаны настройки из этого атрибута. Если файл существует, его содержимое будет прочитано и отображено в этом атрибуте. Атрибут может иметь несколько значений для размещения

нескольких директив конфигурации. Значения по умолчанию не определены, но, чтобы добиться хорошей производительности сервера, в этом атрибуте важно использовать правильные настройки.

Любые изменения данного атрибута будут записаны в файл `DB_CONFIG` и заставят окружение базы данных перечитать свои настройки, поэтому изменения будут применены сразу. Если кэш окружения базы данных большой и давно не сбрасывался при прохождении контрольной точки, данная операция перечитывания настроек может занять продолжительное время. В таком случае может быть целесообразным выполнить единичный сброс в контрольной точке вручную, используя утилиту Berkeley DB *db_checkpoint*, перед тем, как использовать операцию LDAP Modify для изменения этого атрибута.

Пример:

```
olcDbConfig: set_cachesize 0 10485760 0
olcDbConfig: set_lg_bsize 2097512
olcDbConfig: set_lg_dir /var/tmp/bdb-log
olcDbConfig: set_flags DB_LOG_AUTOREMOVE
```

В этом примере устанавливается размер кэша BDB в 10Мб, размер буфера журнала транзакций BDB в 2Мб, и директория /var/tmp/bdb-log указывается как место хранения файлов журнала транзакций. Также устанавливается флаг, требующий, чтобы BDB удалял файлы журнала транзакций по мере сброса их содержимого во время прохождения контрольных точек, когда информация в этих файлах уже не нужна. Без этой настройки файлы журнала транзакций будут складироваться до тех пор, пока не будут удалены какой-нибудь другой процедурой очистки. Подробнее об этом смотрите в описании команды `db_archive` в документации Berkeley DB. Полный список флагов Berkeley DB можно найти на странице http://www.oracle.com/technology/documentation/berkeley-db/db/api_c/env_set_flags.html.

В идеале размер кэша BDB должен быть настолько велик, чтобы в неё как минимум помещался рабочий набор данных базы данных, размер буфера журнала должен вмещать большое количество транзакций без переполнения, а директория хранения журнала должна помещаться на отдельном от основных файлов базы данных физическом диске. Обе эти директории (хранящие базу данных и журналы) не должны также размещаться на диске, используемом для регулярной работы системы, то есть на том, где размещены корневая, загрузочная файловые системы и область подкачки. Дополнительные детали можно найти в FAQ-o-Matic и документации Berkeley DB.

5.2.6.5. olcDbNosync: { TRUE | FALSE }

Эта опция отменяет немедленную синхронизацию между данными в оперативной памяти и содержимым базы данных на диске при внесении изменений. Установка этой опции в `TRUE` может повысить производительность за счёт возможной потери целостности данных при крахе системы. Эта директива имеет тот же эффект, что и использование

```
olcDbConfig: set_flags DB_TXN_NOSYNC
```

5.2.6.6. olcDbIDLcacheSize: <целое число>

Указывает размер (в слотах индексов) кэша индексов в оперативной памяти. По умолчанию равен нулю. Большее значение увеличит скорость выполнения частых запросов на получение индексированных записей. Оптимальный размер зависит от данных и поисковых характеристик базы данных, однако хорошей отправной точкой будет использование кэша индексов, в 3 раза превышающего кэш записей.

Пример:

```
olcDbIDLcacheSize: 3000
```

5.2.6.7. olcDbIndex: {<список атрибутов> | default} [pres,eq,approx,sub,none]

Эта директива определяет индексирование, которое будет применяться к указанным атрибутам. Если задан только `<список атрибутов>`, к этим атрибутам будет применяться индексирование по умолчанию. Ключевые слова, с помощью которых задаются индексы, коррелируют с обычными типами соответствия, которые могут быть использованы в поисковых фильтрах LDAP.

Примеры:

```
olcDbIndex: default pres,eq
olcDbIndex: uid
olcDbIndex: cn,sn pres,eq,sub
olcDbIndex: objectClass eq
```

В первой строке задаётся набор индексов по умолчанию, включающий индексы наличия (pres) и равенства (eq). Во второй строке эти индексы по умолчанию (pres,eq) устанавливаются для типа атрибута `uid`. В третьей строке для типов атрибутов `cn` и `sn` устанавливаются индексы наличия, равенства и равенства подстроке (sub). В четвёртой строке устанавливается индекс равенства для типа атрибута `objectClass`.

Не существует ключевого слова индекса для соответствий типа "неравенство". Как правило, такие соответствия не используют индексы. Однако, некоторые атрибуты поддерживают индексирование для соответствий типа "неравенство", основанное на индексе равенства.

Индекс равенства подстроке может быть задан более конкретно как `subinitial`, `subany` или `subfinal`, коррелируя с тремя возможными компонентами фильтра соответствия равенству подстроке. Индекс subinitial индексирует только подстроки, с которых начинается значение атрибута. Индекс subfinal индексирует только подстроки, которыми закачивается значение атрибута, а индекс subany индексирует подстроки, которые могут встретиться в любом месте значения атрибута.

Обратите внимание, что, по умолчанию, установка индекса на какой-либо атрибут также влияет на каждый подтип этого атрибута. Например, установка индекса равенства на атрибут `name` создаст аналогичное индексирование на `cn`, `sn` и все остальные атрибуты, унаследованные от `name`.

По умолчанию индексирование не применяется. В общем случае, важно установить хотя бы индекс равенства на тип атрибута objectClass.

```
olcDbindex: objectClass eq
```

Дополнительные индексы должны устанавливаться в соответствии с наиболее часто выполняемыми поисковыми запросами к этой базе данных. Индекс наличия стоит устанавливать на атрибут только в том случае, если этот атрибут встречается в базе данных очень редко и поисковые запросы на наличие этого атрибута выполняются очень часто во время нормальной работы каталога. Большинство приложений не используют поисковые запросы на наличие атрибутов, поэтому обычно установка индекса наличия не приносит большой пользы.

Если установки этой директивы изменились во время работы slapd, будет запущено внутреннее задание для создания измененных данных индекса. В то время, как процесс индексирования выполняет свою задачу, все серверные операции могут продолжаться в обычном режиме. Если slapd будет остановлен до того, как процесс индексирования завершится, индексирование должно быть завершено вручную с помощью утилиты slapindex.

5.2.6.8. olcDbLinearIndex: { TRUE | FALSE }

Если данная директива установлена в `TRUE`, slapindex будет индексировать по одному атрибуту за раз, а если в `FALSE` (значение по умолчанию), то все проиндексированные атрибуты записи будут обрабатываться одновременно. Поскольку при установке в `TRUE` каждый индексированный атрибут обрабатывается индивидуально, приходится несколько раз проходить по всей базе данных. Эта опция повышает производительность slapindex, если размер базы данных превышает размер кэша BDB. Если размер кэша BDB достаточно велик, использование этой опции не требуется и может понизить производительность. Также, по умолчанию, slapadd выполняет полное индексирование, поэтому отдельно запускать slapindex не требуется. Если эта опция установлена в `TRUE`, slapadd не выполняет индексирования, и нужно обязательно запускать slapindex.

5.2.6.9. olcDbMode: { <восьмеричное> | <символическое представление> }

Эта директива указывает режим защиты файлов, который будет задан для вновь создаваемых файлов индексов базы данных. Режим может быть указан либо в форме `0600`, либо в форме `-rw-------`

Значение по умолчанию:

```
olcDbMode: 0600
```

5.2.6.10. olcDbSearchStack: <целое число>

Указывает глубину стека, используемого для оценки поискового фильтра. Оценка поисковых фильтров происходит с помощью стека, в котором размещаются вложенные условия AND / OR. Для каждого серверного потока выделяется собственный стек. Глубина стека определяет, насколько комплексный фильтр может быть оценен без необходимости выделения какой-либо дополнительной памяти. Для операции поиска с фильтром, вложенность которого глубже, чем глубина поискового стека, потребуется отдельный стек, который будет выделен для выполнения этой конкретной операции. Такие отдельные выделения памяти могут оказать существенное негативное влияние на производительность сервера, но, с другой стороны, выделение слишком большого стека также приведёт к потреблению большого объёма памяти. Каждый поиск использует 512Кб на один уровень на 32-битной машине, или 1024Кб на один уровень на 64-битной машине. Глубина стека по умолчанию — 16, таким образом, каждому процессу выделяется по 8Мб или по 16Мб оперативной памяти на 32 и 64-битных машинах соответственно. Размер одного слота стека в 512Кб устанавливается с помощью константы во время компиляции, при необходимости его можно изменить; чтобы изменения вступили в силу, требуется перекомпиляция исходного кода.

Значение по умолчанию:

```
olcDbSearchStack: 16
```

5.2.6.11. olcDbShmKey: <целое число>

Указывает ключ разделяемой памяти для окружения BDB. По умолчанию окружение BDB использует отображаемые в памяти файлы. Если задано ненулевое значение, оно будет использовано как ключ для идентификации области разделяемой памяти, в которой будет помещено окружение.

Пример:

```
olcDbShmKey: 42
```

5.2.6.12. Пример записи

```
dn: olcDatabase=hdb,cn=config
objectClass: olcDatabaseConfig
objectClass: olcHdbConfig
olcDatabase: hdb
olcSuffix: "dc=example,dc=com"
olcDbDirectory: /usr/local/var/openldap-data
olcDbCacheSize: 1000
olcDbCheckpoint: 1024 10
olcDbConfig: set_cachesize 0 10485760 0
olcDbConfig: set_lg_bsize 2097152
olcDbConfig: set_lg_dir /var/tmp/bdb-log
olcDbConfig: set_flags DB_LOG_AUTOREMOVE
olcDbIDLcacheSize: 3000
olcDbIndex: objectClass eq
```

5.3. Пример конфигурации

Далее приведён пример конфигурации вперемешку с поясняющим его текстом. В нём определены две базы данных (обе они BDB), содержащие разные части дерева X.500. Номера строк приведены с целью дальнейших пояснений и не должны включаться в реальный файл. Итак, сначала секция глобальных настроек:

```
1.    # пример конфигурационного файла - секция глобальных настроек
2.    dn: cn=config
3.    objectClass: olcGlobal
4.    cn: config
5.    olcReferral: ldap://root.openldap.org
6.
```

Строка 1 — комментарий. Строки 2-4 идентифицируют данную запись как запись глобальной конфигурации. Директива `olcReferral:` в строке 5 означает, что на запросы, не относящиеся ни к одной из баз данных, определённых ниже, будет возвращаться отсылка на LDAP сервер, работающий на стандартном порту (389) по адресу `root.openldap.org`. Пустая строка 6 указывает на окончание определения записи.

```
 7.     # внутренняя схема данных
 8.     dn: cn=schema,cn=config
 9.     objectClass: olcSchemaConfig
10.     cn: schema
11.
```

Строка 7 — комментарий. Строки 8-10 идентифицируют данную запись как корень поддерева схемы данных. Актуальные определения схемы в этой записи жёстко вкомпилированы в slapd, поэтому не требуется дополнительных атрибутов для их указания при конфигурации. Пустая строка 11 указывает на окончание определения записи.

```
12.     # подключение набора схемы core
13.     include: file:///usr/local/etc/openldap/schema/core.ldif
14.
```

Строка 12 — комментарий. Строка 13 — это LDIF-директива include для доступа к определениям набора схемы *core* в формате LDIF. Строка 14 пуста.

Далее идут определения баз данных. Первая база данных — это специальная база данных `frontend`, настройки которой применяются глобально ко всем остальным базам данных.

```
15.     # глобальные параметры баз данных
16.     dn: olcDatabase=frontend,cn=config
17.     objectClass: olcDatabaseConfig
18.     olcDatabase: frontend
19.     olcAccess: to * by * read
20.
```

Строка 15 — комментарий. Строки 16-18 идентифицируют данную запись как запись глобальной базы данных. В строке 19 определён глобальный уровень контроля доступа. Он применяется ко всем записям (после применения к ним правил контроля доступа, специфичных для базы данных). Строка 20 пуста.

Следующая запись определяет механизм манипуляции данными config.

```
21.     # поставим rootpw для БД config, чтобы можно было к ней подсоединиться.
22.     # запретим доступ всем остальным.
23.     dn: olcDatabase=config,cn=config
24.     objectClass: olcDatabaseConfig
25.     olcDatabase: config
26.     olcRootPW: {SSHA}XKYnrjvGT3wZFQrDD5040US592LxsdLy
27.     olcAccess: to * by * none
28.
```

Строки 21 и 22 — комментарии. Строки 23-25 идентифицируют данную запись как запись базы данных config. В строке 26 определён пароль *администратора* для этой базы данных. (По умолчанию DN администратора — *"cn=config"*.) Строка 27 запрещает любой доступ к этой базе данных, таким образом, только администратор может получить к ней доступ. (Такой контроль доступа установлен к базе данных config по умолчанию. Здесь он приведён только для иллюстрации, и для того, чтобы еще раз подчеркнуть, что если параметры аутентификации администратора не будут заданы явно, база данных config будет недоступна).

Строка 28 пуста.

Следующая запись определяет базу данных механизма манипуляции данными BDB, которая будет обрабатывать запросы к части дерева "dc=example,dc=com". К некоторым атрибутам будет применяться индексирование, а атрибут `userPassword` защищается от несанкционированного доступа.

```
29.     # определения BDB для example.com
30.     dn: olcDatabase=bdb,cn=config
31.     objectClass: olcDatabaseConfig
32.     objectClass: olcBdbConfig
33.     olcDatabase: bdb
34.     olcSuffix: dc=example,dc=com
35.     olcDbDirectory: /usr/local/var/openldap-data
36.     olcRootDN: cn=Manager,dc=example,dc=com
37.     olcRootPW: secret
38.     olcDbIndex: uid pres,eq
39.     olcDbIndex: cn,sn pres,eq,approx,sub
40.     olcDbIndex: objectClass eq
41.     olcAccess: to attrs=userPassword
42.       by self write
43.       by anonymous auth
44.       by dn.base="cn=Admin,dc=example,dc=com" write
45.       by * none
46.     olcAccess: to *
```

```
47.       by self write
48.       by dn.base="cn=Admin,dc=example,dc=com" write
49.       by * read
50.
```

Строка 29 — комментарий. Строки 30-33 идентифицируют данную запись как запись конфигурации базы данных BDB. В строке 34 определён DN суффикса запросов, обслуживаемых этой базой данных. В строке 35 указана директория, в которой будут размещаться файлы базы данных.

Строки 36 и 37 идентифицируют запись *администратора* базы данных и соответствующий пароль. На эту запись не налагаются ограничения контроля доступа, размера запрашиваемых данных или продолжительности выполнения запроса.

Строки с 38 по 40 устанавливают индексирование для различных атрибутов.

Строки с 41 по 49 определяют контроль доступа к записям в этой базе данных. Во всех записях с атрибутом `userPassword` изменять этот атрибут может пользователь, ассоциированный с этой же самой записью или с записью "admin". Этот атрибут может использоваться в целях аутентификации/авторизации, в ином случае он недоступен для чтения. Все остальные атрибуты доступны для изменения пользователем, ассоциированным с этой же самой записью или с записью "admin", но доступны для чтения всем пользователям, независимо от того, прошли они аутентификацию или нет.

Пустая строка 50 указывает на окончание определения записи.

Следующая запись определяет другую базу данных BDB. Она обслуживает запросы к поддереву `dc=example,dc=net`, но управляется той же самой учётной записью, что и первая база данных. Обратите внимание, что если опустить строку 60, доступ на чтение всё равно будет предоставлен по глобальному правилу доступа в строке 19.

```
51.    # определения BDB для example.net
52.    dn: olcDatabase=bdb,cn=config
53.    objectClass: olcDatabaseConfig
54.    objectClass: olcBdbConfig
55.    olcDatabase: bdb
56.    olcSuffix: "dc=example,dc=net"
57.    olcDbDirectory: /usr/local/var/openldap-data-net
58.    olcRootDN: "cn=Manager,dc=example,dc=com"
59.    olcDbIndex: objectClass eq
60.    olcAccess: to * by users read
```

5.4. Конвертирование конфигурационного файла в старом стиле *slapd.conf*(5) в формат *cn=config*

Перед тем как конвертировать в формат *cn=config*, Вы должны убедиться, что механизм манипуляции данными config корректно настроен в уже имеющемся у Вас конфигурационном файле. Дело в том, что механизм config является встроенным и всегда присутствует в slapd. По умолчанию права доступа к нему имеет только его rootDN, однако нет учетных данных, ассоциированных с этой rootDN по умолчанию. Таким образом, если Вы явно не настроите учётные данные для аутентификации к механизму config, его нельзя будет использовать.

Если у Вас ещё нет секции `database config`, добавьте что-то вроде этого в конец файла `slapd.conf`

```
database config
rootpw VerySecret
```

Замечание: Поскольку механизм config может быть использован для загрузки произвольного кода в процесс slapd, особенно важно обеспечить тщательную сохранность всех учётных данных для доступа к нему. Простые пароли уязвимы к атакам методом перебора, поэтому, как правило, лучше отказаться от rootpw и использовать только аутентификацию SASL для rootDN механизма config.

Существующий файл *slapd.conf*(5) может быть переконвертирован в новый формат с помощью *slaptest*(8) или любой из slap-утилит:

```
slaptest -f /usr/local/etc/openldap/slapd.conf -F /usr/local/etc/openldap/slapd.d
```

Попробуем возможность доступа к записям поддерева `cn=config` с использованием *rootdn* по умолчанию и *rootpw*, сконфигурированного выше:

```
ldapsearch -x -D cn=config -w VerySecret -b cn=config
```

После этого Вы можете отказаться от старого файла *slapd.conf*(5). Убедитесь, что Вы запускаете *slapd*(8) с параметром *-F* для указания пути к конфигурационной директории, если Вы не используете конфигурационную директорию по умолчанию.

Замечание: При конвертировании из формата slapd.conf в формат slapd.d все подключаемые файлы должны быть также интегрированы в итоговую базу данных конфигурации.

К содержанию

6. Конфигурационный файл slapd

В этом разделе описывается настройка *slapd*(8) с помощью конфигурационного файла *slapd.conf*(5). *slapd.conf*(5) является устаревшим и использовать его нужно лишь в том случае, если для работы Вашего каталога требуется один из тех механизмов манипуляции данными, которые ещё не были адаптированы для работы с более новой системой конфигурации *slapd-config*(5). Настройка *slapd*(8) с помощью *slapd-config*(5) описана в предыдущем разделе.

Файл *slapd.conf*(5) по умолчанию устанавливается в директорию `/usr/local/etc/openldap`. Альтернативное расположение конфигурационного файла можно указать с помощью параметра командной строки при запуске *slapd*(8).

6.1. Формат конфигурационного файла

Файл *slapd.conf*(5) состоит из трёх типов настроечной информации: глобальной, специфичной для механизмов манипуляции данными и специфичной для баз данных. Глобальные настройки указываются первыми, затем следуют настройки для конкретного типа механизма манипуляции данными, за которыми, в свою очередь, следуют настройки конкретного экземпляра базы данных. Глобальные директивы могут переопределяться директивами механизма манипуляции данными и/или базы данных, а директивы механизма манипуляции данными могут переопределяться директивами базы данных.

Пустые строки и строки комментариев, начинающиеся с символа '#' игнорируются. Если строка начинается с пробельного символа, она считается продолжением предыдущей строки (даже если предыдущая строка является комментарием).

Общий формат файла slapd.conf:

```
# глобальные настройки
<глобальные директивы конфигурации>

# определение механизма манипуляции данными
backend <typeA>
<директивы, специфичные для ММД>

# определение и директивы конфигурации первой базы данных
database <typeA>
<директивы, специфичные для БД>

# определение и директивы конфигурации второй базы данных
database <typeB>
<директивы, специфичные для БД>

# определение и директивы конфигурации третьей базы данных
database <typeC>
<директивы, специфичные для БД>
```

```
# последующие определения и директивы конфигурации механизмов манипуляции данными и баз данных
...
```

У директивы конфигурации могут быть аргументы, которые отделяются друг от друга пробельными символами. Если пробельный символ содержится внутри аргумента, аргумент заключается в двойные кавычки `"вот так"`. Если аргумент содержит двойные кавычки или символ обратного слеша `\', данные символы должны экранироваться символом обратного слеша `\'.

В дистрибутиве есть пример конфигурационного файла, устанавливаемый в директорию `/usr/local/etc/openldap`. Также есть несколько файлов, содержащих наборы определения схемы (типы атрибутов и объектные классы), устанавливаемых в директорию `/usr/local/etc/openldap/schema`.

6.2. Директивы конфигурационного файла

В этом подразделе детально описаны часто используемые директивы конфигурации, полный список директив можно найти в man-странице *slapd.conf*(5). Директивы конфигурационного файла разделены на категории глобальных, специфичных для механизмов манипуляции данными и специфичных для баз данных. Для каждой директивы дается её описание, значение по умолчанию (если есть), и пример использования.

6.2.1. Глобальные директивы

Описанные в данном подразделе директивы применяются ко всем механизмам манипуляции данными и базам данных, если они специально не переопределяются настройками конкретного механизма манипуляции или базы данных. Аргументы, которые требуется заменить актуальными значениями, показаны в угловых скобках `<>`.

6.2.1.1. access to <what> [by <who> [<accesslevel>] [<control>]]+

Эта директива предоставляет доступ (заданный аргументом <accesslevel>) к набору записей и/или атрибутов (заданный аргументом <what>) одному или нескольким пользователям (заданным аргументом <who>). Смотрите раздел Контроль доступа данного руководства для получения подробных сведений о настройке.

Замечание: Если не задано ни одной директивы `access`, применяется политика контроля доступа по умолчанию `access to * by * read`, предоставляющая доступ на чтение как авторизованным, так и анонимным пользователям.

6.2.1.2. attributetype <описание типа атрибута согласно RFC4512>

Эта директива определяет тип атрибута. О том, как использовать эту директиву, читайте в разделе Спецификация схемы.

6.2.1.3. idletimeout <целое число>

Определяет количество секунд до того, как принудительно закрыть простаивающее клиентское соединение. При установке idletimeout в 0 (значение по умолчанию), эта функция отключается.

6.2.1.4. include <имя файла>

Эта директива определяет, что slapd должен прочитать дополнительную информацию о конфигурации из указанного файла, перед тем как перейти к обработке следующей строки текущего файла. Формат подключаемого файла должен соответствовать нормальному формату конфигурационного файла slapd. В качестве примера обычно подключаемых файлов можно привести файлы, содержащие наборы спецификаций схемы.

Замечание: При использовании данной директивы соблюдайте осторожность: не существует разумных ограничений на количество вложенных директив include, и зацикливание не распознаётся.

6.2.1.5. loglevel <уровень>

Эта директива определяет уровень отладочной информации и статистики работы slapd, которая должна быть журналирована (в настоящее время журналируется в канал `LOG_LOCAL4` *syslogd*(8)). Чтобы это работало (за исключением двух уровней статистики, которые всегда включены), при конфигурации перед сборкой OpenLDAP нужно указать `--enable-debug` (по умолчанию включено). Уровни журналирования могут быть указаны целым числом или ключевым словом. Можно задать несколько уровней журналирования, либо указать несколько уровней как сумму соответствующих целых чисел. Чтобы узнать соответствие числового значения типу отладочной информации, запустите slapd с ключом `-d?` или посмотрите таблицу ниже. Возможные значения <уровня>:

Таблица 6.1: Уровни отладки

Уровень	**Ключевое слово**	**Описание**
-1	any	включить всю отладочную информацию
0		отключить отладку
1	(0x1 trace)	отслеживать вызовы функций
2	(0x2 packets)	отладка обработки пакетов
4	(0x4 args)	усиленная отладка
8	(0x8 conns)	управление соединениями
16	(0x10 BER)	вывод посылки и приёма пакетов
32	(0x20 filter)	обработка поисковых фильтров
64	(0x40 config)	обработка конфигурации
128	(0x80 ACL)	обработка списков контроля доступа
256	(0x100 stats)	статистика соединений/операций/результатов
512	(0x200 stats2)	статистика посылки записей журнала
1024	(0x400 shell)	вывод взаимодействия с механизмами манипуляции данными shell
2048	(0x800 parse)	вывод отладочной информации разбора записей
16384	(0x4000 sync)	обслуживание потребителей syncrepl
32768	(0x8000 none)	только сообщения, выводимые независимо от заданного уровня журналирования (то есть высокоприоритетные)

Требуемый уровень журналирования может быть задан одним целым числом (десятичным или шестнадцатеричным), которое является комбинацией нужных уровней журналирования (соединяемых через логическое ИЛИ). Также он может быть задан списком целых чисел (опять же соединяемых через логическое ИЛИ), либо списком ключевых слов, указанных в скобках в соответствующем столбце таблицы выше. Таким образом, директивы

```
loglevel 129
loglevel 0x81
loglevel 128 1
loglevel 0x80 0x1
loglevel acl trace
```

эквивалентны.

Примеры:

```
loglevel -1
```

Это вызывает журналирование большого количества отладочной информации.

```
loglevel conns filter
```

Журналирование только информации о соединениях и обработке поисковых фильтров.

```
loglevel none
```

Журналирование только тех сообщений, которые выводятся вне зависимости от настроенного уровня журналирования. Это отличается от установки уровня журналирования в 0, когда журналирование отсутствует. Для журналирования высокоприоритетных сообщений требуется как минимум уровень `None`.

Значение по умолчанию:

```
loglevel stats
```

По умолчанию настраивается журналирование основных статистических сведений. Однако, если не определено никакого loglevel, журналирования не происходит (как в уровне 0).

6.2.1.6. objectclass <описание объектного класса согласно RFC4512>

Эта директива определяет объектный класс. О том, как использовать эту директиву, читайте в разделе Спецификация схемы.

6.2.1.7. referral <URI>

Эта директива определяет отсылку, которая возвращается клиенту, когда slapd не может найти подходящую локальную базу данных для обработки запроса.

Пример:

```
referral ldap://root.openldap.org
```

С помощью этой директивы в проекте OpenLDAP нелокальные запросы перенаправляются на глобальный LDAP-сервер корневого уровня. Продвинутые клиенты LDAP могут сделать повторный запрос на указанный сервер. Но имейте в виду, что большинство клиентов способны обрабатывать только простые LDAP URL, состоящие из адреса хоста и, возможно, из уникального имени записи.

6.2.1.8. sizelimit <целое число>

Эта директива определяет максимальное количество записей, возвращаемых поисковым запросом.

Значение по умолчанию:

```
sizelimit 500
```

Для получения дополнительной информации смотрите раздел Ограничения этого руководства и slapd.conf(5).

6.2.1.9. timelimit <целое число>

Эта директива указывает максимальное количество секунд, которое slapd может потратить, чтобы ответить на поисковый запрос. Если запрос не был обработан за это время, клиенту возвращается уведомление о превышении времени обработки.

Значение по умолчанию:

```
timelimit 3600
```

Для получения дополнительной информации смотрите раздел Ограничения этого руководства и slapd.conf(5).

6.2.2. Общие директивы механизмов манипуляции данными

Директивы этого подраздела применяются только к тем механизмам манипуляции данными, для которых они определены. Они поддерживаются всеми типами механизмов манипуляции данными. Директивы механизма манипуляции данными применяются ко всем базам данных того же типа и, в зависимости от директивы, могут быть переопределены директивами конкретной базы данных.

6.2.2.1. backend <тип>

Эта директива обозначает начало определения настроек экземпляра механизма манипуляции данными. `<Тип>` должен быть одним из поддерживаемых типов механизмов, перечисленных в таблице 6.2.

Таблица 6.2: Механизмы манипуляции данными

Тип	**Описание**
`bdb`	Механизм Berkeley DB с поддержкой транзакций
`dnssrv`	Механизм DNS SRV
`hdb`	Иерархический вариант механизма bdb
`ldap`	Механизм Lightweight Directory Access Protocol (прокси)
`meta`	Механизм мета-каталога
`monitor`	Механизм мониторинга
`passwd`	Предоставление доступа только для чтения к *passwd*(5)
`perl`	Программируемый механизм Perl
`shell`	Механизм запуска внешних программ Shell
`sql`	Программируемый механизм SQL

Пример:

```
backend bdb
```

Здесь обозначено начало определения настроек нового экземпляра механизма манипуляции данными BDB.

6.2.3. Общие директивы баз данных

Директивы этого подраздела применяются только к тем базам данных, для которых они определены. Они поддерживаются всеми типами баз данных.

6.2.3.1. database <тип>

Эта директива обозначает начало определения настроек экземпляра базы данных. `<Тип>` должен быть одним из поддерживаемых типов механизмов манипуляции данными, перечисленных в таблице 6.2.

Пример:

```
database bdb
```

Здесь обозначено начало определения настроек нового экземпляра базы данных BDB.

6.2.3.2. limits <кто> <ограничение> [<ограничение> [...]]

Определяет ограничения по времени и размеру в зависимости от того, кто инициировал операцию.

Для получения дополнительной информации смотрите раздел Ограничения этого руководства и slapd.conf(5).

6.2.3.3. readonly { on | off }

Эта директива переводит базу данных в режим "только для чтения". На все попытки внесения в базу данных изменений возвращается ошибка "unwilling to perform" ("не желают выполнять").

Значение по умолчанию:

```
readonly off
```

6.2.3.4. rootdn <DN>

Эта директива определяет DN, на который не будут накладываться контроль доступа и административные

ограничения при выполнении операций с этой базой данных. Данный DN не обязательно должен указывать на запись из этой базы данных, и даже на запись из этого каталога. Данный DN может ссылаться на идентификационную сущность SASL.

Пример с записью:

```
rootdn "cn=Manager,dc=example,dc=com"
```

Пример с SASL:

```
rootdn "uid=root,cn=example.com,cn=digest-md5,cn=auth"
```

Информацию об аутентификационных идентификационных сущностях SASL можно посмотреть в разделе Аутентификация SASL.

6.2.3.5. rootpw <пароль>

Эта директива может использоваться для назначения пароля для DN из rootdn (когда в качестве rootdn назначен DN из этой базы данных).

Пример:

```
rootpw secret
```

Также можно назначить хэш пароля в форме RFC2307. Для генерации хэша пароля можно использовать *slappasswd*(8).

Пример:

```
rootpw {SSHA}ZKKuqbEKJfKSXhUbHG3fG8MDn9j1v4QN
```

Данный хэш был сгенерирован командой `slappasswd -s secret`.

6.2.3.6. suffix <dn суффикса>

Эта директива определяет DN суффикса запросов, которые будут направляться в эту базу данных. Возможно определение нескольких суффиксов. Как минимум один суффикс должен быть определён для каждой базы данных.

Пример:

```
suffix "dc=example,dc=com"
```

Запросы с DN, оканчивающимся на "dc=example,dc=com" будут направлены в эту базу данных.

Замечание: Когда выбран механизм манипуляции данными для обработки запроса, slapd ищет строку (строки) определения суффикса в каждом определении базы данных в порядке их указания в файле. Поэтому, если один суффикс базы данных является префиксом к другому, то более общий суффикс должен указываться позже в конфигурационном файле.

6.2.3.7. syncrepl

```
syncrepl rid=<ID реплики>
        provider=ldap[s]://<имя хоста>[:порт]
        [type=refreshOnly|refreshAndPersist]
        [interval=dd:hh:mm:ss]
        [retry=[<интервал между повторами> <количество повторов>]+]
        searchbase=<base DN>
        [filter=<строка фильтра>]
        [scope=sub|one|base]
        [attrs=<список атрибутов>]
        [attrsonly]
        [sizelimit=<limit>]
        [timelimit=<limit>]
        [schemachecking=on|off]
```

```
[bindmethod=simple|sasl]
[binddn=<DN>]
[saslmech=<mech>]
[authcid=<identity>]
[authzid=<identity>]
[credentials=<passwd>]
[realm=<realm>]
[secprops=<properties>]
[starttls=yes|critical]
[tls_cert=<file>]
[tls_key=<file>]
[tls_cacert=<file>]
[tls_cacertdir=<path>]
[tls_reqcert=never|allow|try|demand]
[tls_ciphersuite=<ciphers>]
[tls_crlcheck=none|peer|all]
[logbase=<base DN>]
[logfilter=<filter str>]
[syncdata=default|accesslog|changelog]
```

Эта директива определяет, что текущая база данных будет репликой содержимого каталога основного сервера, и текущий *slapd*(8) будет работать в роли сервера-потребителя репликации путём запуска механизма syncrepl. Основная база данных располагается на сервере-поставщике репликации, указанном в параметре `provider`. База данных реплики синхронизируется с содержимым каталога основного сервера с использованием LDAP Content Synchronization protocol (протокол синхронизации содержимого LDAP). Описание протокола можно найти в RFC4533 (рус.) (RFC4533 (ориг.)).

Параметр `rid` используется для идентификации текущей директивы `syncrepl` в пределах сервера-потребителя репликации. `<ID реплики>` - уникальный идентификатор спецификации syncrepl, описываемой текущей директивой `syncrepl`. `<ID реплики>` должен быть положительным числом длиной не более трёх десятичных цифр.

Параметр `provider` определяет адрес поставщика репликации как LDAP URI. Параметр `provider` указывается схемой (ldap или ldaps), именем хоста и, опционально, номером порта, на котором будет слушать slapd поставщика. В качестве <имени хоста> может быть как доменное имя, так и IP-адрес. Например, `ldap://provider.example.com:389` или `ldaps://192.168.1.1:636`. Если <порт> не задан, используются стандартные номера портов LDAP (389 или 636). Обратите внимание, что механизм syncrepl использует протокол, инициируемый со стороны потребителя, поэтому директивы его настройки располагаются на сервере потребителя, в то время как директивы `replica` располагаются на стороне поставщика. Директивы `syncrepl` и `replica` определяют две независимые части механизма репликации. Они не устанавливают никаких взаимных потоков репликации друг к другу.

Содержимое реплики syncrepl формируется как результат поискового запроса. slapd потребителя посылает поисковые запросы к slapd поставщика согласно данной ему спецификации поискового запроса. Спецификация поискового запроса включает в себя параметры `searchbase`, `scope`, `filter`, `attrs`, `attrsonly`, `sizelimit` и `timelimit`, как и в случае обычного поискового запроса. У параметра `searchbase` нет значения по умолчанию и он должен быть задан обязательно. Значения по умолчанию остальных параметров: `scope - sub`, `filter - (objectclass=*)`, `attrs - "*,+"` (то есть репликация и пользовательских, и операционных атрибутов), `attrsonly` по умолчанию не задан. Параметры `sizelimit` и `timelimit` по умолчанию установлены в "unlimited", и могут задаваться либо положительным целым числом, либо "unlimited".

У протокола LDAP Content Synchronization два типа операций: `refreshOnly` и `refreshAndPersist`. Тип операций задаётся параметром `type`. При типе операций `refreshOnly` время выполнения следующей операции синхронизационного поиска периодически переназначается на заданный временной интервал после окончания предыдущей операции синхронизации. Интервал указывается параметром `interval`, по умолчанию заданным в 1 день. При типе операций `refreshAndPersist` синхронизационный поиск постоянно выполняется в экземпляре *slapd* поставщика. При дальнейшем обновлении каталога поставщика будет генерироваться `searchResultEntry` и отправляться slapd потребителя как ответ на постоянный синхронизационный поиск на стороне поставщика.

При возникновении ошибок в процессе репликации, потребитель будет пытаться сделать повторное подключение в соответствии с указанным в параметре retry списком пар значений <интервал между повторами> и <количество повторов>. Например, retry="60 10 300 3" позволяет получателю повторять запрос через каждые 60 секунд первые 10 раз, а затем через каждые 300 секунд еще 3 раза перед тем, как прекратить попытки. Если в <количестве повторов> указан +, то повторные попытки будут осуществляться до получения положительного результата (неограниченное количество раз).

С помощью параметра `schemachecking` можно включить принудительную проверку получаемых данных на

соответствие схеме данных на стороне потребителя LDAP Sync. Если он установлен в on, каждая реплицируемая запись будет проверяться, не нарушит ли она целостности схемы данных при помещении её в базу данных реплики. Каждая запись в реплике должна содержать те атрибуты, которые помечены как обязательные в определении схемы. Если этот параметр установлен в off, записи будут сохраняться без проверки на соответствие схеме. Значение по умолчанию - off.

Параметр `binddn` указывает DN, под которым будет происходить соединение с slapd поставщика для выполнения поисковых запросов syncrepl. Этот DN должен иметь права на чтение к реплицируемому содержимому каталога на главном сервере.

Параметр `bindmethod` может быть либо `simple`, либо `sasl`, в зависимости от метода аутентификации (простая парольная или SASL), используемого при соединении с *slapd* поставщика.

Лучше не использовать простую аутентификацию, если не предприняты адекватные меры защиты целостности и конфиденциальности данных (такие как TLS или IPsec). При использовании простой аутентификации требуется задать параметры `binddn` и `credentials`.

В общем случае рекомендуется использование аутентификации SASL. Аутентификация SASL требует указания механизма с использованием параметра `saslmech`. В зависимости от механизма, можно задать аутентификационную идентификационную сущность и/или учётные данные с помощью соответствующих параметров `authcid` и `credentials`. Параметр `authzid` может быть использован для определения авторизационной идентификационной сущности.

Параметр `realm` указывает realm, с которым некоторые механизмы проводят аутентификацию идентификационной сущности. В параметре `secprops` указываются настройки безопасности Cyrus SASL.

Параметр `starttls` указывает использование расширенной операции StartTLS для установки сессии TLS до начала аутентификации при подключении к серверу-поставщику. Если установлен аргумент `critical`, сессия будет прервана в случае неудачного завершения запроса StartTLS. Если этот аргумент не был установлен, сессия syncrepl будет продолжена без TLS. Обратите внимание, что основные настройки TLS для slapd не используются механизмом syncrepl; по умолчанию будут использоваться параметры TLS из конфигурационного файла *ldap.conf*(5). Если же указать настройки TLS в данной секции конфигурационного файла *slapd.conf*(5), то любые настройки из *ldap.conf*(5) будут полностью проигнорированы.

Вместо того, чтобы реплицировать записи целиком, потребитель может запрашивать журналы изменений данных. Этот режим работы называется *дельта-syncrepl*. В дополнение к вышеперечисленным параметрам, нужно установить параметры `logbase` и `logfilter` в соответствии с типом журнала, который будет использоваться. Параметр `syncdata` должен быть задан либо в `"accesslog"` если журнал соответствует формату *slapo-accesslog*(5), либо в `"changelog"` если журнал соответствует устаревшему формату *changelog*. Если параметр `syncdata` пропущен или установлен в `"default"`, то параметры журналирования игнорируются.

Механизм репликации *syncrepl* поддерживается механизмами манипуляции данными *bdb* и *hdb*.

Дополнительная информация по использованию этой директивы в разделе Репликация на основе LDAP Sync этого руководства.

6.2.3.8. updateref <URL>

Эта директива применяется только в случае запуска *slapd*(8) в режиме *подчинённого* (или *теневого*) сервера. Она указывает URL, возвращаемый клиентам при попытке выполнить на реплике запрос на обновление. Если директива указывается несколько раз, клиенту возвращаются все URL.

Пример:

```
updateref       ldap://master.example.net
```

6.2.4. Директивы баз данных BDB и HDB

Директивы этой категории применимы только к базам данных BDB и HDB. Поэтому они должны следовать за строками "database bdb" или "database hdb" и до любых последующих строк "backend" или "database". Полное руководство по конфигурационным директивам BDB/HDB смотрите в *slapd-bdb*(5).

6.2.4.1. directory <директория>

Эта директива указывает директорию, где будут находиться файлы BDB, содержащие базу данных, и соответствующие им индексы.

Значение по умолчанию:

```
directory /usr/local/var/openldap-data
```

6.3. Пример конфигурационного файла

Далее приведён пример конфигурационного файла вперемешку с поясняющим его текстом. В нём определены две базы данных (обе они BDB), содержащие разные части дерева X.500. Номера строк приведены с целью дальнейших пояснений и не должны включаться в реальный файл. Итак, сначала секция глобальных настроек:

```
1.    # пример конфигурационного файла - секция глобальных настроек
2.    include /usr/local/etc/schema/core.schema
3.    referral ldap://root.openldap.org
4.    access to * by * read
```

Строка 1 - комментарий. В строке 2 подключается другой конфигурационный файл, содержащий определение набора схемы *core*. Директива `referral` в строке 3 означает, что на запросы, не относящиеся ни к одной из баз данных, определённых ниже, будет возвращаться отсылка на LDAP сервер, работающий на стандартном порту (389) по адресу `root.openldap.org`.

В строке 4 определён глобальный уровень контроля доступа. Он применяется ко всем записям (после применения к ним правил контроля доступа, специфичных для базы данных).

В следующей секции конфигурационного файла определена база данных механизма манипуляции данными BDB, которая будет обрабатывать запросы к части дерева "dc=example,dc=com". База данных будет реплицироваться на два подчинённых slapd, один из которых на хосте truelies, а другой - на judgmentday. К некоторым атрибутам будет применяться индексирование, а атрибут `userPassword` защищается от несанкционированного доступа.

```
5.    # Определение BDB для example.com
6.    database bdb
7.    suffix "dc=example,dc=com"
8.    directory /usr/local/var/openldap-data
9.    rootdn "cn=Manager,dc=example,dc=com"
10.   rootpw secret
11.   # определения индексирования атрибутов
12.   index uid pres,eq
13.   index cn,sn pres,eq,approx,sub
14.   index objectClass eq
15.   # определения контроля доступа к базе данных
16.   access to attrs=userPassword
17.       by self write
18.       by anonymous auth
19.       by dn.base="cn=Admin,dc=example,dc=com" write
20.       by * none
21.   access to *
22.       by self write
23.       by dn.base="cn=Admin,dc=example,dc=com" write
24.       by * read
```

Строка 5 - комментарий. Начало определения базы данных обозначено ключевым словом database в строке 6. В строке 7 определён DN суффикса запросов, обслуживаемых этой базой данных. В строке 8 указана директория, в которой будут размещаться файлы базы данных.

В строках 9 и 10 определены запись администратора базы данных и соответствующий пароль. На эту запись не налагаются ограничения контроля доступа, размера запрашиваемых данных или продолжительности выполнения запроса.

Строки с 12 по 14 устанавливают индексирование для различных атрибутов.

Строки с 16 по 24 определяют контроль доступа к записям в этой базе данных. Во всех записях с атрибутом

`userPassword` изменять этот атрибут может пользователь, ассоциированный с этой же самой записью или с записью "admin". Этот атрибут может использоваться в целях аутентификации/авторизации, в ином случае он недоступен для чтения. Все остальные атрибуты доступны для изменения пользователем, ассоциированным с этой же самой записью или с записью "admin", но доступны для чтения всем пользователям, независимо от того, прошли они аутентификацию или нет.

В следующей секции нашего примера файла конфигурации определяется другая база данных BDB. Она обслуживает запросы к поддереву `dc=example,dc=net`, но управляется той же самой учётной записью, что и первая база данных. Обратите внимание, что если опустить строку 39, доступ на чтение всё равно будет предоставлен по глобальному правилу доступа в строке 4.

```
33.    # Определение BDB для example.net
34.    database bdb
35.    suffix "dc=example,dc=net"
36.    directory /usr/local/var/openldap-data-net
37.    rootdn "cn=Manager,dc=example,dc=com"
38.    index objectClass eq
39.    access to * by users read
```

К содержанию

7. Запуск slapd

slapd(8) предназначен для работы в качестве автономной службы. Такой подход позволяет серверу воспользоваться преимуществом кэширования, управлять проблемами параллельной работы основных баз данных и экономить системные ресурсы. Запуск с помощью *inetd*(8) *НЕ* допускается.

7.1. Параметры командной строки

slapd(8) поддерживает несколько параметров командной строки, подробно описанных в man-странице. В этом подразделе дано описание некоторых часто используемых параметров.

```
-f <filename>
```

Этот параметр задает альтернативный файл конфигурации для slapd. Обычно по умолчанию данный файл располагается в `/usr/local/etc/openldap/slapd.conf`.

```
-F <slapd-config-directory>
```

Задаёт расположение конфигурационной директории slapd. По умолчанию `/usr/local/etc/openldap/slapd.d`.

Если задано сразу оба параметра `-f` и `-F`, конфигурационный файл будет прочитан, переконвертирован в формат конфигурационной директории и записан в указанную директорию. Если ни одного из этих двух параметров не было задано, slapd попытается прочитать конфигурационную директорию по умолчанию перед тем, как попытаться использовать конфигурационный файл по умолчанию. Если существует конфигурационная директория правильного формата, тогда конфигурационный файл по умолчанию игнорируется. Все slap-утилиты, использующие настройки конфигурации, ведут себя аналогично.

```
-h <URLs>
```

Данный параметр указывает настройки того, на каких интерфейсах/портах slap будет ожидать подключения. Значение по умолчанию - `ldap:///`, что означает, что slapd будет ожидать подключения LDAP по TCP на всех интерфейсах на стандартном порту LDAP 389. Вы можете задать конкретные пары хост-порт, либо другие схемы протокола (такие, как `ldaps://` или `ldapi://`).

URL	Протокол	Транспорт
ldap:///	LDAP	TCP порт 389
ldaps:///	LDAP поверх SSL	TCP порт 636
ldapi:///	LDAP	IPC (Unix-сокеты)

Например, `-h "ldaps:// ldap://127.0.0.1:666"` укажет slapd обслуживать два варианта подключений: один из них по (нестандартной) схеме `ldaps://` на всех интерфейсах на стандартном для `ldaps://` порту 636, а второй - по стандартной схеме `ldap://` на интерфейсе `localhost` (*loopback*) на порту 666. Хосты можно задавать либо именем хоста, либо адресом IPv4 или IPv6. Значение номера порта должно быть числовым.

Для LDAP поверх IPC, в URL может быть указан путь к Unix-сокету. Обратите внимание, что разделительные символы директорий должны быть закодированы в формате URL, как и все остальные символы, являющиеся специальными для URL. Таким образом, сокет `/usr/local/var/ldapi` должен быть закодирован как

```
ldapi://%2Fusr%2Flocal%2Fvar%2Fldapi
```

ldapi: подробно описан в *Using LDAP Over IPC Mechanisms (Использование LDAP поверх механизмов IPC)* [Chu-LDAPI].

Имейте ввиду, что транспорт ldapi:/// пока не получил широкого распространения: не OpenLDAP-клиенты могут быть неспособны его использовать.

```
-n <имя сервиса>
```

Этот параметр указывает имя сервиса, которое используется при журналировании и для других целей. Имя сервиса по умолчанию - `slapd`.

```
-l <syslog-local-user>
```

Этот параметр указывает имя пользовательского канала для *syslog*(8). Значения могут быть `LOCAL0`, `LOCAL1`, `LOCAL2`, ..., и до `LOCAL7`. Значение по умолчанию - `LOCAL4`. Данный параметр может не поддерживаться на всех системах.

```
-u user -g group
```

Эти параметры указывают, соответственно, пользователя и группу, под которыми будет запущен slapd. `user` может быть либо именем пользователя, либо uid. `group` может быть либо именем группы, либо gid.

```
-r директория
```

Этот параметр задаёт директорию времени исполнения. slapd выполнит *chroot*(2) в эту директорию после того, как начнёт слушать на указанных ему портах, но до того, как начнёт чтение любых конфигурационных файлов или инициализацию любых механизмов манипуляции данными.

```
-d <уровень> | ?
```

Этот параметр устанавливает уровень отладки в требуемый <уровень>. Если в качестве уровня был передан символ `?', slapd выведет различные доступные уровни отладки и завершит работу, независимо от того, какие ещё параметры Вы ему передали. Текущие уровни отладки:

Таблица 7.1: Уровни отладки

Уровень	Ключевое слово	Описание
-1	any	включить всю отладочную информацию
0		отключить отладку
1	(0x1 trace)	отслеживать вызовы функций
2	(0x2 packets)	отладка обработки пакетов
4	(0x4 args)	усиленная отладка
8	(0x8 conns)	управление соединениями
16	(0x10 BER)	вывод посылки и приёма пакетов
32	(0x20 filter)	обработка поисковых фильтров
64	(0x40 config)	обработка конфигурации
128	(0x80 ACL)	обработка списков контроля доступа
256	(0x100 stats)	статистика соединений/операций/результатов

512	(0x200 stats2)	статистика посылки записей журнала
1024	(0x400 shell)	вывод взаимодействия с механизмами манипуляции данными shell
2048	(0x800 parse)	вывод отладочной информации разбора записей
16384	(0x4000 sync)	обслуживание потребителей syncrepl
32768	(0x8000 none)	только сообщения, выводимые независимо от заданного уровня журналирования (то есть высокоприоритетные)

Вы можете задать несколько уровней отладки одновременно, указав по одному параметру `-d` для каждого желаемого уровня. Либо, поскольку уровни отладки можно складывать, Вы можете сами вычислить сумму и передать её slapd. Например, если Вам нужно отслеживать вызовы функций и проследить за обработкой конфигурационного файла, можно задать требуемый уровень как сумму этих двух уровней (в данном случае, `-d 65`). Либо, Вы можете позволить самому slapd произвести вычисления, (то есть `-d 1 -d 64`). Подробности смотрите в `<ldap_log.h>`.

Примечание: для того, чтобы была доступна вся отладочная информация, помимо двух уровней stats, доступных по умолчанию, slapd должен быть скомпилирован с опцией `--enable-debug`.

7.2. Старт slapd

Обычно slapd запускается так:

```
/usr/local/libexec/slapd [<параметр>]*
```

где `/usr/local/libexec` определяется во время выполнения `configure`, а <параметр> - один из описанных выше (или в *slapd*(8)) параметров. Если Вы не указали уровень отладки (включая уровень `0`), slapd автоматически выполнит ответвление (fork), отделит себя от управляющего терминала и начнёт работать в фоновом режиме.

7.3. Остановка slapd

Для безопасной остановки *slapd*(8) нужно выполнить такую команду:

```
kill -INT `cat /usr/local/var/slapd.pid`
```

где `/usr/local/var` определяется во время выполнения `configure`.

Остановка slapd более радикальными методами может привести к потере информации или повреждению базы данных.

К содержанию

8. Контроль доступа

8.1. Введение

По мере заполнения каталога всё большим количеством данных различной степени важности, контроль за тем, какого рода доступ предоставляется к каталогу, становится всё более критичным. К примеру, каталог может содержать информацию конфиденциального характера, которую, возможно, требуется защищать согласно договору или закону. Или, если каталог используется для организации контроля доступа к другим сервисам, несанкционированный доступ к нему может создать предпосылки для атак на Вашу систему безопасности, которые, в свою очередь, могут привести к плачевным результатам.

Доступ к Вашему каталогу может быть сконфигурирован двумя методами, первый из которых использует Конфигурационный файл slapd, а второй использует формат *slapd-config*(5) (описанный в разделе Настройка slapd).

Политика контроля доступа по умолчанию - разрешить доступ на чтение всем клиентам. Независимо от того, какая политика контроля доступа определена, для *rootdn* всегда разрешены полные права (то есть аутентификация, поиск, сравнение, чтение и запись) над чем угодно и где угодно в каталоге.

Как следствие, явное указание *rootdn* среди условий *<by>* бесполезно и, в добавок, приводит к снижению производительности.

В следующих подразделах даётся достаточно глубокое описание списков контроля доступа (Access Control List), за которым следует несколько примеров и рекомендаций. Полное описание смотрите в *slapd.access*(5).

8.2. Контроль доступа при использовании статической конфигурации

Доступ к записям и атрибутам назначается директивой access конфигурационного файла. Общая форма строки access следующая:

```
<access directive> ::= access to <what>
    [by <who> [<access>] [<control>] ]+
<what> ::= * |
    [dn[.<basic-style>]=<regex> | dn.<scope-style>=<DN>]
    [filter=<ldapfilter>] [attrs=<attrlist>]
<basic-style> ::= regex | exact
<scope-style> ::= base | one | subtree | children
<attrlist> ::= <attr> [val[.<basic-style>]=<regex>] | <attr> , <attrlist>
<attr> ::= <attrname> | entry | children
<who> ::= * | [anonymous | users | self
        | dn[.<basic-style>]=<regex> | dn.<scope-style>=<DN>]
    [dnattr=<attrname>]
    [group[/<objectclass>[/<attrname>][.<basic-style>]]=<regex>]
    [peername[.<basic-style>]=<regex>]
    [sockname[.<basic-style>]=<regex>]
    [domain[.<basic-style>]=<regex>]
    [sockurl[.<basic-style>]=<regex>]
    [set=<setspec>]
    [aci=<attrname>]
<access> ::= [self]{<level>|<priv>}
<level> ::= none | disclose | auth | compare | search | read | write | manage
<priv> ::= {=|+|-}{m|w|r|s|c|x|d|0}+
<control> ::= [stop | continue | break]
```

где часть <what> определяет записи и/или атрибуты, к которым применяется доступ, часть `<who>` указывает, каким записям предоставляется доступ, и, наконец, часть `<access>` указывает предоставляемые полномочия доступа. Поддерживается указание нескольких триплетов `<who> <access> <control>`, что позволяет предоставить разным записям разный уровень доступа к одному и тому же набору записей и атрибутов. Здесь описаны не все опции контроля доступа, более детальное описание Вы найдёте в man-странице *slapd.access*(5).

8.2.1. Над чем осуществляется контроль доступа

Часть спецификации access <what> определяет записи и атрибуты, к которым применяется контроль доступа. Записи обычно выбираются двумя путями: по DN и по фильтру. В следующих критериях отбора описывается выбор записи по DN:

```
to *
to dn[.<basic-style>]=<regex>
to dn.<scope-style>=<DN>
```

В первой форме выбираются все записи. Вторая форма может быть использована для выбора записей, соответствующих регулярному выражению, а не *нормализованному DN* целевой записи (в этом документе не даётся развёрнутого обсуждения использования второй формы). Третья форма используется для выбора записей, попадающих в запрашиваемый диапазон DN. <DN> - это строка, представляющая собой уникальное имя (Distinguished Name), как описано в RFC4514.

Диапазон может задаваться как `base`, `one`, `subtree`, или `children`. Здесь `base` соответствует только записи с указанным DN, `one` соответствует записям, для которых указанный DN является родительским, `subtree` соответствует всем записям поддерева, корнем которого является указанный DN, и `children` соответствует всем записям ниже указанного DN (но не самой записи с указанным DN).

Например, если в каталоге есть записи со следующими именами:

```
0: o=suffix
1: cn=Manager,o=suffix
2: ou=people,o=suffix
3: uid=kdz,ou=people,o=suffix
4: cn=addresses,uid=kdz,ou=people,o=suffix
5: uid=hyc,ou=people,o=suffix
```

то:

`dn.base="ou=people,o=suffix"` соответствует строке 2;
`dn.one="ou=people,o=suffix"` соответствует строкам 3 и 5;
`dn.subtree="ou=people,o=suffix"` соответствует строкам 2, 3, 4 и 5; и
`dn.children="ou=people,o=suffix"` соответствует строкам 3, 4, и 5.

Записи также могут выбираться с использованием фильтра:

```
to filter=<ldap filter>
```

где <ldap filter> - это строка, представляющая собою поисковый фильтр LDAP, как описано в RFC4515. Например:

```
to filter=(objectClass=person)
```

Обратите внимание, что записи могут выбираться сразу и по DN и по фильтру путём включения обоих критериев отбора в условие <what>.

```
to dn.one="ou=people,o=suffix" filter=(objectClass=person)
```

Атрибуты в составе записи указываются путём включения разделённого запятыми списка имён атрибутов в условие <what>:

```
attrs=<attribute list>
```

Определённое значение атрибута выбирается путём использования одного имени атрибута и критерия отбора значений:

```
attrs=<attribute> val[.<style>]=<regex>
```

Есть два специальных *псевдо*-атрибута `entry` и `children`. Чтобы прочитать (и, следовательно, вернуть) целевую запись, субъект должен иметь права `read` на атрибут *entry* целевой записи. Чтобы осуществить поиск, субъект должен иметь права `search` на атрибут *entry* записи - базы поиска. Чтобы добавить или удалить запись, субъект должен иметь права `write` на атрибут `entry` этой записи, А ТАКЖЕ должен иметь права `write` на атрибут `children` родительской записи. Чтобы переименовать запись, субъект должен иметь права `write` на атрибут `entry` этой записи, А ТАКЖЕ иметь права `write` на атрибуты `children` как старой, так и новой родительских записей. В конце подраздела будут даны примеры, которые помогут Вам прояснить ситуацию.

И наконец, есть специальный критерий отбора записей `"*"`, который используется для выбора любой записи. Он используется, когда в условии `<what>` не было указано других критериев отбора. Это эквивалент `"dn=.*"`

8.2.2. Кому даются права на доступ

Часть спецификации access <who> определяет сущность или сущности, которым предоставляются права на доступ. Обратите внимание, что права предоставляются "сущностям", а не "записям". В следующей таблице приведены синтаксисы спецификации сущностей:

Таблица 8.1: Синтаксисы спецификации сущностей доступа

Синтаксис спецификации	**Описание сущности**

*	Все, включая анонимных пользователей и пользователей, прошедших аутентификацию
anonymous	Анонимные (не прошедшие аутентификацию) пользователи
users	Пользователи, прошедшие аутентификацию
self	Пользователь, ассоциированный с целевой записью
dn[.<basic-style>]=<regex>	Пользователи, записи которых соответствуют регулярному выражению
dn.<scope-style>=<DN>	Пользователи в диапазоне DN

Синтаксисы спецификации DN подчиняются тем же правилам, что и синтаксисы спецификации DN в условии <what>.

Также поддерживаются другие факторы контроля. Например, сущности в условии <who> могут назначаться как записи, перечисленные в атрибуте, значением которого являются DN, той записи, к которой применяется правило доступа:

```
dnattr=<имя атрибута, значением которого является DN>
```

Спецификация dnattr используется для предоставления доступа записям, DN которых перечислены в соответствующем атрибуте указанной записи (например, предоставление доступа к записи группы тем, кто перечислен как владельцы данной записи группы).

Некоторые факторы неуместно использовать в определённых (или даже во всех) окружениях. Например, фактор домена опирается на IP-адреса для разрешения доменных имён. Поскольку они легко подменяются, следует избегать использования фактора домена.

8.2.3. Какие права на доступ могут предоставляться

В условии <access>может указываться один из следующих типов прав доступа:

Таблица 8.2: Уровни доступа

Уровень	**Привилегии**	**Описание**
none =	0	нет доступа
disclose =	d	требуется для выдачи информации в случае ошибки
auth =	dx	требуется для аутентификации (подключения)
compare =	cdx	требуется для сравнения
search =	scdx	требуется для применения поисковых фильтров
read =	rscdx	требуется для чтения результатов поиска
write =	wrscdx	требуется для изменения/переименования
manage =	mwrscdx	требуется для осуществления управления

Каждый уровень включает в себя все более низкие уровни доступа. Таким образом, например, предоставление кому-нибудь доступа к какой-либо записи уровня write даёт ему также привилегии доступа read, search, compare, auth и disclose. Однако, их можно задать и явно для предоставления каких-нибудь специальных разрешений.

8.2.4. Принятие решения о предоставлении доступа

При принятии решения о предоставлении кому-либо, совершающему запрос, доступа к той или иной записи и/или атрибуту, slapd сравнивает запрашиваемые запись и/или атрибут с критериями отбора <what>, заданными в конфигурационном файле. Для каждой записи сначала применяются директивы контроля доступа, относящиеся к той базе данных, в которой эта запись содержится (или глобальные директивы, если эта запись не содержится ни в одной из баз данных), а затем применяются глобальные директивы контроля доступа. Существует некоторая тонкость при применении списков контроля доступа. Глобальные списки контроля доступа добавляются к спискам, определённым для каждой базы данных, и, если после такого сложения результирующий список оказывается непустым, тогда в его конец добавляется неявная директива access to *

by * none. Если же результирующий список пуст, то есть для какого-либо механизма манипуляции данными не задано директив, то по умолчанию предоставляется доступ на чтение.

Следуя этому приоритету, директивы access проверяются в порядке их указания в конфигурационном файле. Slapd останавливает проверку на первом критерии отбора <what>, который соответствует целевой записи и/или атрибуту. Директива access с этим критерием отбора и будет той, по которой slapd будет принимать решение о предоставлении доступа.

Затем slapd сравнивает сущность, запрашивающую доступ, с критериями отбора <who> выбранной выше директивы access в порядке их указания. Он останавливает сравнение на первом критерии отбора <who>, который соответствует целевой сущности. Так определяется, будет ли сущность, запрашивающая доступ, реально иметь доступ к записи и/или атрибуту.

Наконец, slapd сравнивает уровень доступа, предоставленный в выбранном условии <access>, с уровнем доступа, который запрашивает клиент. Если в условии разрешён больший или равный уровень доступа, то клиенту предоставляется доступ. В противном случае доступ запрещается.

Порядок оценки директив access делает важным порядок их размещения в конфигурационном файле. Если одна директива access выдвигает более конкретные условия отбора записей, чем другая, она должна быть указана раньше в конфигурационном файле. Аналогично, если один критерий отбора <who> выдвигает более конкретные условия отбора, нежели другой, он должен указываться раньше в директиве access. Приведённые ниже примеры настройки контроля доступа помогут прояснить ситуацию.

8.2.5. Примеры настройки контроля доступа

Инструмент контроля доступа, описанный выше, довольно мощный. Чтобы продемонстрировать это, приведём несколько примеров его использования.

Простой пример:

```
access to * by * read
```

Эта директива access предоставляет доступ на чтение всем.

```
access to *
    by self write
    by anonymous auth
    by * read
```

Эта директива позволяет пользователю изменять свою запись, позволяет анонимным пользователям производить аутентификацию по этим записям, наконец, позволяет всем остальным читать эти записи. Обратите внимание, что применяется только первое совпавшее условие by <who>. Поэтому анонимным пользователям предоставляются только права auth, а не read. Последнее же условие могло было быть "by users read".

Часто администратору требуется ограничивать операции, основываясь на том, соблюдается ли должный уровень защиты. Следующий пример показывает, как для этих целей можно использовать факторы силы безопасности (security strength factors, SSF):

```
access to *
    by ssf=128 self write
    by ssf=64 anonymous auth
    by ssf=64 users read
```

Эта директива позволяет пользователям изменять свои собственные записи, если при соединении применялись средства защиты силой 128 или более (то есть сессия шифровалась с ключом длиной не менее 128), позволяет производить аутентификацию анонимным пользователям и чтение пользователям, если при соединении применялись средства защиты силой 64 или более. Если клиент не обеспечил должного уровня защиты при соединении, применяется неявное условие by * none.

В следующем примере показано использование указателей стиля диапазона для выбора записей по DN. Приведены две директивы, для которых важен порядок их применения:

```
access to dn.children="dc=example,dc=com"
     by * search
```

```
access to dn.children="dc=com"
     by * read
```

Доступ на чтение предоставляется к дочерним записям поддерева dc=com, за исключением тех записей (являющихся дочерними для поддерева dc=example,dc=com), на которые предоставлены права на поиск. На саму запись dc=com не предоставляется никаких прав, поскольку нет директив, условия <what> которых совпадают с этим DN. Если поменять порядок этих директив access, то последняя из них никогда не будет применяться, поскольку все записи, дочерние для dc=example,dc=com также являются дочерними и для dc=com.

Также обратите внимание, что если не было совпадений ни с одной из директив access to или ни с одним из условий by <who>, **доступ будет запрещён**. То есть, каждая директива access to заканчивается неявным условием by * none. Глобальные списки контроля доступа добавляются к спискам, определённым для каждой базы данных, и, если после такого сложения результирующий список оказывается непустым, тогда в его конец добавляется неявная директива access to * by * none. Если же результирующий список пуст, то есть для какого-либо механизма манипуляции данными не задано директив, то по умолчанию предоставляется доступ на чтение.

В следующем примере снова показана важность порядка указания как директив access, так и условий by <who>. В нём также показано использование критерия отбора атрибутов для предоставления доступа к указанному атрибуту, а также различные критерии отбора в условиях <who>.

```
access to dn.subtree="dc=example,dc=com" attrs=homePhone
    by self write
    by dn.children="dc=example,dc=com" search
    by peername.regex=IP:10\..+ read
access to dn.subtree="dc=example,dc=com"
    by self write
    by dn.children="dc=example,dc=com" search
    by anonymous auth
```

В этом примере контроль доступа осуществляется над записями поддерева "dc=example,dc=com". Ко всем атрибутам, за исключением homePhone, у пользователя есть право изменять свою запись, у записей, дочерних к example.com есть право производить поиск по этим записям, больше прав никому не даётся (неявное условие by * none), за исключением аутентификации/авторизации (которые всегда производятся анонимным пользователем). Атрибут homePhone доступен для изменения самой записью, для поиска - записям, дочерним к example.com, для чтения - клиентам, подключающимся из сети 10, в остальных случаях доступа нет (неявное условие by * none). Кроме того, остальные попытки доступа запрещены неявной директивой access to * by * none.

Иногда бывает полезно разрешить некоторым DN добавлять или удалять самих себя из атрибута. Например, если Вам нужно создать группу и разрешить людям добавлять или удалять только свои собственные DN из атрибута member, Вы можете сделать это с помощью такой директивы access:

```
access to attrs=member,entry
     by dnattr=member selfwrite
```

Критерий отбора dnattr в условии <who> говорит о том, что доступ осуществляется к записям, перечисленным в атрибуте member. Уровень доступа selfwrite говорит о том, что записи, перечисленные в атрибуте member, могут добавлять или удалять только свои собственные DN из этого атрибута, и ни какие другие значения. Добавление атрибута entry необходимо, поскольку доступ к записи требуется для доступа к любому атрибуту этой записи.

8.3. Контроль доступа при использовании динамической конфигурации

Доступ к записям и атрибутам slapd контролируется атрибутом olcAccess, значения которого - это последовательность директив access. Общая форма конфигурации olcAccess следующая:

```
olcAccess: <access directive>
<access directive> ::= to <what>
    [by <who> [<access>] [<control>] ]+
<what> ::= * |
    [dn[.<basic-style>]=<regex> | dn.<scope-style>=<DN>]
    [filter=<ldapfilter>] [attrs=<attrlist>]
<basic-style> ::= regex | exact
```

```
<scope-style> ::= base | one | subtree | children
<attrlist> ::= <attr> [val[.<basic-style>]=<regex>] | <attr> , <attrlist>
<attr> ::= <attrname> | entry | children
<who> ::= * | [anonymous | users | self
        | dn[.<basic-style>]=<regex> | dn.<scope-style>=<DN>]
    [dnattr=<attrname>]
    [group[/<objectclass>[/<attrname>][.<basic-style>]]=<regex>]
    [peername[.<basic-style>]=<regex>]
    [sockname[.<basic-style>]=<regex>]
    [domain[.<basic-style>]=<regex>]
    [sockurl[.<basic-style>]=<regex>]
    [set=<setspec>]
    [aci=<attrname>]
<access> ::= [self]{<level>|<priv>}
<level> ::= none | disclose | auth | compare | search | read | write | manage
<priv> ::= {=|+|-}{m|w|r|s|c|x|d|0}+
<control> ::= [stop | continue | break]
```

где часть <what> определяет записи и/или атрибуты, к которым применяется доступ, часть `<who>` указывает, каким записям предоставляется доступ, и, наконец, часть `<access>` указывает предоставляемые полномочия доступа. Поддерживается указание нескольких триплетов `<who> <access> <control>`, что позволяет предоставить разным записям разный уровень доступа к одному и тому же набору записей и атрибутов. Здесь описаны не все опции контроля доступа, более детальное описание Вы найдёте в man-странице *slapd.access*(5).

8.3.1. Над чем осуществляется контроль доступа

Часть спецификации access <what> определяет записи и атрибуты, к которым применяется контроль доступа. Записи обычно выбираются двумя путями: по DN и по фильтру. В следующих критериях отбора описывается выбор записи по DN:

```
to *
to dn[.<basic-style>]=<regex>
to dn.<scope-style>=<DN>
```

В первой форме выбираются все записи. Вторая форма может быть использована для выбора записей, соответствующих регулярному выражению а не *нормализованному DN* целевой записи (в этом документе не даётся развёрнутого обсуждения использования второй формы). Третья форма используется для выбора записей, попадающих в запрашиваемый диапазон DN. <DN> - это строка, представляющая собой уникальное имя (Distinguished Name), как описано в RFC4514.

Диапазон может задаваться как `base`, `one`, `subtree`, или `children`. Здесь `base` соответствует только записи с указанным DN, `one` соответствует записям, для которых указанный DN является родительским, `subtree` соответствует всем записям поддерева, корнем которого является указанный DN, и `children` соответствует всем записям ниже указанного DN (но не самой записи с указанным DN).

Например, если в каталоге есть записи со следующими именами:

```
0: o=suffix
1: cn=Manager,o=suffix
2: ou=people,o=suffix
3: uid=kdz,ou=people,o=suffix
4: cn=addresses,uid=kdz,ou=people,o=suffix
5: uid=hyc,ou=people,o=suffix
```

то:

`dn.base="ou=people,o=suffix"` соответствует строке 2;
`dn.one="ou=people,o=suffix"` соответствует строкам 3 и 5;
`dn.subtree="ou=people,o=suffix"` соответствует строкам 2, 3, 4 и 5; и
`dn.children="ou=people,o=suffix"` соответствует строкам 3, 4, и 5.

Записи также могут выбираться с использованием фильтра:

```
to filter=<ldap filter>
```

где <ldap filter> - это строка, представляющая собою поисковый фильтр LDAP, как описано в RFC4515. Например:

```
to filter=(objectClass=person)
```

Обратите внимание, что записи могут выбираться сразу и по DN, и по фильтру путём включения обоих критериев отбора в условие <what>.

```
to dn.one="ou=people,o=suffix" filter=(objectClass=person)
```

Атрибуты в составе записи указываются путём включения разделённого запятыми списка имён атрибутов в условие <what>:

```
attrs=<attribute list>
```

Определённое значение атрибута выбирается путём использования одного имени атрибута и критерия отбора значений:

```
attrs=<attribute> val[.<style>]=<regex>
```

Есть два специальных *псевдо*-атрибута `entry` и `children`. Чтобы прочитать (и, следовательно, вернуть) целевую запись, субъект должен иметь права `read` на атрибут *entry* целевой записи. Чтобы осуществить поиск, субъект должен иметь права `search` на атрибут *entry* записи - базы поиска. Чтобы добавить или удалить запись, субъект должен иметь права `write` на атрибут `entry` этой записи, А ТАКЖЕ должен иметь права `write` на атрибут `children` родительской записи. Чтобы переименовать запись, субъект должен иметь права `write` на атрибут `entry` этой записи, А ТАКЖЕ иметь права `write` на атрибуты `children` как старой, так и новой родительских записей. В конце подраздела будут даны примеры, которые помогут Вам прояснить ситуацию.

И наконец, есть специальный критерий отбора записей `"*"`, который используется для выбора любой записи. Он используется, когда в условии `<what>` не было указано других критериев отбора. Это эквивалент `"dn=.*"`

8.3.2. Кому даются права на доступ

Часть спецификации access <who> определяет сущность или сущности, которым предоставляются права на доступ. Обратите внимание, что права предоставляются "сущностям", а не "записям". В следующей таблице приведены синтаксисы спецификации сущностей:

Table 8.3: Синтаксисы спецификации сущностей доступа

Синтаксис спецификации	**Описание сущности**
`*`	Все, включая анонимных пользователей и пользователей, прошедших аутентификацию
`anonymous`	Анонимные (не прошедшие аутентификацию) пользователи
`users`	Пользователи, прошедшие аутентификацию
`self`	Пользователь, ассоциированный с целевой записью
`dn[.<basic-style>]=<regex>`	Пользователи, записи которых соответствуют регулярному выражению
`dn.<scope-style>=<DN>`	Пользователи в диапазоне DN

Синтаксисы спецификации DN подчиняются тем же правилам, что и синтаксисы спецификации DN в условии <what>.

Также поддерживаются другие факторы контроля. Например, сущности в условии `<who>` могут назначаться как записи, перечисленные в атрибуте, значением которого являются DN, той записи, к которой применяется правило доступа:

```
dnattr=<имя атрибута, значением которого является DN>
```

Спецификация dnattr используется для предоставления доступа записям, DN которых перечислены в соответствующем атрибуте указанной записи (например, предоставление доступа к записи группы тем, кто перечислен как владельцы данной записи группы).

Некоторые факторы неуместно использовать в определённых (или даже во всех) окружениях. Например, фактор домена опирается на IP-адреса для разрешения доменных имён. Поскольку они легко подменяются, следует избегать использования фактора домена.

8.3.3. Какие права на доступ могут предоставляться

В условии <access>может указываться один из следующих типов прав доступа:

Table 8.4: Уровни доступа

Уровень	Привилегии	Описание
none	=0	нет доступа
disclose	=d	требуется для выдачи информации в случае ошибки
auth	=dx	требуется для аутентификации (подключения)
compare	=cdx	требуется для сравнения
search	=scdx	требуется для применения поисковых фильтров
read	=rscdx	требуется для чтения результатов поиска
write	=wrscdx	требуется для изменения/переименования
manage	=mwrscdx	требуется для осуществления управления

Каждый уровень включает в себя все более низкие уровни доступа. Таким образом, например, предоставление кому-нибудь доступа к какой-либо записи уровня `write` даёт ему также привилегии доступа `read`, `search`, `compare`, `auth` и `disclose`. Однако, их можно задать и явно для предоставления каких-нибудь специальных разрешений.

8.3.4. Принятие решения о предоставлении доступа

При принятии решения о предоставлении кому-либо, совершающему запрос, доступа к той или иной записи и/или атрибуту, slapd сравнивает запрашиваемые запись и/или атрибут с критериями отбора `<what>`, заданными при конфигурации. Для каждой записи сначала применяются директивы контроля доступа, относящиеся к той базе данных, в которой эта запись содержится (или глобальные директивы, если эта запись не содержится ни в одной из баз данных), а затем применяются глобальные директивы контроля доступа (которые содержатся в определении базы данных `frontend`). Существует некоторая тонкость при применении списков контроля доступа. Глобальные списки контроля доступа добавляются к спискам, определённым для каждой базы данных, и, если после такого сложения результирующий список оказывается непустым, тогда в его конец добавляется неявная директива `access to * by * none`. Если же результирующий список пуст, то есть для какого-либо механизма манипуляции данными не задано директив, то по умолчанию предоставляется доступ на чтение.

Следуя этому приоритету, директивы access проверяются в порядке их указания в конфигурационном атрибуте. Slapd останавливает проверку на первом критерии отбора `<what>`, который соответствует целевой записи и/или атрибуту. Директива access с этим критерием отбора и будет той, по которой slapd будет принимать решение о предоставлении доступа.

Затем slapd сравнивает сущность, запрашивающую доступ, с критериями отбора `<who>` выбранной выше директивы access в порядке их указания. Он останавливает сравнение на первом критерии отбора `<who>`, который соответствует целевой сущности. Так определяется, будет ли сущность, запрашивающая доступ, реально иметь доступ к записи и/или атрибуту.

Наконец, slapd сравнивает уровень доступа, предоставленный в выбранном условии `<access>`, с уровнем доступа, который запрашивает клиент. Если в условии разрешён больший или равный уровень доступа, то клиенту предоставляется доступ. В противном случае доступ запрещается.

Порядок оценки директив access делает важным порядок их размещения в конфигурационном файле. Если одна директива access выдвигает более конкретные условия отбора записей, чем другая, она должна быть указана раньше при конфигурации. Аналогично, если один критерий отбора `<who>` выдвигает более конкретные условия отбора, нежели другой, он должен указываться раньше в директиве access. Приведённые ниже примеры настройки контроля доступа помогут прояснить ситуацию.

8.3.5. Примеры настройки контроля доступа

Инструмент контроля доступа, описанный выше, довольно мощен. Чтобы продемонстрировать это, приведём несколько примеров его использования.

Простой пример:

```
olcAccess: to * by * read
```

Эта директива access предоставляет доступ на чтение всем.

```
olcAccess: to *
    by self write
    by anonymous auth
    by * read
```

Эта директива позволяет пользователю изменять свою запись, позволяет анонимным пользователям производить аутентификацию по этим записям, наконец, позволяет всем остальным читать эти записи. Обратите внимание, что применяется только первое совпавшее условие `by <who>`. Поэтому анонимным пользователям предоставляются только права `auth`, а не `read`. Последнее же условие могло было быть "`by users read`".

Часто администратору требуется ограничивать операции, основываясь на том, соблюдается ли должный уровень защиты. Следующий пример показывает, как для этих целей можно использовать факторы силы безопасности (security strength factors, SSF):

```
olcAccess: to *
    by ssf=128 self write
    by ssf=64 anonymous auth
    by ssf=64 users read
```

Эта директива позволяет пользователям изменять свои собственные записи, если при соединении применялись средства защиты силой 128 или более (то есть сессия шифровалась с ключом длиной не менее 128), позволяет производить аутентификацию анонимным пользователям и чтение пользователям, если при соединении применялись средства защиты силой 64 или более. Если клиент не обеспечил должного уровня защиты при соединении, применяется неявное условие `by * none`.

В следующем примере показано использование указателей стиля диапазона для выбора записей по DN. Приведены две директивы, для которых важен порядок их применения:

```
olcAccess: to dn.children="dc=example,dc=com"
     by * search
olcAccess: to dn.children="dc=com"
     by * read
```

Доступ на чтение предоставляется к дочерним записям поддерева `dc=com`, за исключением тех записей (являющихся дочерними для поддерева `dc=example,dc=com`), на которые предоставлены права на поиск. На саму запись `dc=com` не предоставляется никаких прав, поскольку нет директив, условия <what> которых совпадают с этим DN. Если поменять порядок этих директив access, то последняя из них никогда не будет применяться, поскольку все записи, дочерние для `dc=example,dc=com` также являются дочерними и для `dc=com`.

Также обратите внимание, что если не было совпадений ни с одной из директив `olcAccess: to` или ни с одним из условий `by <who>`, **доступ будет запрещён**. Глобальные списки контроля доступа добавляются к спискам, определённым для каждой базы данных, и, если после такого сложения результирующий список оказывается непустым, тогда в его конец добавляется неявная директива `access to * by * none`. Если же результирующий список пуст, то есть для какого-либо механизма манипуляции данными не задано директив, то по умолчанию предоставляется доступ на чтение.

В следующем примере снова показана важность порядка указания как директив access, так и условий `by <who>`. В нём также показано использование критерия отбора атрибутов для предоставления доступа к указанному атрибуту, а также различные критерии отбора в условиях `<who>`.

```
olcAccess: to dn.subtree="dc=example,dc=com" attrs=homePhone
    by self write
    by dn.children=dc=example,dc=com" search
    by peername.regex=IP:10\..+ read
olcAccess: to dn.subtree="dc=example,dc=com"
    by self write
    by dn.children="dc=example,dc=com" search
    by anonymous auth
```

В этом примере контроль доступа осуществляется над записями поддерева "`dc=example,dc=com`". Ко всем

атрибутам, за исключением homePhone, у пользователя есть право изменять свою запись, у записей, дочерних к example.com есть право производить поиск по этим записям, больше прав никому не даётся (неявное условие by * none), за исключением аутентификации/авторизации (которые всегда производятся анонимным пользователем). Атрибут homePhone доступен для изменения самой записью, для поиска - записям, дочерним к example.com, для чтения - клиентам, подключающимся из сети 10, в остальных случаях доступа нет (неявное условие by * none). Кроме того, остальные попытки доступа запрещены неявной директивой access to * by * none.

Иногда бывает полезно разрешить некоторым DN добавлять или удалять самих себя из атрибута. Например, если Вам нужно создать группу и разрешить людям добавлять или удалять только свои собственные DN из атрибута member, Вы можете сделать это с помощью такой директивы access:

```
olcAccess: to attrs=member,entry
     by dnattr=member selfwrite
```

Критерий отбора dnattr в условии <who> говорит о том, что доступ осуществляется к записям, перечисленным в атрибуте member. Уровень доступа selfwrite говорит о том, что записи, перечисленные в атрибуте member, могут добавлять или удалять только свои собственные DN из этого атрибута, и ни какие другие значения. Добавление атрибута entry необходимо, поскольку доступ к записи требуется для доступа к любому атрибуту этой записи.

8.3.6. Порядок применения контроля доступа

Поскольку порядок директив olcAccess имеет важное значение для принятия по ним правильного решения, а атрибуты LDAP по умолчанию не заботятся о сохранении порядка своих значений, OpenLDAP использует обычное расширение схемы данных для поддержания фиксированного порядка этих значений. Упорядочение устанавливается добавлением числового индекса "{X}" перед каждым значением атрибута, аналогично тому, как упорядочиваются записи конфигурации. Эти метки индексов автоматически создаются slapd, их не надо указывать при первоначальном определении значений. Например, когда Вы создаёте настройки

```
olcAccess: to attrs=member,entry
     by dnattr=member selfwrite
olcAccess: to dn.children="dc=example,dc=com"
     by * search
olcAccess: to dn.children="dc=com"
     by * read
```

, то при их извлечении с помощью slapcat или ldapsearch, в них будет содержаться:

```
olcAccess: {0}to attrs=member,entry
     by dnattr=member selfwrite
olcAccess: {1}to dn.children="dc=example,dc=com"
     by * search
olcAccess: {2}to dn.children="dc=com"
     by * read
```

Числовой индекс может быть использован для указания того, какое конкретно значение Вы хотите изменить при редактировании правил контроля доступа с помощью утилиты ldapmodify. Этот индекс может быть использован вместо (или в дополнение к) фактическому значению атрибута olcAccess. Использование числового индекса может очень помочь, если требуется управлять сразу несколькими правилами контроля доступа.

Например, Вам нужно изменить второе правило из приведённого выше примера, чтобы предоставить доступ на запись вместо доступа на выполнение поиска. Можно попробовать использовать такой LDIF:

```
changetype: modify
delete: olcAccess
olcAccess: to dn.children="dc=example,dc=com" by * search
-
add: olcAccess
olcAccess: to dn.children="dc=example,dc=com" by * write
-
```

Но данный пример **не** гарантирует, что существующие значения останутся в их первоначальном порядке, поэтому Вы наверняка получите неверные настройки безопасности. Чтобы избежать подобной ситуации, следует использовать числовой индекс:

```
changetype: modify
```

```
delete: olcAccess
olcAccess: {1}
-
add: olcAccess
olcAccess: {1}to dn.children="dc=example,dc=com" by * write
-
```

В этом примере удаляется любое правило, содержащееся в значении номер 1 атрибута `olcAccess` (независимо от содержимого этого значения) и добавляется новое значение, которое явно вставлено как значение номер 1. В результате получаем:

```
olcAccess: {0}to attrs=member,entry
     by dnattr=member selfwrite
olcAccess: {1}to dn.children="dc=example,dc=com"
     by * write
olcAccess: {2}to dn.children="dc=com"
     by * read
```

, а это именно то, что было задумано.

8.4. Типичные примеры контроля доступа

8.4.1. Основные ACL

В общем случае нужно начать с некоторых основных ACL, таких как:

```
access to attr=userPassword
    by self =xw
    by anonymous auth
    by * none


  access to *
    by self write
    by users read
    by * none
```

Первый ACL позволяет пользователям обновлять (но не читать) свои пароли, анонимным пользователям проходить аутентификацию с использованием этого атрибута, и (неявно) запрещает любой доступ всем остальным.

Второй ACL позволяет пользователям полный доступ к своим записям, пользователям, прошедшим аутентификацию, - доступ на чтение ко всему, и (неявно) запрещает любой доступ всем остальным (в данном случае, анонимным пользователям).

8.4.2. Соответствие анонимным пользователям и пользователям, прошедшим аутентификацию

Анонимный пользователь имеет пустой DN. Хотя для обозначения анонимного пользователя можно использовать конструкции *dn.exact=""* или *dn.regex="^$"*, *slapd*(8) предлагает использовать вместо них термин anonymous.

```
access to *
  by anonymous none
  by * read
```

Здесь полностью запрещён доступ анонимным пользователям, остальным пользователям предоставляются права на чтение.

Пользователи, прошедшие аутентификацию, имеют предметные DN. Хотя для обозначения любого пользователя, прошедшего аутентификацию, можно использовать конструкцию *dn.regex=".+"*, OpenLDAP предлагает использовать вместо неё термин users.

```
access to *
  by users read
  by * none
```

Этот ACL предоставляет права на чтение пользователям, прошедшим аутентификацию, и запрещает доступ всем остальным (то есть анонимным пользователям).

8.4.3. Контроль доступа для rootdn

Вы можете задать *rootdn* в *slapd.conf*(5) или в *slapd.d* без указания *rootpw*. Затем Вы должны добавить актуальную запись каталога с тем же самым DN, например:

```
dn: cn=Manager,o=MyOrganization
cn: Manager
sn: Manager
objectClass: person
objectClass: top
userPassword: {SSHA}someSSHAdata
```

После этого для подключения под учётной записью *rootdn* будет необходимо произвести стандартную процедуру подключения для заданного DN, которая в свою очередь, требует прав доступа auth к записи с этим DN и атрибуту *userPassword*, и эти права можно ограничить с помощью ACL. Например:

```
access to dn.base="cn=Manager,o=MyOrganization"
  by peername.regex=127\.0\.0\.1 auth
  by peername.regex=192\.168\.0\..* auth
  by users none
  by * none
```

ACL, приведённые выше, разрешают подключение под учётной записью rootdn только с localhost и из сети 192.168.0.0/24.

8.4.4. Управление доступом с помощью групп

Это можно сделать несколькими способами. Покажем здесь один из них. Рассмотрим следующий макет DIT:

```
+-dc=example,dc=com
+---cn=administrators,dc=example,dc=com
+---cn=fred blogs,dc=example,dc=com
```

и следующий объект группы (в формате LDIF):

```
dn: cn=administrators,dc=example,dc=com
cn: administrators of this region
objectclass: groupOfNames  (важно для применения группового acl)
member: cn=fred blogs,dc=example,dc=com
member: cn=somebody else,dc=example,dc=com
```

Теперь можно предоставить доступ членам этой группы, добавив соответствующие условия *by group* в директиве access в *slapd.conf*(5). Например:

```
access to dn.children="dc=example,dc=com"
    by self write
    by group.exact="cn=Administrators,dc=example,dc=com" write
    by * auth
```

Как и в условиях *dn*, можно также использовать ключевое слово *expand*, чтобы расширить имя группы, основываясь на совпадении, найденном в результате применения регулярного выражения к целевой записи (с помощью *dn.regex*). Например,

```
access to dn.regex="(.+,)?ou=People,(dc=[^,]+,dc=[^,]+)$"
         attrs=children,entry,uid
    by group.expand="cn=Managers,$2" write
    by users read
    by * auth
```

В нашем примере предполагается, что члены группы находятся в атрибуте типа *member* объектного класса *groupOfNames*. Если Вам нужно использовать для группы другой объектный класс и/или другой тип атрибута, то воспользуйтесь следующим (сокращённым) синтаксисом в *slapd.conf*(5):

```
access to <what>
        by group/<объектный класс>/<имя атрибута>=<DN> <access>
```

Например:

```
access to *
  by group/organizationalRole/roleOccupant="cn=Administrator,dc=example,dc=com" write
```

В этом случае используется объектный класс *organizationalRole*, содержащий DN администратора в атрибуте *roleOccupant*. Например:

```
dn: cn=Administrator,dc=example,dc=com
cn: Administrator
objectclass: organizationalRole
roleOccupant: cn=Jane Doe,dc=example,dc=com
```

Примечание: Указанный для членства в группе тип атрибута ДОЛЖЕН ИМЕТЬ синтаксис DN или *NameAndOptionalUID*, а указанный объектный класс ДОЛЖЕН ПОЗВОЛЯТЬ использование этого типа атрибута.

При контроле доступа также могут применяться динамические группы. Узнать о них можно в *slapo-dynlist*(5) и в подразделе о наложении Динамические списки.

8.4.5. Предоставление доступа к подмножеству атрибутов

Вы можете предоставить доступ к набору атрибутов, указав список имен атрибутов в условии ACL *to*. Чтобы это работало, Вам необходимо также предоставить доступ к самой записи с помощью псевдо-атрибута *entry*. Также обратите внимание на то, как псевдо-атрибут *children* позволяет управлять возможностью добавлять, удалять и переименовывать записи.

```
# mail: пользователь, ассоциированный с самой записью, имеет право на изменение,
# пользователи, прошедшие аутентификацию, имеют право на чтение
access to attrs=mail
  by self write
  by users read
  by * none

# cn, sn: пользователь, ассоциированный с самой записью, имеет право на изменение,
# все остальные имеют право на чтение
access to attrs=cn,sn
  by self write
  by * read

# непосредственные подзаписи (children): только пользователь, ассоциированный с самой записью,
# может добавлять/удалять записи, непосредственно дочерние данной записи
access to attrs=children
  by self write

# сама запись (entry): пользователь, ассоциированный с самой записью, имеет право на изменение,
# все остальные имеют право на чтение
access to attrs=entry
  by self write
  by * read

# другие атрибуты: пользователь, ассоциированный с самой записью, имеет право на изменение,
# всем остальным доступ запрещён
access to *
  by self write
  by * none
```

В списке атрибутов можно также указать имена объектных классов, в этом случае права доступа назначаются всем атрибутам, которые требуются и/или разрешены данным объектным классом. Имена в списке *attrlist*, начинающиеся с *@*, сразу рассматриваются как имена объектных классов. Если же имя начинается с *!*, оно также рассматривается как имя объектного класса, но в этом случае права доступа назначаются всем атрибутам, которые не требуются и не разрешены данным объектным классом.

8.4.6. Разрешение пользователю изменять все записи, находящиеся ниже его собственной

Дадим права пользователю изменять свою собственную запись и все записи-потомки:

```
access to dn.regex="(.+,)?(uid=[^,]+,o=Company)$"
   by dn.exact,expand="$2" write
   by anonymous auth
```

(Позже сюда будут добавлены дополнительные примеры).

8.4.7. Разрешение на создание записи

Предположим, у Вас есть такая структура:

```
o=<basedn>
    ou=domains
        associatedDomain=<somedomain>
            ou=users
                uid=<someuserid>
                uid=<someotheruserid>
            ou=addressbooks
                uid=<someuserid>
                    cn=<someone>
                    cn=<someoneelse>
```

и, для другого домена <someotherdomain>:

```
o=<basedn>
    ou=domains
        associatedDomain=<someotherdomain>
            ou=users
                uid=<someuserid>
                uid=<someotheruserid>
            ou=addressbooks
                uid=<someotheruserid>
                    cn=<someone>
                    cn=<someoneelse>
```

Теперь, если Вам нужно позволить пользователю *uid=<someuserid>* создавать записи **ТОЛЬКО** в пределах своей собственной области, можно написать примерно такой ACL:

```
# данное правило позволяет пользователям из "associatedDomain=<совпавший домен>"
# вносить изменения ниже "ou=addressbook,associatedDomain=<совпавший домен>,ou=domains,o=<basedn>",
# то есть пользователь может изменять ЛЮБУЮ запись в адресной книге своего домена;
# это разрешение необходимое, но не достаточное, следующее
# разрешение дополнительно ограничит данное разрешение


access to dn.regex="^ou=addressbook,associatedDomain=([^,]+),ou=domains,o=<basedn>$" attrs=children
        by dn.regex="^uid=([^,]+),ou=users,associatedDomain=$1,ou=domains,o=<basedn>$$" write
        by * none


# Обратите внимание, что приведённое выше условие "by" должно быть в стиле "regex"
# чтобы убедиться, что оно распространяется на DN, начинающиеся с шаблона "uid=<someuserid>"
# с заменой associatedDomain совпавшей подстрокой из условия "what".


# А это правило позволяет пользователям "uid=<совпавший uid>" of "associatedDomain=<совпавший домен>"
# вносить изменения (то есть добавлять, модифицировать, удалять) запись с DN, точно соответствующим
# "uid=<совпавший uid>,ou=addressbook,associatedDomain=<совпавший домен>,ou=domains,o=<basedn>"
# и ЛЮБУЮ запись-потомок этой записи


access to dn.regex="^(.+,)?uid=([^,]+),ou=addressbook,associatedDomain=([^,]+),ou=domains,o=<basedn>$"
        by dn.exact,expand="uid=$2,ou=users,associatedDomain=$3,ou=domains,o=<basedn>" write
        by * none


# Обратите внимание, что приведённое выше условие "by" использует стиль "exact"
# с модификатором "expand", поскольку теперь всё условие может быть построено с помощью
# совпавших подстрок из условия "what", следовательно компиляция и сопоставление "regex"
# больше не требуется.
```

8.4.8. Советы по использованию регулярных выражений при контроле доступа

Всегда указывайте *dn.regex=<шаблон>*, когда Вы собираетесь использовать регулярные выражения. Если указать только *dn=<шаблон>*, то по умолчанию он будет соответствовать *dn.exact<шаблон>*.

Если Вам нужно, чтобы было найдено совпадение хотя бы с одним символом, используйте *(.+)* вместо *(.*)*. *(.*)* может совпасть и с пустой строкой.

Не используйте регулярные выражения для совпадений, которые могут быть найдены более безопасным и менее затратным способом. Примеры:

```
dn.regex=".*dc=example,dc=com"
```

небезопасно и затратно:

- небезопасно, поскольку будет найдено совпадение с любой строкой, содержащей *dc=example,dc=com*, а не только заканчивающейся на запрошенный шаблон; вместо этого используйте *.*dc=example,dc=com$*.
- также небезопасно, поскольку название типа атрибута в первом RDN строки запрашиваемого шаблона будет соответствовать любому *типу атрибута*, заканчивающемуся на *dc*, например, пользовательскому типу атрибута *mydc*. Если Вам действительно нужно регулярное выражение, позволяющее находить *dc=example,dc=com* или любой из его потомков, используйте *^(.+,)?dc=example,dc=com$*, что означает: что-нибудь слева от dc=..., если оно вообще существует (смотрите вопросительный знак после шаблона в скобках), должно заканчиваться запятой;
- затратно, поскольку, если вам впоследствии не понадобятся совпавшие подстроки, можно использовать соответствия в стиле диапазона, например:

```
dn.subtree="dc=example,dc=com"
```

чтобы включить *dc=example,dc=com* в шаблон для поиска совпадения,

```
dn.children="dc=example,dc=com"
```

чтобы исключить *dc=example,dc=com* из шаблона для поиска совпадения, или

```
dn.onelevel="dc=example,dc=com"
```

для соответствия только одному подуровню.

Всегда, когда это уместно, используйте *^* и *$* в регулярных выражениях, поскольку без них *ou=(.+),ou=(.+),ou=addressbooks,o=basedn* будет совпадать с *something=bla,ou=xxx,ou=yyy,ou=addressbooks,o=basedn,ou=addressbooks,o=basedn,dc=some,dc=org*

Всегда используйте *([^,]+)* чтобы указать соответствие точно одному RDN, поскольку *(.+)* может включать любое количество RDN; например, *ou=(.+),dc=example,dc=com* будет совпадать с *ou=My,o=Org,dc=example,dc=com*, а это, возможно, не то, что Вы ожидали.

Никогда не добавляйте rootdn в условия <by>. Во время операций, выполняемых от имени идентификационной сущности rootdn, ACL даже не обрабатываются (в противном случае, не было бы никакого смысла вообще определять rootdn).

Используйте термины. Термин users совпадает с пользователями, прошедшими аутентификацию, а термин anonymous совпадает с анонимными пользователями.

Не используйте форму *dn.regex* в условиях <by>, если Вам нужно только найти совпадение с диапазоном и/или заменить подстроку; используйте соответствия в стиле диапазона (например *exact*, *onelevel*, *children* или *subtree*) и модификатор стиля expand, чтобы указать расширение подстроки.

Например,

```
access to dn.regex=".+,dc=([^,]+),dc=([^,]+)$"
  by dn.regex="^[^,],ou=Admin,dc=$1,dc=$2$$" write
```

хотя и верно, но может быть заменено на более безопасный и эффективный вариант

```
access to dn.regex=".+,(dc=[^,]+,dc=[^,]+)$"
  by dn.onelevel,expand="ou=Admin,$1" write
```

где регулярное выражение в условии *<what>* является более компактным, а регулярное выражение в условии *<by>* заменено на гораздо более эффективное соответствие в стиле диапазона onelevel с расширением подстроки.

8.4.9. Предоставление и запрет доступа на основе факторов силы безопасности (ssf)

Вы можете ограничивать доступ, основываясь на факторах силы безопасности (SSF)

```
access to dn="cn=example,cn=edu"
      by * ssf=256 read
```

0 (ноль) подразумевает, что защиты не требуется, 1 подразумевает только защиту целостности данных, 56 - требование использовать DES или другие слабые шифры, 112 - требование использовать triple DES или другие сильные шифры, 128 - требование использовать RC4, Blowfish или другие современные сильные шифры.

Другие возможности:

```
transport_ssf=<n>
tls_ssf=<n>
sasl_ssf=<n>
```

Рекомендуется требования уровня безопасности 256.

Информацию по *ssf* можно найти в *slapd.conf*(5).

8.4.10. Если что-то работает не так, как ожидалось

Рассмотрим следующий пример:

```
access to *
  by anonymous auth

access to *
  by self write

access to *
  by users read
```

Вам может показаться, что данные правила позволят любому пользователю пройти аутентификацию, прочитать любые записи из каталога и изменить свои данные, если аутентификация пройдена. Однако в этом примере будет работать только аутентификация, а ldapsearch никогда не вернёт данных. Проблема в том, что SLAPD применяет настройки доступа последовательно, строка за строкой, и останавливается на первой совпавшей части правил доступа (в данном случае: *to **).

Чтобы получить то, что мы хотели, файл должен выглядеть следующим образом:

```
access to *
  by anonymous auth
  by self write
  by users read
```

Основное правило: "сначала определяются более конкретные правила, а в конце - более общие".

Смотрите также *slapd.access*(5), а для поиска ошибок - loglevel 128 и *slapacl*(8).

8.5. Наборы - предоставление прав на основе взаимоотношений

Применение наборов лучше всего показать на примерах. В следующих подразделах представлено несколько примеров ACL для облегчения понимания их использования.

FAQ по применению наборов при контроле доступа: http://www.openldap.org/faq/data/cache/1133.html.

Примечание: Применение наборов пока считается экспериментальным.

8.5.1. Группы групп

Групповые ACL в OpenLDAP не имеют расширения для групп, содержащих группы, то есть групп, имеющих в качестве члена другую группу. Например:

```
dn: cn=sudoadm,ou=group,dc=example,dc=com
cn: sudoadm
```

```
objectClass: groupOfNames
member: uid=john,ou=people,dc=example,dc=com
member: cn=accountadm,ou=group,dc=example,dc=com

dn: cn=accountadm,ou=group,dc=example,dc=com
cn: accountadm
objectClass: groupOfNames
member: uid=mary,ou=people,dc=example,dc=com
```

Если использовать стандартные групповые ACL с записями из примера выше и разрешить членам группы `sudoadm` писать куда-либо, запись `mary` не будет включена:

```
access to dn.subtree="ou=sudoers,dc=example,dc=com"
        by group.exact="cn=sudoadm,ou=group,dc=example,dc=com" write
        by * read
```

При использовании наборов мы можем сделать ACL рекурсивным и указать ему рассматривать группу внутри группы. Таким образом, для каждого члена внешней группы будет происходить дальнейшее расширение:

```
access to dn.subtree="ou=sudoers,dc=example,dc=com"
      by set="[cn=sudoadm,ou=group,dc=example,dc=com]/member* & user" write
      by * read
```

Данный ACL с набором (set) означает: взять DN `cn=sudoadm`, проверить его атрибут (атрибуты) `member` (где "`*`" означает рекурсивную проверку) и найти пересечение результата этой проверки с DN пользователя, прошедшего аутентификацию. Если итоговый результат не пуст, ACL считается совпавшим и предоставляется право на запись.

Следующий рисунок поясняет, как строится данный набор:

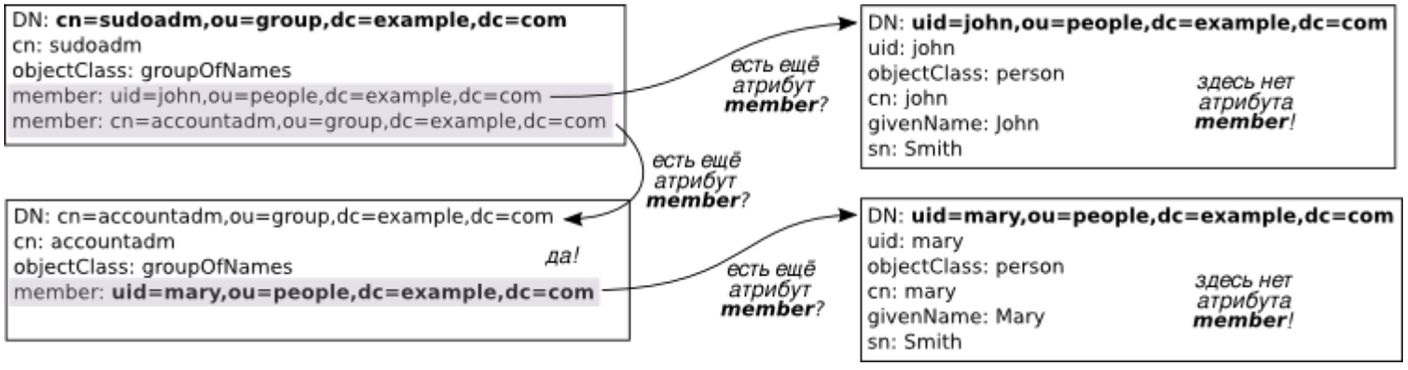

Рисунок 8.1: Заполнение набора рекурсивной группы

Сначала мы берём DN `uid=john`. Эта запись не имеет атрибута `member`, поэтому дальше мы не идём. Переходим к `cn=accountadm`. У этой записи есть атрибут `member`, содержащий `uid=mary`. Однако запись `uid=mary`, не имеет атрибута `member`, поэтому здесь мы снова останавливаемся. В конце операции получаем:

```
{"uid=john,ou=people,dc=example,dc=com","uid=mary,ou=people,dc=example,dc=com"} & user
```

Если DN пользователя, прошедшего аутентификацию, совпадает с одним из этих двух, ему предоставляется доступ на запись. Таким образом, данный набор включает `mary` в группу `sudoadm` и она получит право на запись.

8.5.2. Групповые ACL, где членство указывается без применения DN-синтакса

Традиционные групповые ACL, и даже предыдущий пример с рекурсивными группами, требуют, чтобы члены группы указывались своими DN, а не просто именами пользователей.

Однако, применение наборов позволяет использовать также простые имена в групповых ACL, что мы сейчас и продемонстрируем.

Предположим, мы хотим позволить членам группы `sudoadm` вносить изменения в ветку `ou=suders` нашего дерева. Но, в данном случае, для определения членов группы мы используем атрибут `memberUid`:

```
dn: cn=sudoadm,ou=group,dc=example,dc=com
cn: sudoadm
```

```
objectClass: posixGroup
gidNumber: 1000
memberUid: john
```

Мы не можем использовать групповые ACL с таким типом группы. Но желаемый контроль доступа можно установить с помощью наборов в ACL:

```
access to dn.subtree="ou=sudoers,dc=example,dc=com"
      by set="[cn=sudoadm,ou=group,dc=example,dc=com]/memberUid & user/uid" write
      by * read
```

Мы используем простое пересечение, где сравниваем атрибут `uid` подключившегося (и прошедшего аутентификацию) пользователя с атрибутом `memberUid` группы. Если они совпадут, тогда пересечение окажется непустым и ACL предоставит доступ на изменение.

Данный рисунок иллюстрирует работу этого набора при подключении пользователя, прошедшего аутентификацию как `uid=john,ou=people,dc=example,dc=com`:

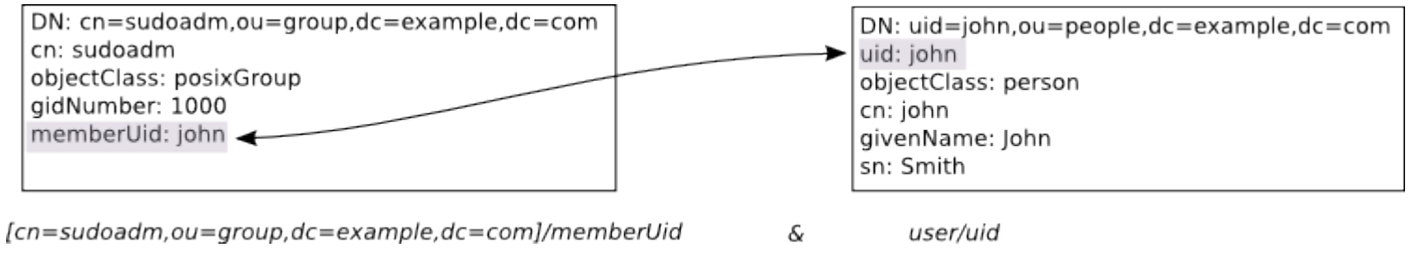

Рисунок 8.2: Применение набора с `memberUid`

В данном случае совпадение будет найдено. Однако, если же пользователь аутентифицировался бы как `mary`, доступ на изменение `ou=sudoers` был бы ей запрещён, поскольку её атрибут `uid` не перечислен в атрибуте `memberUid` группы.

8.5.3. Переход по ссылкам

Сейчас мы покажем довольно мощный пример того, что можно сделать с помощью наборов. После того, как администраторы OpenLDAP поймут данный пример и его последствия, он вызовет у них улыбку.

Начнём с записи пользователя:

```
dn: uid=john,ou=people,dc=example,dc=com
uid: john
objectClass: inetOrgPerson
givenName: John
sn: Smith
cn: john
manager: uid=mary,ou=people,dc=example,dc=com
```

С помощью наборов довольно просто составить ACL, позволяющий менеджеру (manager) обновлять некоторые атрибуты этой записи:

```
access to dn.exact="uid=john,ou=people,dc=example,dc=com"
   attrs=carLicense,homePhone,mobile,pager,telephoneNumber
   by self write
   by set="this/manager & user" write
   by * read
```

В данном наборе `this` расширяется до записи, к которой нужно получить доступ, таким образом, при получении доступа к записи john, `this/manager` расширяется до `uid=mary,ou=people,dc=example,dc=com`. Если сама менеджер получает доступ к записи John, в ACL будет найдено совпадение и доступ на изменение указанных атрибутов будет предоставлен.

Подобное поведение можно получить и с помощью ключевого слова `dnattr`. Однако, применение наборов может способствовать дальнейшему усовершенствованию этого ACL. Допустим, мы хотим разрешить секретарю (secretary) менеджера также обновлять эти атрибуты. Вот как мы это сделаем:

```
access to dn.exact="uid=john,ou=people,dc=example,dc=com"
   attrs=carLicense,homePhone,mobile,pager,telephoneNumber
```

```
  by self write
  by set="this/manager & user" write
  by set="this/manager/secretary & user" write
  by * read
```

Попробуем пояснить, что тут происходит, с помощью рисунка (записи сокращены для ясности):

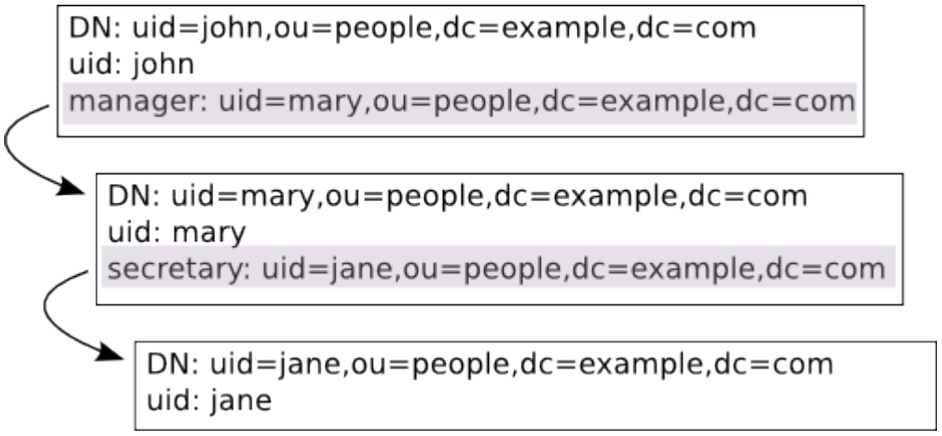

Рисунок 8.3: Переход по записям с помощью набора

В этом примере, Jane - секретарь Mary, которая, в свою очередь, является менеджером John. Все эти взаимоотношения определяются атрибутами `manager` и `secretary`, оба из которых имеют синтаксис distinguishedName (то есть, полные DN). Таким образом, при получении доступа к записи `uid=john`, набор `this/manager/secretary` становится `{"uid=jane,ou=people,dc=example,dc=com"}` (следуя по ссылкам, как показано ниже):

```
this = [uid=john,ou=people,dc=example,dc=com]
this/manager = \
  [uid=john,ou=people,dc=example,dc=com]/manager = uid=mary,ou=people,dc=example,dc=com
this/manager/secretary = \
  [uid=mary,ou=people,dc=example,dc=com]/secretary = uid=jane,ou=people,dc=example,dc=com
```

В итоге, когда Jane захочет получить доступ к записи John, ей будет предоставлено право на изменение указанных атрибутов. Более того, это касается и других записей, для которых Mary назначена менеджером.

Это всё, конечно, здорово и красиво, но не слишком ли мы развязываем руки секретарям? Думается, нужно побольше их ограничить. К примеру, давайте дадим такие привилегии только исполнительным секретарям:

```
access to dn.exact="uid=john,ou=people,dc=example,dc=com"
  attrs=carLicense,homePhone,mobile,pager,telephoneNumber
  by self write
  by set="this/manager & user" write
  by set="this/manager/secretary &
          [cn=executive,ou=group,dc=example,dc=com]/member* &
          user" write
  by * read
```

Это практически тот же самый ACL, что и прежде, но сейчас мы также требуем, чтобы подключившийся пользователь был членом группы (возможно, вложенной) `cn=executive`.

К содержанию

9. Ограничения

9.1. Введение

Как правило, желательно установить ограничения на то, сколько ресурсов сервера может потребляться каждым клиентом LDAP. OpenLDAP предоставляет два набора ограничений: ограничения по размеру, лимитирующие *количество* записей, которые клиент может получить за одну операцию, и ограничения по

времени, лимитирующие промежуток времени на выполнение операции. Оба типа ограничений могут принимать различные значения в зависимости от того, кем инициирована операция.

9.2. Мягкие и жёсткие ограничения

Администратор сервера может определить как *мягкие*, так и *жёсткие ограничения*. Мягкие ограничения можно рассматривать как значения ограничений по умолчанию. Жёсткие ограничения не могут быть превышены непривилегированными LDAP-пользователями.

При запросе поисковых операций клиенты LDAP могут определять свои собственные ограничения по размеру и по времени. Эта возможность существовала еще со времён самых ранних версий X.500.

Если клиент определил ограничение, то наименьшее значение из определённого им и *жёсткого ограничения* станет фактическим ограничением при выполнении операции.

Если клиент не определил ограничение, тогда сервер применяет *мягкое ограничение*.

Мягкие и жёсткие ограничения часто рассматриваются вместе как *административные ограничения*. Поэтому, когда клиент LDAP пытается выполнить поисковую операцию, которая вернёт больше данных, чем позволяют ограничения, он получит ошибку *adminLimitExceeded (превышены административные ограничения)*. Обратите внимание, что сервер обычно возвращает некоторое количество результатов, даже если ограничения превышены: данная возможность полезна клиентам, которые хотят только убедиться в существовании определённых записей и не нуждаются в получении их всех.

На *rootdn* не накладываются никакие ограничения.

9.3. Глобальные ограничения

Ограничения, определённые в глобальной части конфигурации сервера, выступают в качестве ограничений по умолчанию, которые используются в случае, если для той или иной базы данных не установлены более конкретные ограничения.

В конфигурационном файле *slapd.conf*(5) применяются директивы `sizelimit` и `timelimit`. При использовании механизма манипуляции данными *slapd config* применяются соответствующие атрибуты `olcSizeLimit` и `olcTimeLimit`. В обоих случаях синтаксис этих значений одинаков.

В простой форме записи мягкие и жёсткие ограничения устанавливаются в одно и то же значение:

```
sizelimit {<целое число>|unlimited}
timelimit {<целое число>|unlimited}
```

Значения по умолчанию: для sizelimit - 500 записей, для timelimit - 3600 секунд.

Расширенная форма позволяет отдельно задавать мягкие и жёсткие ограничения:

```
sizelimit size[.{soft|hard|unchecked}]=<целое число> [...]
timelimit time[.{soft|hard}]=<целое число> [...]
```

Таким образом, чтобы установить мягкие ограничения по размеру в 10 записей, а жёсткие - в 75 записей, потребуется следующая директива:

```
sizelimit size.soft=10 size.hard=75
```

Ключевое слово *unchecked* устанавливает ограничение на количество записей, которые рассмотрит сервер после того, как он создал первоначальный набор записей-кандидатов на возвращение в ответ на поисковый запрос с использованием индексов. В больших каталогах это может быть очень важно, поскольку, если ответ на поисковый запрос невозможно найти с помощью индексов, то сервер будет пересматривать миллионы записей. Поэтому всегда старайтесь настроить правильное индексирование.

9.4. Ограничения для отдельной базы данных

Каждая база данных может иметь свой собственный набор ограничений, переопределяющих глобальные ограничения. Синтаксис таких ограничений более гибок и позволяет применять различные ограничения к разным сущностям. Обратите внимание, что *сущность* не то же самое, что и *запись*: термин *сущность* в данном случае обозначает идентификатор человека или процесса, инициирующего операцию LDAP.

В конфигурационном файле *slapd.conf*(5) используется директива `limits`. При использовании механизма манипуляции данными *slapd config* применяется соответствующий атрибут `olcLimits`. В обоих случаях синтаксис этих значений одинаков.

```
limits <who> <limit> [<limit> [...]]
```

Условия *limits* могут быть указаны несколько раз, чтобы задать разные ограничения для разных сущностей. Сервер проверяет поочерёдно каждое условие, пока не найдёт такое, которое совпадает с сущностью, запросившей операцию. Если соответствие не найдено, применяются глобальные ограничения.

9.4.1. Определение того, на кого накладываются ограничения

Часть `<who>` условия *limits* может принимать любое из следующих значений:

Таблица 9.1: Спецификаторы сущностей

Спецификатор	**Сущности**
*	Все, включая анонимных пользователей и пользователей, прошедших аутентификацию
`anonymous`	Анонимные (не прошедшие аутентификацию) пользователи
`users`	Пользователи, прошедшие аутентификацию
`self`	Пользователь, ассоциированный с целевой записью
`dn[.<basic-style>]=<regex>`	Пользователи, записи которых соответствуют регулярному выражению
`dn.<scope-style>=<DN>`	Пользователи в диапазоне DN
`group[/oc[/at]]=<шаблон>`	Члены группы

Правила спецификации условия `<who>` аналогичны используемым при контроле доступа.

9.4.2. Определение ограничений по времени

Синтаксис ограничений по времени:

```
time[.{soft|hard}]=<целое число>
```

где `<целое число>` - количество секунд, которое потратит slapd, отвечая на поисковый запрос.

Если не указано ни одно из ключевых слов *soft* или *hard*, значение используется для обоих типов ограничений, например:

```
limits anonymous time=27
```

Значение *unlimited* может быть использовано, чтобы полностью удалить жёсткое ограничение по времени, например:

```
limits dn.exact="cn=anyuser,dc=example,dc=org" time.hard=unlimited
```

9.4.3. Определение ограничений по размеру

Синтаксис ограничений по размеру:

```
size[.{soft|hard|unchecked}]=<целое число>
```

где `<целое число>` - максимальное количество записей, которое вернёт slapd, отвечая на поисковый запрос.

Доступные ключевые слова "soft", "hard" и "unchecked" имеют то же самое значение, что и в описанных выше настройках глобальных ограничений.

9.4.4. Ограничение по размеру и постраничный вывод результатов

Если клиент LDAP добавляет к операции поиска элемент управления *pagedResultsControl*, по умолчанию используются жёсткие ограничения по размеру, поскольку запрос определённого размера страницы считается явным запросом ограничения на количество возвращаемых записей. Однако, ограничение по размеру применяется к общему числу записей, возвращаемых в результате поиска, а не к одной странице.

Для поиска с постраничным выводом результатов можно принудительно установить дополнительные ограничения по размеру.

Ограничение `size.pr` контролирует максимальный размер страницы:

```
size.pr={<целое число>|noEstimate|unlimited}
```

`<Целое число>` - это максимальный размер страницы, если он не был явно задан при запросе. Значение `noEstimate` в данной реализации программного обеспечения не имеет эффекта, поскольку сервер в любом случае не возвращает оценку предполагаемых размеров результата. Значение `unlimited` указывает на то, что к максимальному размеру страницы не применяется ограничений.

Ограничение `size.prtotal` контролирует общее количество записей, возвращаемых при поиске с постраничным выводом результатов. По умолчанию оно совпадает с нормальным ограничением `size.hard`.

```
size.prtotal={<целое число>|unlimited|disabled}
```

Значение `unlimited` снимает ограничение на количество записей, возвращаемых при поиске с постраничным выводом результатов. Значение `disabled` может быть использовано для выборочного отключения возможности постраничного возврата результатов поиска.

9.5. Примеры настройки ограничений

9.5.1. Простые глобальные ограничения

Этот простой фрагмент глобальной конфигурации устанавливает ограничения по размеру и времени на все операции поиска всех пользователей, за исключением *rootdn*. Он ограничивает результаты поиска 50-ю записями, а общее время на выполнение операции - 10-ю секундами.

```
sizelimit 50
timelimit 10
```

9.5.2. Мягкие и жёсткие глобальные ограничения

Иногда полезно ограничить размер возвращаемых результатов, но позволить клиентам запрашивать больший размер при необходимости. Это может быть достигнуто путём установки отдельно жёстких и мягких ограничений.

```
sizelimit size.soft=5 size.hard=100
```

Для предотвращения выполнения клиентами очень неэффективных операций поиска по неиндексированным значениям, добавим ограничение *unchecked*:

```
sizelimit size.soft=5 size.hard=100 size.unchecked=100
```

9.5.3. Установка определённым пользователям ограничений большего размера

Установив ограничения по умолчанию в глобальной конфигурации, Вы можете дать определённым

пользователям возможность получения большего количества результатов. Это можно сделать при настройке отдельной базы данных:

```
limits dn.exact="cn=anyuser,dc=example,dc=org" size=100000
limits dn.exact="cn=personnel,dc=example,dc=org" size=100000
limits dn.exact="cn=dirsync,dc=example,dc=org" size=100000
```

Как правило, при настройке сервера лучше избегать упоминания конкретных пользователей, а установить ограничения большего размера группе:

```
limits group/groupOfNames/member="cn=bigwigs,dc=example,dc=org" size=100000
```

9.5.4. Установка ограничений на возможность выполнения поиска с постраничным выводом результатов

Может возникнуть ситуация, когда некоторым приложениям требуется выполнять поиск с очень большим количеством результатов, для которого они используют постраничный вывод, но Вы не хотите разрешать обычным пользователям LDAP использовать элемент управления pagedResults. В этом случае могут помочь ограничения *pr* и *prtotal*:

```
limits group/groupOfNames/member="cn=dirsync,dc=example,dc=org" size.prtotal=unlimited
limits users size.soft=5 size.hard=100 size.prtotal=disabled
limits anonymous size.soft=2 size.hard=5 size.prtotal=disabled
```

9.6. Дополнительная информация

Дополнительную информацию можно найти в *slapd.conf*(5), *ldapsearch*(1) и *slapd.access*(5)

К содержанию

10. Инструменты создания и обслуживания баз данных

Данный раздел расскажет Вам, как создавать базу данных slapd с нуля и устранять возникающие при этом проблемы, если Вы с ними столкнётесь. Существует два пути создания базы данных. Первый из них - создание базы данных при запущенном slapd с использованием операций LDAP. При использовании этого метода Вы просто запускаете slapd и добавляете записи с помощью выбранного Вами клиента LDAP. Такой метод хорош для относительно маленьких баз данных (несколько сот или тысяч записей, в зависимости от Ваших потребностей). Данный метод применим к тем типам баз данных, которые поддерживают обновления.

Второй метод - создание базы данных при остановленном slapd с помощью специальных утилит, поставляемых с *slapd*(8). Такой метод лучше всего подходит, если Вам надо создать много тысяч записей, что займёт неприемлемо долгое время при использовании LDAP-метода, или Вы хотите убедиться, что во время создания базы данных к ней не будет осуществляться доступ. Обратите внимание, что не все типы баз данных поддерживают эти утилиты.

10.1. Создание базы данных через LDAP

В данном методе Вы используете выбранный Вами клиент LDAP (например, *ldapadd*(1)) для добавления записей, точно также, как Вы это делаете в уже созданных базах данных. Перед запуском *slapd*(8) убедитесь, что в конфигурационном файле заданы указанные ниже директивы.

```
suffix <dn>
```

Как описано в подразделе Общие директивы баз данных, данная директива определяет, какие именно записи будут содержаться в этой базе данных. Задайте в ней DN корневой записи того поддерева, которое Вы

собираетесь создать. Например:

```
suffix "dc=example,dc=com"
```

Убедитесь, что Вы указали директорию, в которую будут помещаться индексные файлы:

```
directory <директория>
```

Например:

```
directory /usr/local/var/openldap-data
```

Необходимо создать данную директорию с соответствующими правами, такими, чтобы slapd мог производить в неё запись.

Кроме того, необходимо настроить slapd так, чтобы Вы могли подсоединиться к нему под учётной записью пользователя каталога с правами на добавление записей. Можно настроить каталог, чтобы он поддерживал специального *администратора* или пользователя *root* только для этой цели. Это делается с помощью следующих двух директив в определении базы данных:

```
rootdn <dn>
rootpw <пароль>
```

Например:

```
rootdn "cn=Manager,dc=example,dc=com"
rootpw secret
```

Эти директивы указывают DN и пароль, которые могут быть использованы для аутентификации в качестве записи *администратора* базы данных (то есть записи, которой разрешено делать всё, что угодно). DN и пароль, указанные здесь, будут работать всегда, независимо от того, существует ли актуальная запись с этим именем в каталоге и задан ли для неё пароль. Таким способом решается проблема курицы и яйца применительно к тому, как проходить аутентификацию и добавлять записи, если в каталоге еще не существует ни одной записи.

Наконец, Вы должны убедиться, что определение базы данных содержит нужные Вам определения индексов:

```
index {<список атрибутов> | default} [pres,eq,approx,sub,none]
```

Например, для индексирования атрибутов `cn`, `sn`, `uid` и `objectclass` могут быть использованы следующие директивы `index`:

```
index cn,sn,uid pres,eq,approx,sub
index objectClass eq
```

В данном примере создаются индексы наличия, равенства, приблизительного равенства и равенства подстроке для атрибутов `cn`, `sn` и `uid`, а также индекс равенства для атрибута `objectClass`. Имейте ввиду, что не все типы индексов доступны для каждого типа атрибута. Для получения дополнительной информации по данной директиве смотрите раздел Конфигурационный файл slapd.

После того, как Вы установили эти параметры под Ваши нужды, запустите slapd, подсоединитесь к нему с помощью клиента LDAP и начинайте добавлять записи. Например, чтобы добавить записи организации и организационной роли с помощью инструмента *ldapadd*, Вы можете создать LDIF-файл `entries.ldif` следующего содержания:

```
# Организация для Example Corporation
dn: dc=example,dc=com
objectClass: dcObject
objectClass: organization
dc: example
o: Example Corporation
description: The Example Corporation

# Организационная роль для Directory Manager
dn: cn=Manager,dc=example,dc=com
objectClass: organizationalRole
cn: Manager
description: Directory Manager
```

а затем использовать такую команду, чтобы фактически создать запись:

```
ldapadd -f entries.ldif -x -D "cn=Manager,dc=example,dc=com" -w secret
```

В этой команде используются настройки, заданные в приведённых выше примерах.

10.2. Создание базы данных при отключенном slapd

Второй метод - создать базу данных при отключенном slapd, используя описанные ниже инструменты slapd для работы с базами данных. Этот метод лучше всего подходит, если Вам надо создать много тысяч записей, что займёт неприемлемо долгое время при использовании описанного выше LDAP-метода. Данные инструменты читают конфигурационный файл slapd, а также поданный на вход файл, содержащий текстовое представление добавляемых записей. Они непосредственно создают файлы баз данных для тех типов баз данных, которые поддерживаются этими инструментами. Для других типов баз данных Вы должны использовать описанный выше метод создания базы данных при запущенном slapd. Сначала нужно убедиться, что в определении базы данных в конфигурационном файле заданы несколько важных директив:

```
suffix <dn>
```

Как описано в подразделе Общие директивы баз данных, данная директива определяет, какие именно записи будут содержаться в этой базе данных. Задайте в ней DN корневой записи того поддерева, которое Вы собираетесь создать. Например:

```
suffix "dc=example,dc=com"
```

Убедитесь, что Вы указали директорию, в которую будут помещаться индексные файлы:

```
directory <директория>
```

Например:

```
directory /usr/local/var/openldap-data
```

Наконец, Вам необходимо определить индексы, которые Вы хотите построить. Это делается с помощью одной или нескольких директив index.

```
index {<список атрибутов> | default} [pres,eq,approx,sub,none]
```

Например:

```
index cn,sn,uid pres,eq,approx,sub
index objectClass eq
```

В данном примере создаются индексы наличия, равенства, приблизительного равенства и равенства подстроке для атрибутов `cn`, `sn` и `uid`, а также индекс равенства для атрибута `objectClass`. Имейте в виду, что не все типы индексов доступны для каждого типа атрибута. Для получения дополнительной информации по данной директиве смотрите раздел Конфигурационный файл slapd.

10.2.1. Программа `slapadd`

После того, как Вы установили эти параметры под Ваши нужды, Вы создаёте основную базу данных и соответствующие индексы путём запуска программы *slapadd*(8):

```
slapadd -l <входной файл> -f <файл настроек slapd>
        [-d <уровень отладки>] [-n <целое число>|-b <суффикс>]
```

Значения параметров следующие:

```
-l <входной файл>
```

Указывает подаваемый на вход файл LDIF, содержащий требуемые для добавления записи в текстовой форме, описанной ниже в подразделе Формат текстового представления записи LDIF.

```
-f <файл настроек slapd>
```

Указывает конфигурационный файл slapd, в котором определено, где создавать индексы, какие индексы создавать и т.д.

```
-F <директория настроек slapd>
```

Указывает конфигурационную директорию. Если задано сразу оба параметра `-f` и `-F`, конфигурационный файл будет прочитан, переконвертирован в формат конфигурационной директории и записан в указанную директорию. Если ни одного из этих двух параметров не было задано, slapd попытается прочитать конфигурационную директорию по умолчанию перед тем, как попытаться использовать конфигурационный файл по умолчанию. Если существует конфигурационная директория правильного формата, тогда конфигурационный файл по умолчанию игнорируется. Если при этом также указан режим холостого запуска, никакого преобразования произведено не будет.

```
-d <уровень отладки>
```

Включает вывод отладочной информации согласно указанному `<уровню отладки>`. Уровни отладки аналогичны соответствующим уровням для slapd. Смотрите подраздел Параметры командной строки раздела Запуск slapd.

```
-n <номер базы данных>
```

Необязательный параметр, указывающий, в какую из баз данных вносить изменения. Первая из перечисленных в конфигурационном файле баз данных получает номер `1`, вторая - `2` и т.д. По умолчанию используется первая база данных из перечисленных в конфигурационном файле. Данный параметр не должен использоваться совместно с параметром `-b`.

```
-b <суффикс>
```

Необязательный параметр, указывающий, в какую из баз данных вносить изменения. Для определения искомой базы данных устанавливается соответствие между передаваемым суффиксом и суффиксом, указанным в директиве `suffix` базы данных. Данный параметр не должен использоваться совместно с параметром `-n`.

10.2.2. Программа `slapindex`

Иногда возникает необходимость пересоздать индексы (например, после изменения *slapd.conf*(5)). Это можно сделать с помощью программы *slapindex*(8). *slapindex* вызывается следующим образом:

```
slapindex -f <файл настроек slapd>
        [-d <уровень отладки>] [-n <номер базы данных>|-b <суффикс>]
```

где параметры `-f`, `-d`, `-n` и `-b` аналогичны соответствующим параметрам программы *slapadd*(1). *slapindex* перестраивает все индексы на основании текущего содержимого базы данных.

10.2.3. Программа `slapcat`

Программа `slapcat` используется для выгрузки содержимого базы данных в LDIF-файл. Это может быть полезно, если Вы хотите сделать читабельную резервную копию Вашей базы данных или если Вы хотите отредактировать Вашу базу данных при остановленном slapd. Программа вызывается следующим образом:

```
slapcat -l <имя файла> -f <файл настроек slapd>
        [-d <уровень отладки>] [-n <номер базы данных>|-b <суффикс>]
```

где `-n` или `-b` используются для выбора базы данных из перечисленных в файле *slapd.conf*(5), указанном с помощью `-f`. Соответствующий выходной LDIF пишется на стандартный вывод, либо в файл, указанный в параметре `-l`.

10.3. Формат текстового представления записи LDIF

LDAP Data Interchange Format (LDIF, Формат обмена данными LDAP) используется для представления записей LDAP в простом текстовом формате. В этом подразделе дано краткое описание формата представления записи LDIF, дополнительные сведения можно найти в *ldif*(5) и технической спецификации RFC2849 (рус.) (RFC2849 (ориг.)).

Основная форма представления записи следующая:

```
# комментарий
dn: <уникальное имя (distinguished name)>
<описание атрибута>: <значение атрибута>
<описание атрибута>: <значение атрибута>

...
```

Строки, начинающиеся с символа '#', - комментарии. Описание атрибута может быть либо простым типом атрибута, таким как `cn` или `objectClass`, либо в форме `1.2.3` (OID, ассоциированный с типом атрибута), либо может включать опции, такие как `cn;lang_en_US` или `userCertificate;binary`.

Строка может быть продолжена путём добавления *одиночного* символа пробела или табуляции в начало следующей строки. Например:

```
dn: cn=Barbara J Jensen,dc=example,dc=
 com
cn: Barbara J
  Jensen
```

эквивалентно:

```
dn: cn=Barbara J Jensen,dc=example,dc=com
cn: Barbara J Jensen
```

Несколько значений одного и того же атрибута указывается в отдельных строках. Например:

```
cn: Barbara J Jensen
cn: Babs Jensen
```

Если `<значение атрибута>` содержит непечатные символы или начинается с пробела, двоеточия (':') или символа "меньше" ('<'), то за `<описанием атрибута>` следует двойное двоеточие и значение атрибута, закодированное в формате base64. Например, значение `" begins with a space"` будет закодировано в таком виде:

```
cn:: IGJlZ2lucyB3aXRoIGEgc3BhY2U=
```

Также можно указать URL ресурса, содержащего значение атрибута. В следующем примере указывается, что значение атрибута `jpegPhoto` будет получено из файла `/path/to/file.jpeg`.

```
cn:< file:///path/to/file.jpeg
```

В одном файле LDIF можно указать несколько записей, разделяя их пустой строкой. Вот пример LDIF-файла, содержащего три записи:

```
# Запись Barbara
dn: cn=Barbara J Jensen,dc=example,dc=com
cn: Barbara J Jensen
cn: Babs Jensen
objectClass: person
sn: Jensen

# Запись Bjorn
dn: cn=Bjorn J Jensen,dc=example,dc=com
cn: Bjorn J Jensen
cn: Bjorn Jensen
objectClass: person
sn: Jensen
# JPEG-фотография, закодированная в формате Base64
jpegPhoto:: /9j/4AAQSkZJRgABAAAAAQABAAD/2wBDABALD
 A4MChAODQ4SERATGCgaGBYWGDEjJR0oOjM9PDkzODdASFxOQ
 ERXRTc4UG1RV19iZ2hnPk1xeXBkeFxlZ2P/2wBDARESEhgVG

# Запись Jennifer
dn: cn=Jennifer J Jensen,dc=example,dc=com
cn: Jennifer J Jensen
cn: Jennifer Jensen
objectClass: person
sn: Jensen
# JPEG-фотография из файла
jpegPhoto:< file:///path/to/file.jpeg
```

Обратите внимание, что `jpegPhoto` в записи Bjorn закодировано в формате base64, а `jpegPhoto` в записи

Jennifer будет получено из расположения, указанного URL.

Примечание: В файле LDIF из значений атрибутов не обрезаются конечные пробелы. Также несколько пробелов внутри значения не усекаются до одного. Если Вы не хотите, чтобы они присутствовали в Ваших данных, просто не помещайте их туда.

К содержанию

11. Механизмы манипуляции данными

Механизмы манипуляции данными выполняют реальную работу по хранению и извлечению данных в ответ на запросы LDAP. Они могут быть статически скомпилированы в *slapd*, или, если включена поддержка модулей, они могут быть загружены динамически.

Если ваша инсталляция slapd использует динамические модули, вам может понадобиться добавить соответствующие директивы *moduleload* как показано в следующих примерах. Название модуля механизма манипуляции обычно указывается в форме:

```
back_<имя механизма>.la
```

Например, для загрузки механизма *hdb* Вам нужно указать:

```
moduleload back_hdb.la
```

11.1. Механизмы Berkeley DB

11.1.1. Обзор

Механизм *hdb* является первичным и рекомендуемым механизмом для нормальной базы данных *slapd*. Для хранения данных он использует пакет Oracle Berkeley DB (BDB). Этот механизм позволяет широко применять индексирование и кэширование для ускорения доступа к данным (смотрите раздел Настройка производительности).

hdb — это вариант оригинального механизма *bdb*, первоначально написанного для работы с BDB. *hdb* использует иерархическую структуру базы данных с поддержкой переименований на уровне поддеревьев. Во всём остальном его поведение аналогично поведению *bdb*, и к обоим применимы одинаковые параметры конфигурации.

Замечание: База данных *hdb* требует большого размера *idlcachesize* для хорошей производительности операций поиска, как правило в три раза или более превышающего *cachesize* (размер кэша записи).

Замечание: Механизм *hdb* вытеснил *bdb*, а скоро оба они будут считаться устаревшими в пользу нового механизма *mdb*. Смотрите ниже.

11.1.2. Настройка back-bdb/back-hdb

Подробности будут позже.

11.1.3. Дополнительная информация

slapd-bdb(5)

11.2. LDAP

11.2.1. Обзор

Механизм *slapd* LDAP в действительности не является базой данных; вместо этого он выступает как прокси для перенаправления входящих запросов на другой сервер LDAP. Во время обработки запроса происходит также разрешение ссылок, то есть ссылки обрабатываются полностью, а не возвращаются LDAP-клиенту.

При работе сессий, *подключающихся* к базе данных *back-ldap*, для каждой из них всегда создается отдельное соединение к удаленному серверу LDAP. При этом анонимные сессии будут работать в рамках одного фактического анонимного подключения к удалённому серверу. Сессии, устанавливаемые с использованием какого-либо механизма авторизации, при использовании одного и того же DN также будут использовать одно фактическое подключение. Подобная стратегия группировки соединений в пул может повысить эффективность прокси-сервера за счет снижения накладных расходов при неоднократной установке/разрыве многочисленных однотипных соединений.

Также механизм манипуляции ldap может использоваться для получения какой-либо дополнительной информации из службы каталогов, для этого на удалённый сервер передаются данные аутентификации локально-авторизованных клиентов, возможно в несколько изменённой форме. С этой целью прокси присоединяет к удалённому серверу под учетной записью с административными правами и, если необходимо, запрашивает данные для той учётной записи, которую ему подали на вход.

Данный механизм доступа очень часто используется многими другими механизмами манипуляции данными и наложениями.

11.2.2. Настройка back-ldap

Как было сказано выше, *slapd-ldap(5)* используется за кулисами многих других механизмов манипуляции данными и наложений. Часто они имеют лишь небольшое количество собственных настроек, но предоставляют администратору возможность использовать все настройки *slapd-ldap(5)*.

Например, Translucent Proxy (прозрачный прокси), получающий записи с удалённого LDAP-сервера с возможностью частичной их замены из определённой базы данных, имеет только четыре собственных специфических *translucent*-директивы настройки, однако может быть сконфигурирован с использованием всех стандартных настроек *slapd-ldap(5)*. За подробностями обратитесь к *slapo-translucent(5)*.

Другие наложения позволяют Вам добавлять директивы перед вызовом нормальных директив *slapd-ldap(5)*. Например, так происходит с наложением *slapo-chain(5)*:

"Существует совсем немного директив, специфичных для наложения сцепления; однако, директивы, имеющие отношение к механизму манипуляции данными ldap, который неявно вызывается этим наложением, могут иметь особое значение в случае их использования с этим наложением (то есть отличное от директив самого механизма манипуляции данными ldap). Они описаны в slapd-ldap(5), и к ним также нужно добавлять префикс chain-."

Также можно встретить применение механизма манипуляции данными *slapd-ldap(5)* в описании репликации, основанной на посылках, в соответствующем разделе данного руководства.

Как видите, механизм манипуляции данными *slapd-ldap(5)* чрезвычайно гибок и активно используется во всем ПО OpenLDAP.

Пример, приведённый ниже, очень прост, но и он показывает сильные стороны механизма манипуляции данными *slapd-ldap(5)*, достигаемые за счёт использования списка *uri*:

```
database        ldap
suffix          "dc=suretecsystems,dc=com"
rootdn          "cn=slapd-ldap"
uri             ldap://localhost/ ldap://remotehost ldap://remotehost2
```

Список URI разделяется пробелами или запятыми. Всякий раз, когда первый в списке сервер не отвечает, список пересортируется таким образом, что первым оказывается отвечающий в данный момент сервер, и при

следующем запросе первая попытка обращения будет уже к нему.

Эта особенность может быть использована для обеспечения балансировки нагрузки при репликации в режиме зеркала (MirrorMode).

11.2.3. Дополнительная информация

slapd-ldap(5)

11.3. LDIF

11.3.1. Обзор

Механизм манипуляции данными LDIF — это один из основных механизмов для *slapd*(8), который хранит записи в текстовых файлах в формате LDIF и использует файловую систему для создания древовидной структуры базы данных. Этот механизм низкопроизводителен, однако прост в использовании и нетребователен к ресурсам.

При использовании динамической конфигурации *cn=config* для хранения постоянной базы данных конфигурации используется как раз этот механизм манипуляции данными. За дополнительной информацией обращайтесь к *slapd-config*(5).

11.3.2. Настройка back-ldif

Как и многие другие механизмы манипуляции данными, механизм LDIF подключается всего несколькими строками конфигурации:

```
include ./schema/core.schema

database  ldif
directory ./ldif
suffix    "dc=suretecsystems,dc=com"
rootdn    "cn=LDIF,dc=suretecsystems,dc=com"
rootpw    LDIF
```

Проследим, что происходит при добавлении новой записи. Чтобы добавить *dcObject* для *dc=suretecsystems,dc=com* создадим файл suretec.ldif со следующим содержимым:

```
dn: dc=suretecsystems,dc=com
objectClass: dcObject
objectClass: organization
dc: suretecsystems
o: Suretec Systems Ltd
```

Теперь добавим его в наш каталог:

```
ldapadd -x -H ldap://localhost:9011 -f suretec.ldif -D "cn=LDIF,dc=suretecsystems,dc=com" -w LDIF
adding new entry "dc=suretecsystems,dc=com"
```

Посмотрим, что получилось в директории `./ldif`:

```
ls ./ldif
dc=suretecsystems,dc=com.ldif
```

Наконец, заглянем внутрь этого файла:

```
cat ldif/dc\=suretecsystems\,dc\=com.ldif

dn: dc=suretecsystems
objectClass: dcObject
objectClass: organization
dc: suretecsystems
o: Suretec Systems Ltd.
structuralObjectClass: organization
entryUUID: 2134b714-e3a1-102c-9a15-f96ee263886d
creatorsName: cn=LDIF,dc=suretecsystems,dc=com
```

```
createTimestamp: 20080711142643Z
entryCSN: 20080711142643.661124Z#000000#000#000000
modifiersName: cn=LDIF,dc=suretecsystems,dc=com
modifyTimestamp: 20080711142643Z
```

Данный полный формат можно получить при экспорте Вашего каталога с помощью `slapcat`.

11.3.3. Дополнительная информация

slapd-ldif(5)

11.4. LMDB

11.4.1. Обзор

Механизм *slapd*(8) *mdb* рассматривается в перспективе как основной механизм для нормальных баз данных *slapd*. Он предназначен для замены механизмов манипуляции данными Berkeley DB и использует для хранения данных собственную библиотеку OpenLDAP Lightning Memory-Mapped Database (LMDB, высокоскоростная отображаемая в памяти база данных).

Как и механизмы, BDB он поддерживает индексирование, но не использует кэширование и не требует тонкой подстройки для обеспечения максимальной производительности поиска. Также как и *hdb*, этот механизм полностью иерархичен и поддерживает одновременное переименование поддеревьев.

11.4.2. Настройка back-mdb

В отличие от механизмов BDB, *mdb* может быть проинициализирован с помощью всего нескольких строк конфигурации:

```
include ./schema/core.schema

database  mdb
directory ./mdb
suffix    "dc=suretecsystems,dc=com"
rootdn    "cn=mdb,dc=suretecsystems,dc=com"
rootpw    mdb
maxsize   1073741824
```

В дополнение к обычным параметрам, необходимым для минимальной конфигурации, механизм *mdb* требует указания максимального размера. Он задаётся в байтах и должен быть больше ожидаемого размера базы данных, даже с учётом её прироста. В файловой системе также должно быть достаточно свободного места для размещения базы такого размера.

11.4.3. Дополнительная информация

slapd-mdb(5)

11.5. Метакаталог

11.5.1. Обзор

Механизм манипуляции данными *slapd*(8) meta выполняет основные операции LDAP-проксирования для множества удалённых серверов LDAP, называемых "цели" ("targets"). Информация, содержащаяся на этих серверах, может быть представлена как принадлежащая к одному информационному дереву каталога (DIT).

Для понимания работы данного механизма рекомендуется разобраться в функционировании другого механизма — *slapd-ldap*(5), поскольку данный механизм был разработан как усовершенствование механизма ldap. Оба этих механизма имеют много общих особенностей (в действительности, они разделяют также части исходного кода). Если механизм ldap предназначен для выполнения прокси-операций в отношении одного

сервера, то механизм meta в основном предназначен для проксирования нескольких серверов с возможностью подмены пространства имён.

Эти функции, безусловно полезные во многих ситуациях, могут привести к чрезмерному расходу ресурсов для некоторых приложений, поэтому использование данного механизма должно быть тщательно продумано.

11.5.2. Настройка back-meta

БУДЕТ ПОЗЖЕ

11.5.3. Дополнительная информация

slapd-meta(5)

11.6. Мониторинг

11.6.1. Обзор

Механизм манипуляции данными *slapd*(8) monitor в действительности не является базой данных; если он активирован, в каталоге автоматически создаётся и динамически поддерживается ветка с информацией о текущем статусе демона slapd.

Для просмотра полных данных мониторинга нужно сделать поисковый запрос с базовым DN *cn=Monitor* и указанием необходимости получения атрибутов "+" и "*". Дело в том, что механизм манипуляции данными monitor позволяет отслеживать большинство операционных атрибутов, и, поскольку их очень много, LDAP возвращает только те из них, которые были явно запрошены. Запрос атрибута "+" — это метафора, запрашивающая все операционные атрибуты.

За дополнительной информацией обратитесь к разделу Мониторинг.

11.6.2. Настройка back-monitor

База данных monitor может быть подключена только один раз, то есть только одно включение "database monitor" может присутсвовать в файле *slapd.conf(5)*. Суффикс ветки мониторинга также автоматически назначается как *"cn=Monitor"*.

Однако вы можете установить *rootdn* и *rootpw*. В примере ниже Вы найдёте всё, что нужно для подключения механизма манипуляции данными monitor:

```
include ./schema/core.schema

database monitor
rootdn "cn=monitoring,cn=Monitor"
rootpw monitoring
```

Также вы можете назначить этой базе данных контроль доступа так же, как и любой другой, например:

```
access to dn.subtree="cn=Monitor"
     by dn.exact="uid=Admin,dc=my,dc=org" write
     by users read
     by * none
```

Замечание: Набор схемы `core.schema` должен быть подгружен, чтобы база данных monitor могла функционировать.

Вот небольшой пример данных, возвращаемых *ldapsearch*:

```
ldapsearch -x -H ldap://localhost:9011 -b 'cn=Monitor'
# extended LDIF
#
# LDAPv3
```

```
# base <cn=Monitor> with scope subtree
# filter: (objectclass=*)
# requesting: ALL
#

# Monitor
dn: cn=Monitor
objectClass: monitorServer
cn: Monitor
description: This subtree contains monitoring/managing objects.
description: This object contains information about this server.
description: Most of the information is held in operational attributes, which
 must be explicitly requested.

# Backends, Monitor
dn: cn=Backends,cn=Monitor
objectClass: monitorContainer
cn: Backends
description: This subsystem contains information about available backends.
```

Чтобы посмотреть более развёрнутые примеры того, какая информация доступна с помощью данного механизма манипуляции данными, обратитесь к разделу Мониторинг.

11.6.3. Дополнительная информация

slapd-monitor(5)

11.7. Null

11.7.1. Обзор

Механизм манипуляции данными *slapd*(8) Null — безусловно, самая полезная часть slapd:

- Поисковые запросы возвращают код успешного завершения, но записей не возвращают.
- Запросы на сравнение возвращают compareFalse.
- Запросы на обновление возвращают код успешного завершения (если, конечно, не включен режим "только для чтения"), но ничего не делают.
- Подключения не от rootdn заканчиваются неудачей, если не была указана директива "bind on" в секции database.
- Весьма интересно также поведение инструментов slapadd(8) and slapcat(8).

На создание данного механизма вдохновило устройство `/dev/null`.

11.7.2. Настройка back-null

Этот механизм обладает, пожалуй, самой простой и короткой настройкой, какую Вам когда-либо приходилось делать. Чтобы его протестировать, Ваш файл `slapd.conf` должен выглядеть примерно так:

```
database null
suffix "cn=Nothing"
bind on
```

bind on означает:

"Разрешается подключение от имени любого DN в данном суффиксе с любым паролем. Значение по умолчанию "off"."

Протестируем данных механизм с помощью *ldapsearch*:

```
ldapsearch -x -H ldap://localhost:9011 -D "uid=none,cn=Nothing" -w testing -b 'cn=Nothing'
# extended LDIF
#
# LDAPv3
# base <cn=Nothing> with scope subtree
# filter: (objectclass=*)
# requesting: ALL
#
```

```
# search result
search: 2
result: 0 Success

# numResponses: 1
```

11.7.3. Дополнительная информация

slapd-null(5)

11.8. Passwd

11.8.1. Обзор

Механизм манипуляции данными *slapd*(8) PASSWD предназначен для отображения учётных данных, перечисленных в системном файле *passwd*(5) (по умолчанию это `/etc/passwd`).

Данный механизм предоставляется только в демонстрационных целях. DN каждой записи выглядит так: "uid=<username>,<suffix>".

11.8.2. Настройка back-passwd

Настройка данного механизма `slapd.conf` длиннее, чем предыдущего, но совсем немного:

```
include ./schema/core.schema

database passwd
suffix "cn=passwd"
```

Результат запроса с помощью *ldapsearch* будет примерно такой:

```
ldapsearch -x -H ldap://localhost:9011 -b 'cn=passwd'
# extended LDIF
#
# LDAPv3
# base <cn=passwd> with scope subtree
# filter: (objectclass=*)
# requesting: ALL
#

# passwd
dn: cn=passwd
cn: passwd
objectClass: organizationalUnit

# root, passwd
dn: uid=root,cn=passwd
objectClass: person
objectClass: uidObject
uid: root
cn: root
sn: root
description: root
```

11.8.3. Дополнительная информация

slapd-passwd(5)

11.9. Perl/Shell

11.9.1. Обзор

Механизм манипуляции данными *slapd*(8) Perl встраивает в *slapd*(8) интерпретатор *perl*(1). В каждой секции database perl конфигурационного файла *slapd.conf*(5) должно быть определено какие модули Perl следует

использовать. Затем slapd создаёт новый объект Perl, который обрабатывает все запросы, относящиеся к данному экземпляру настроек механизма манипуляции данными.

Механизм манипуляции данными *slapd*(8) Shell вызывает внешние программы для выполнения операций. Он разработан, чтобы облегчить привязку уже существующих баз данных к интерфейсу запросов slapd. Этот механизм предназначен, прежде всего, для использования в прототипах.

11.9.2. Настройка back-perl/back-shell

БУДЕТ ПОЗЖЕ

11.9.3. Дополнительная информация

slapd-shell(5) и *slapd-perl*(5)

11.10. Транслятор

11.10.1. Обзор

Основное назначение данного механизма манипуляции данными *slapd*(8) — отображение пространства имён, определённого в базе данных этого же экземпляра *slapd*(8), в виртуальное пространство имён, с выполнением, если это необходимо, манипуляций над типами атрибутов и объектными классами. Для его работы требуется наложение rwm.

Этот механизм и вышеупомянутое наложение являются экспериментальными.

11.10.2. Настройка back-relay

БУДЕТ ПОЗЖЕ

11.10.3. Дополнительная информация

slapd-relay(5)

11.11. SQL

11.11.1. Обзор

Основное назначение данного механизма манипуляции данными *slapd*(8) — представить информацию, хранящуюся в некоторой реляционной СУБД как поддерево LDAP без написания какого-либо программного кода (не будем принимать за программный код немного SQL и, возможно, хранимых процедур, верно?).

Где этому можно найти применение? К примеру, Вы (или некий поставщик услуг) храните учётные записи (адресные книги, телефонные справочники) в СУБД, и Вам хотелось бы использовать эти же данные в современных приложениях, ожидающих получить их из LDAP. Возможно, Вам нужно синхронизировать (или предоставить в общий доступ) информацию между различными сайтами/приложениями, использующими СУБД и/или LDAP. Или еще что-нибудь...

Отметим, что данный механизм **НЕ** разрабатывался в качестве полнофункционального механизма манипуляции данными, использующего реляционную СУБД вместо BerkeleyDB (то есть с полной поддержкой функциональности LDAP, как это делает стандартный механизм BDB). Его стоит рассматривать как механизм манипуляции данными с рядом ограничений. Дискуссию на этот счёт можно посмотреть в разделе Непростые взаимоотношения LDAP и реляционных СУБД.

Идея состоит в использовании некой мета-информации для трансляции LDAP-запросов в SQL-запросы,

оставляя нетронутой реляционную схему данных, чтобы старые приложения могли и дальше ее использовать без внесения в них изменений. Это позволяет SQL- и LDAP-приложениям взаимодействовать и обмениваться данными друг с другом без репликации.

Механизм SQL способен настраиваться под практически любую реляционную схему, без внесения в неё изменений (с помощью вышеупомянутой мета-информации). Для подключения к СУБД он использует ODBC, кроме того есть возможность подстройки под любой SQL-диалет, используемый в СУБД. Таким образом, он может быть использован для интеграции и распространения информации на любой СУБД, ОС, сетевом хосте и т.д., то есть в высокогетерогенной среде.

Данный механизм хранения и доступа к данным является экспериментальным.

11.11.2. Настройка back-sql

Настройка данного механизма одна из самых сложных и комплексных среди рассматриваемых нами. Поэтому здесь мы рассмотрим небольшой простой пример, идущий с дистрибутивом OpenLDAP, который можно найти в `servers/slapd/back-sql/rdbms_depend/README`

В этом примере будем использовать PostgreSQL.

Во-первых, добавим в `/etc/odbc.ini` такой блок настроек:

```
[example]                       <===
Description         = Example for OpenLDAP's back-sql
Driver              = PostgreSQL
Trace               = No
Database            = example    <===
Servername          = localhost
UserName            = manager    <===
Password            = secret     <===
Port                = 5432
;Protocol            = 6.4
ReadOnly            = No
RowVersioning       = No
ShowSystemTables    = No
ShowOidColumn       = No
FakeOidIndex        = No
ConnSettings        =
```

Информация, имеющая непосредственное отношение к нашему примеру, подсвечена стрелками '<===' справа.

Затем добавим в `/etc/odbcinst.ini` такой блок настроек:

```
[PostgreSQL]
Description     = ODBC for PostgreSQL
Driver          = /usr/lib/libodbcpsql.so
Setup           = /usr/lib/libodbcpsqlS.so
FileUsage       = 1
```

Предполагается, что Вам известно, как в PostgreSQL создать базу данных, учётную запись пользователя и назначить ей пароль. Также предполагается, что Вы сможете наполнить созданную Вами базу данных 'example' с помощью следующих файлов из директории `servers/slapd/back-sql/rdbms_depend/pgsql` :

```
backsql_create.sql, testdb_create.sql, testdb_data.sql, testdb_metadata.sql
```

Наконец, запустим тест:

```
[root@localhost]# cd $SOURCES/tests
[root@localhost]# SLAPD_USE_SQL=pgsql ./run sql-test000
```

На выходе будет что-то вроде этого (урезано в целях экономии места):

```
Cleaning up test run directory leftover from previous run.
Running ./scripts/sql-test000-read...
running defines.sh
Starting slapd on TCP/IP port 9011...
Testing SQL backend read operations...
Waiting 5 seconds for slapd to start...
Testing correct bind... dn:cn=Mitya Kovalev,dc=example,dc=com
Testing incorrect bind (should fail)... ldap_bind: Invalid credentials (49)
```

```
......

Filtering original ldif...
Comparing filter output...
>>>>> Test succeeded
```

Данный тест выполняется в режиме "только чтения" данных; он может быть применим к любой реляционной СУБД.

Другой тест (на запись), sql-test900-write, на сегодняшний момент может быть выполнен только в PostgreSQL и IBM db2.

Чтобы получить больше информации, посмотрите `sql-test000`, файлы в директории `servers/slapd/back-sql/rdbms_depend/pgsql/` и man-страницы.

Замечание: Этот механизм манипуляции данными является экспериментальным.

11.11.3. Дополнительная информация

slapd-sql(5) и `servers/slapd/back-sql/rdbms_depend/README`

К содержанию

12. Наложения

Наложения - это программные компоненты, переопределяющие поведение функций, выполняемых механизмами манипуляции данными. Они располагаются "над" механизмами манипуляции, перехватывая и перенаправляя им обращения, направленные от клиента, а также перехватывая ответы механизмов манипуляции и перенаправляя их клиентам. При этом информация, передаваемая и получаемая от механизма манипуляции данными, может изменяться, изменяя тем самым поведение механизмов манипуляции.

Наложения могут быть статически скомпилированы в *slapd*, или, если включена поддержка модулей, они могут быть загружены динамически. Большинство наложений применимы только к конкретным видам баз данных.

Некоторые из них могут применяться на глобальном уровне `frontend`, то есть после того, как запрос от клиента был принят, разобран и проверен, но еще перед выбором конкретного механизма манипуляции данными. Основная цель таких наложений - повлиять на выполнение запроса независимо от того, каким механизмом манипуляции он будет обработан, а также, в некоторых случаях, повлиять на выбор механизма манипуляции данными, модифицируя DN запроса.

Итак, наложения предназначены для:

- переопределения поведения существующих механизмов манипуляции данными без изменения их исходного кода, или без необходимости написания нового полнофункционального механизма манипуляции данными.
- переопределения общей функциональности, безотносительно конкретного типа механизма манипуляции данными.

При использовании *slapd.conf*(5), наложения, сконфигурированные до настроек какой-либо базы данных, считаются глобальными, как было описано выше. На самом деле они неявно накладываются на базу данных `frontend`. Их также можно наложить явно, прописав следующую конфигурацию:

```
database frontend
overlay <имя наложения>
```

Обычно документация к наложению помещается в отдельную man-страницу в секции 5; общепринятое именование таких страниц:

```
slapo-<имя наложения>
```

Все основные наложения, входящие в дистрибутив, имеют man-страницы. Вы можете смело их дополнять, если считаете, что что-то упущено в описании поведения компонента или применении соответствующих директив конфигурации.

Официально-поддерживаемые наложения располагаются в директории

```
servers/slapd/overlays/
```

В ней также сеть файл slapover.txt, в котором описаны области применения того или иного наложения, а также рекомендации по разработке собственных наложений.

Другие распространённые наложения расположены в

```
contrib/slapd-modules/<имя наложения>/
```

вместе с другими типами компонентов, загружаемых во время выполнения сервера; они официально распространяются, но не поддерживаются проектом.

Все текущие наложения в OpenLDAP перечислены и подробно описаны в следующих разделах.

12.1. Ведение журнала доступа

12.1.1. Обзор

Данное наложение записывает попытки доступа к определённой базе данных в другую базу данных.

Это позволяет выполнять различные выборки пользовательских обращений к целевой базе данных из любой точки сети с помощью разнообразных LDAP-запросов, а не просто просматривать обычные локальные текстовые файлы журналов. С помощью опций конфигурации можно настроить, какие типы операций доступа попадут в журнал, и когда следует автоматически чистить базу данных журнала от устаревших записей. Записи журнала хранятся как экземпляры объектных классов набора схемы audit для обеспечения их читабельности в виде файлов LDIF либо в форме ответов на поисковые запросы.

Данное наложение используется также для дельта-syncrepl репликации

12.1.2. Настройка наложения ведения журнала доступа

Вот простой пример применения журналирования доступа:

```
database bdb
suffix dc=example,dc=com
...
overlay accesslog
logdb cn=log
logops writes reads
logold (objectclass=person)

database bdb
suffix cn=log
...
index reqStart eq
access to *
  by dn.base="cn=admin,dc=example,dc=com" read
```

А следующий пример используется для дельта-syncrepl репликации:

```
database hdb
suffix cn=accesslog
directory /usr/local/var/openldap-accesslog
rootdn cn=accesslog
index default eq
index entryCSN,objectClass,reqEnd,reqResult,reqStart
```

Установки наложения accesslog для основной базы данных:

```
database bdb
suffix dc=example,dc=com
...
overlay accesslog
logdb cn=accesslog
logops writes
logsuccess TRUE
# проверять БД accesslog каждый день и удалять записи старше 7-ми дней
logpurge 07+00:00 01+00:00
```

Пример данных, возвращаемых поисковым запросом к ветви **cn=accesslog**:

```
[ghenry@suretec ghenry]# ldapsearch -x -b cn=accesslog
# extended LDIF
#
# LDAPv3
# base <cn=accesslog> with scope subtree
# filter: (objectclass=*)
# requesting: ALL
#

# accesslog
dn: cn=accesslog
objectClass: auditContainer
cn: accesslog

# 20080110163829.000004Z, accesslog
dn: reqStart=20080110163829.000004Z,cn=accesslog
objectClass: auditModify
reqStart: 20080110163829.000004Z
reqEnd: 20080110163829.000005Z
reqType: modify
reqSession: 196696
reqAuthzID: cn=admin,dc=suretecsystems,dc=com
reqDN: uid=suretec-46022f8$,ou=Users,dc=suretecsystems,dc=com
reqResult: 0
reqMod: sambaPwdCanChange:- ###CENSORED###
reqMod: sambaPwdCanChange:+ ###CENSORED###
reqMod: sambaNTPassword:- ###CENSORED###
reqMod: sambaNTPassword:+ ###CENSORED###
reqMod: sambaPwdLastSet:- ###CENSORED###
reqMod: sambaPwdLastSet:+ ###CENSORED###
reqMod: entryCSN:= 20080110163829.095157Z#000000#000#000000
reqMod: modifiersName:= cn=admin,dc=suretecsystems,dc=com
reqMod: modifyTimestamp:= 20080110163829Z

# search result
search: 2
result: 0 Success

# numResponses: 3
# numEntries: 2
```

12.1.3. Дополнительная информация

slapo-accesslog(5) и раздел дельта-syncrepl репликаций.

12.2. Ведение журнала изменений

Данное наложение может быть использовано для складирования всех изменений, произошедших в определённой базе данных, в указанный файл журнала.

12.2.1. Обзор

Если возникает потребность журналировать изменения в виде стандартного LDIF, есть возможность применить наложение auditlog. Подробные примеры его использования можно найти в man-странице **slapo-auditlog (5)**.

12.2.2. Настройка наложения ведения журнала изменений

Если служба каталогов настраивается через `slapd.d`, то можно использовать следующий LDIF, чтобы добавить наложение к списку наложений в **cn=config** и указать в какой файл будет записываться журнал в

формате LDIF (измените путь и название файла под Ваши нужды):

```
dn: olcOverlay=auditlog,olcDatabase={1}hdb,cn=config
changetype: add
objectClass: olcOverlayConfig
objectClass: olcAuditLogConfig
olcOverlay: auditlog
olcAuditlogFile: /tmp/auditlog.ldif
```

В этом тестовом примере мы складываем изменения в файл `/tmp/auditlog.ldif`

Типичный LDIF-файл, создаваемый **slapo-auditlog(5)**, выглядит следующим образом:

```
# add 1196797576 dc=suretecsystems,dc=com cn=admin,dc=suretecsystems,dc=com
dn: dc=suretecsystems,dc=com
changetype: add
objectClass: dcObject
objectClass: organization
dc: suretecsystems
o: Suretec Systems Ltd.
structuralObjectClass: organization
entryUUID: 1606f8f8-f06e-1029-8289-f0cc9d81e81a
creatorsName: cn=admin,dc=suretecsystems,dc=com
modifiersName: cn=admin,dc=suretecsystems,dc=com
createTimestamp: 20051123130912Z
modifyTimestamp: 20051123130912Z
entryCSN: 20051123130912.000000Z#000001#000#000000
auditContext: cn=accesslog
# end add 1196797576

# add 1196797577 dc=suretecsystems,dc=com cn=admin,dc=suretecsystems,dc=com
dn: ou=Groups,dc=suretecsystems,dc=com
changetype: add
objectClass: top
objectClass: organizationalUnit
ou: Groups
structuralObjectClass: organizationalUnit
entryUUID: 160aaa2a-f06e-1029-828a-f0cc9d81e81a
creatorsName: cn=admin,dc=suretecsystems,dc=com
modifiersName: cn=admin,dc=suretecsystems,dc=com
createTimestamp: 20051123130912Z
modifyTimestamp: 20051123130912Z
entryCSN: 20051123130912.000000Z#000002#000#000000
# end add 1196797577
```

12.2.3. Дополнительная информация

slapo-auditlog(5)

12.3. Сцепление

12.3.1. Обзор

Это наложение обеспечивает возможность простого сцепления между серверами, обслуживающими базу данных LDAP.

Что такое сцепление? Оно представляет собою возможность DSA следовать по отсылкам LDAP от имени клиентов, которые сами "перейти" по этим отсылкам не могут. Таким образом, с точки зрения клиента, распределённая система представляется как единственный виртуальный DSA.

Наложение сцепления накладывается поверх механизма манипуляции данными ldap; оно компилируется автоматически, если при запуске configure была указана опция **--enable-ldap**.

12.3.2. Настройка наложения сцепления

Чтобы продемонстрировать работу данного наложения, рассмотрим типичный случай с использованием одного основного сервера и трёх подчиненных серверов с Syncrepl-репликами.

На каждой реплике в начале файла *slapd.conf*(5) (глобальные настройки, до определения какой-либо базы

данных) добавляются следующие строки:

```
overlay                 chain
chain-uri               "ldap://ldapmaster.example.com"
chain-idassert-bind     bindmethod="simple"
                        binddn="cn=Manager,dc=example,dc=com"
                        credentials="<secret>"
                        mode="self"
chain-tls               start
chain-return-error      TRUE
```

В настройках *syncrepl* добавляется:

```
updateref               "ldap://ldapmaster.example.com/"
```

Директива **chain-tls** включает TLS между подчинённым и основным ldap-сервером. Поскольку на всех серверах одинаковые DIT, DN пользователя, подключающегося к подчинённому серверу, существует и на основном сервере. Если на основном сервере у этого DN нет прав на обновление, операция будет просто проигнорирована и ничего не произойдёт.

После внесения изменений в *slapd.conf* требуется перезапустить демоны на подчинённых серверах (при использовании *cn=config* перезапуск не требуется). После этого, если Вы используете уровень журналирования *stats* (256), Вы можете отследить изменения, вносимые, например, с помощью *ldapmodify*, на подчинённых и основном серверах.

Итак, запустим *ldapmodify* на подчинённом сервере и посмотрим записи журнала. Там будет что-то вроде:

```
Sep  6 09:27:25 slave1 slapd[29274]: conn=11 fd=31 ACCEPT from IP=143.199.102.216:45181
(IP=143.199.102.216:389)
Sep  6 09:27:25 slave1 slapd[29274]: conn=11 op=0 STARTTLS
Sep  6 09:27:25 slave1 slapd[29274]: conn=11 op=0 RESULT oid= err=0 text=
Sep  6 09:27:25 slave1 slapd[29274]: conn=11 fd=31 TLS established tls_ssf=256 ssf=256
Sep  6 09:27:28 slave1 slapd[29274]: conn=11 op=1 BIND dn="uid=user1,ou=people,dc=example,dc=com"
method=128
Sep  6 09:27:28 slave1 slapd[29274]: conn=11 op=1 BIND dn="uid=user1,ou=People,dc=example,dc=com"
mech=SIMPLE ssf=0
Sep  6 09:27:28 slave1 slapd[29274]: conn=11 op=1 RESULT tag=97 err=0 text=
Sep  6 09:27:28 slave1 slapd[29274]: conn=11 op=2 MOD dn="uid=user1,ou=People,dc=example,dc=com"
Sep  6 09:27:28 slave1 slapd[29274]: conn=11 op=2 MOD attr=mail
Sep  6 09:27:28 slave1 slapd[29274]: conn=11 op=2 RESULT tag=103 err=0 text=
Sep  6 09:27:28 slave1 slapd[29274]: conn=11 op=3 UNBIND
Sep  6 09:27:28 slave1 slapd[29274]: conn=11 fd=31 closed
Sep  6 09:27:28 slave1 slapd[29274]: syncrepl_entry: LDAP_RES_SEARCH_ENTRY(LDAP_SYNC_MODIFY)
Sep  6 09:27:28 slave1 slapd[29274]: syncrepl_entry: be_search (0)
Sep  6 09:27:28 slave1 slapd[29274]: syncrepl_entry: uid=user1,ou=People,dc=example,dc=com
Sep  6 09:27:28 slave1 slapd[29274]: syncrepl_entry: be_modify (0)
```

А на основном сервере будет такое:

```
Sep  6 09:23:57 ldapmaster slapd[2961]: conn=55902 op=3 PROXYAUTHZ
dn="uid=user1,ou=people,dc=example,dc=com"
Sep  6 09:23:57 ldapmaster slapd[2961]: conn=55902 op=3 MOD
dn="uid=user1,ou=People,dc=example,dc=com"
Sep  6 09:23:57 ldapmaster slapd[2961]: conn=55902 op=3 MOD attr=mail
Sep  6 09:23:57 ldapmaster slapd[2961]: conn=55902 op=3 RESULT tag=103 err=0 text=
```

Замечание: Обратите внимание на строку PROXYAUTHZ в записях журнала основного сервера, она говорит о использовании верных идентификационных данных при выполнении операции обновления на основном сервере. Также имейте ввиду, что подчинённые сервера немедленно получают Syncrepl-обновления с основного сервера.

12.3.3. Обработка ошибок сцепления

По умолчанию, если операция обновления при сцеплении окончилось неудачей, клиенту возвращается оригинальная отсылка для того, чтобы он мог сам попробовать перейти по этой отсылке.

Однако, при использовании следующей директивы, если операция обновления при сцеплении окончилось неудачей на стороне сервера, клиенту возвращается актуальная ошибка.

```
chain-return-error TRUE
```

12.3.4. Чтение изменений, внесенных с использованием сцепления

Иногда приложениям требуется заново прочитать данные, которые они только что записали. Если запрос на изменение данных на подчиненном сервере был перенаправлен на основной сервер, немедленный запрос данных с подчиненного сервера может привести к тому, что будут получены данные, которые еще не были синхронизированы. В подобных случаях клиентам нужно использовать элемент управления **dontusecopy** для получения актуальных данных напрямую с основного сервера.

Обычно, при получении элемента управления **dontusecopy**, клиенту возвращается отсылка на актуальный источник данных. Однако, при использовании наложения *slapo-chain(5)*, оно перехватывает такие отсылки и пытается самостоятельно получить запрашиваемые данные.

12.3.5. Дополнительная информация

slapo-chain(5)

12.4. Ограничения на значения

12.4.1. Обзор

Данное наложение создаёт ограничения посредством регулярных выражений на все значения указанного атрибута во время выполнения LDAP-запросов, содержащих команды добавления или изменения данных. Оно используется для обеспечения более строгого синтаксиса, когда основной синтаксис атрибута является слишком общим.

12.4.2. Настройка наложения ограничений

Пример конфигурации через *slapd.conf*(5):

```
       overlay constraint
       constraint_attribute mail regex ^[[:alnum:]]+@mydomain.com$
       constraint_attribute title uri ldap:///dc=catalog,dc=example,dc=com?title?sub?
(objectClass=titleCatalog)
```

С подобными настройками будет отклоняться любой атрубут *mail*, содержимое которого не соответствует *<буквенно-цифровая строка>@mydomain.com*.

Также будет отклоняться любой атрибут title, значения которого не встречаются в атрибутах title любых записей *titleCatalog* в указанной области поиска.

Тот же пример с использованием *cn=config*:

```
      dn: olcOverlay=constraint,olcDatabase={1}hdb,cn=config
      changetype: add
      objectClass: olcOverlayConfig
      objectClass: olcConstraintConfig
      olcOverlay: constraint
      olcConstraintAttribute: mail regex ^[[:alnum:]]+@mydomain.com$
      olcConstraintAttribute: title uri ldap:///dc=catalog,dc=example,dc=com?title?sub?
(objectClass=titleCatalog)
```

12.4.3. Дополнительная информация

slapo-constraint(5)

12.5. Динамические службы каталогов (Dynamic Directory Services)

12.5.1. Обзор

Наложение *dds* реализует в *slapd*(8) динамические объекты в соответствии с RFC2589. Название *dds* - акроним от Dynamic Directory Services (динамические службы каталогов). Данное наложение позволяет определить динамические объекты, описываемые объектным классом *dynamicObject*.

Время жизни динамических объектов ограничено и определяется так называемым time-to-live (TTL, время жизни), которое может быть обновлено с помощью специальной операции расширенного обновления. Эта операция позволяет задать Client Refresh Period (CRP, клиентский период обновления), а именно промежуток времени между обновлениями, необходимый для предохранения динамического объекта от истечения срока его действия. Срок действия вычисляется путём добавления запрашиваемого TTL к текущему времени. Если объект достигает конца срока жизни и этот срок не был дополнительно обновлён, то такой объект автоматически *удаляется*. Однако нет гарантии, что объект будет удалён немедленно, поэтому клиенты не должны на это рассчитывать.

12.5.2. Настройка наложения динамических служб каталогов

Рассмотрим использование динамических объектов на примере реализации проведения динамических собраний. Для определённости установим, что участники собрания могут обновлять объект собрания, но только создатель данного объекта собрания может его удалить (либо он удалится по истечении TTL).

Применяя данное наложение к базе данных нашего примера, установим максимальное время жизни объекта в 1 день, минимальное - 10 секунд, а время жизни по умолчанию - 1 час. Мы также установим интервал проверок устаревания объекта в 120 секунд (менее 60-ти будет слишком мало), и срок толерантности в 5 секунд (фактическое время жизни объекта таким образом будет *entryTtl + срок толерантности*).

```
overlay dds
dds-max-ttl     1d
dds-min-ttl     10s
dds-default-ttl 1h
dds-interval    120s
dds-tolerance   5s
```

и добавим индекс:

```
entryExpireTimestamp
```

Создадим собрание следующей записью:

```
dn: cn=OpenLDAP Documentation Meeting,ou=Meetings,dc=example,dc=com
objectClass: groupOfNames
objectClass: dynamicObject
cn: OpenLDAP Documentation Meeting
member: uid=ghenry,ou=People,dc=example,dc=com
member: uid=hyc,ou=People,dc=example,dc=com
```

12.5.2.1. Контроль доступа к динамическим службам каталога

Разрешим пользователям создавать новое собрание и присоединяться к существующим; операции обновления TTL разрешим только пользователям, перечисленным в атрибутах *member*; создателю разрешим удаление объекта:

```
access to attrs=userPassword
   by self write
   by * read

access to dn.base="ou=Meetings,dc=example,dc=com"
          attrs=children
     by users write

access to dn.onelevel="ou=Meetings,dc=example,dc=com"
          attrs=entry
     by dnattr=creatorsName write
     by * read

access to dn.onelevel="ou=Meetings,dc=example,dc=com"
          attrs=participant
     by dnattr=creatorsName write
     by users selfwrite
```

```
        by * read

    access to dn.onelevel="ou=Meetings,dc=example,dc=com"
            attrs=entryTtl
        by dnattr=member manage
        by * read
```

Проще говоря, пользователь, создавший *OpenLDAP Documentation Meeting*, может добавлять новых участников, обновлять время жизни собрания (практически, полный контроль):

```
    ldapexop -x -H ldap://ldaphost "refresh" "cn=OpenLDAP Documentation
Meeting,ou=Meetings,dc=example,dc=com" "120" -D "uid=ghenry,ou=People,dc=example,dc=com" -W
```

Любой пользователь может присоединиться к собранию, но не может добавить другого участника; кроме того, участники могут обновлять время жизни собрания. Приведенные выше ACL достаточно прямолинейны и понятны, чтобы в этом разобраться.

12.5.3. Дополнительная информация

slapo-dds(5)

12.6. Динамические группы

12.6.1. Обзор

Данное наложение расширяет операции сравнения для выявления членов динамической группы. В настоящий момент оно устарело, поскольку все его функции реализованы в наложении Динамические списки.

12.7. Динамические списки

12.7.1. Обзор

Данное наложение даёт возможность расширить записи каталога динамическими группами и списками. Вместо того, чтобы реально хранить членов группы или список значений атрибута для определённой записи каталога, оно позволяет определить поисковый LDAP-запрос, результаты которого будут представлены в качестве такой группы или списка.

12.7.2. Настройка наложения динамических списков

В зависимости от конфигурации, наложение может вести себя и как динамический список, и как динамическая группа. Синтаксис следующий:

```
    overlay dynlist
    dynlist-attrset <group-oc> <URL-ad> [member-ad]
```

Параметры директивы `dynlist-attrset` имеют следующие значения:

- `<group-oc>`: определяет присутствие какого объектного класса вызывает последующий поисковый LDAP-запрос. Всякий раз, когда извлекается запись с данным объектным классом, выполняется соответствующий поиск.
- `<URL-ad>`: определяет имя атрибута, содержащего поисковый URI. Он должен быть подтипом типа `labeledURI`. Если не указан параметр `member-ad` (смотрите ниже), найденные в результате поискового запроса атрибуты и их значения добавляются к записи.
- `member-ad`: необязательный параметр. Если он присутствует, наложение ведёт себя как динамическая группа. Вместо добавления к исходной записи результата поискового запроса, к ней добавляются DN найденных записей как значения указанного в этом параметре атрибута.

Следующий пример создаёт почтовый псевдоним, ссылающийся автоматически на адреса электронной почты всех пользователей из диапазона, заданного фильтрами нашего поискового LDAP-запроса:

В файле *slapd.conf*(5) укажем следующее:

```
overlay dynlist
dynlist-attrset nisMailAlias labeledURI
```

Это значит, что при извлечении записи, содержащей объектный класс `nisMailAlias`, будет выполняться поисковый запрос по URI, указанному в атрибуте `labeledURI`.

Допустим, в нашем каталоге есть такая запись:

```
cn=all,ou=aliases,dc=example,dc=com
cn: all
objectClass: nisMailAlias
labeledURI: ldap:///ou=People,dc=example,dc=com?mail?one?(objectClass=inetOrgPerson)
```

При её извлечении будет выполнен поисковый запрос согласно значению `labeledURI`, и его результаты будут подставлены к записи, как будто они всегда тут и были. В данном случае поисковый фильтр выбирает все записи на один уровень ниже `ou=People`, имеющие объектный класс `inetOrgPerson`, и извлекает из них атрибут `mail`, если таковой имеется.

Вот что добавится к нашей записи, если в ветви `ou=People` нашлись две записи о пользователях, удовлетворяющие фильтру:

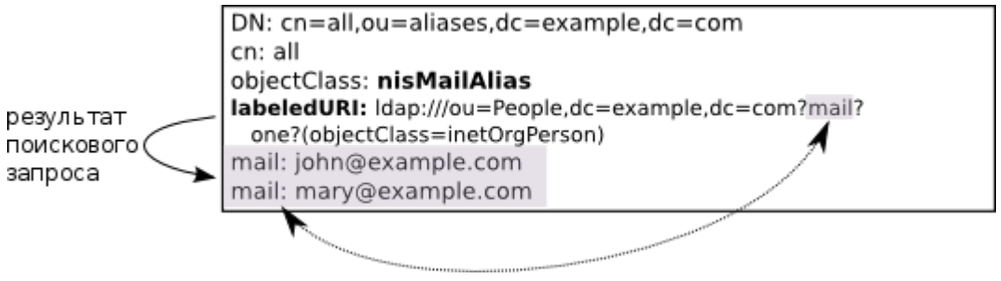

Рисунок 12.1: Динамический список всех адресов электронной почты

Конфигурация для динамической группы аналогична. Рассмотрим пример, автоматически заполняющий группу `allusers` group всеми учетными записями пользователей из определённой ветви каталога.

В файле *slapd.conf*(5) укажем следующее:

```
include /path/to/dyngroup.schema
...
overlay dynlist
dynlist-attrset groupOfURLs labeledURI member
```

Замечание: Для нашего примера нужно подключить файл `dyngroup.schema`, в котором определяется объектный класс `groupOfURLs`.

Попробуем применить наши настройки к следующей записи:

```
cn=allusers,ou=group,dc=example,dc=com
cn: all
objectClass: groupOfURLs
labeledURI: ldap:///ou=people,dc=example,dc=com??one?(objectClass=inetOrgPerson)
```

В данном случае всё происходит аналогично предыдущему примеру с динамическими списками: при извлечении записи с объектным классом `groupOfURLs`, выполняется поисковый запрос, определённый в атрибуте `labeledURI`. Но на этот раз получаются только DN найденных записей, и они добавляются к нашей исходной записи как значения атрибута `member`.

Вот что мы получим:

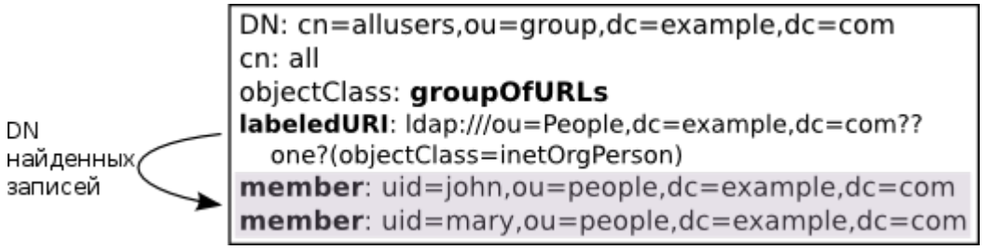

Рисунок 12.2: Динамическая группа всех пользователей

Заметьте, что побочным эффектом этой схемы динамических групп является то, что члены группы должны быть указаны своими полными DN. Так что если Вы планируете использовать данную схему для записей с объектным классом `posixGroup`, убедитесь, что Вы используете RFC2307bis и такие типы атрибутов, которые могут содержать DN. Например атрибут `memberUid` объектного класса `posixGroup` может содержать только имена, а не DN, поэтому для динамических групп он не подходит.

12.7.3. Дополнительная информация

slapo-dynlist(5)

12.8. Обратное обслуживание членства в группах

12.8.1. Обзор

В некоторых случаях клиентам желательно определить, в каких группах состоит та или иная запись, не прибегая к дополнительным запросам. Примером могут служить приложения, использующие DIT для контроля доступа на основе групповой авторизации.

Наложение **memberof** обновляет атрибут (по умолчанию **memberOf**) записи-члена всякий раз при изменении атрибута членства (по умолчанию **member**) записей-групп с теми объектными классами, которые определены для отслеживания подобных обновлений (по умолчанию **groupOfNames**).

Таким образом, наложение обеспечивает обслуживание списка групп, членом которых является данная запись, если обслуживание групп осуществляется изменением атрибутов членства записи-группы.

12.8.2. Настройка наложения членства в группах

Типичная настройка данного наложения заключается в простом добавлении его к нужной базе данных. Например, имея такие минимальные настройки в slapd.conf:

```
include /usr/share/openldap/schema/core.schema
include /usr/share/openldap/schema/cosine.schema

authz-regexp "gidNumber=0\\\+uidNumber=0,cn=peercred,cn=external,cn=auth"
        "cn=Manager,dc=example,dc=com"
database        bdb
suffix          "dc=example,dc=com"
rootdn          "cn=Manager,dc=example,dc=com"
rootpw          secret
directory       /var/lib/ldap2.4
checkpoint 256 5
index   objectClass   eq
index   uid           eq,sub

overlay memberof
```

добавим в каталог следующий ldif:

```
cat memberof.ldif
dn: dc=example,dc=com
objectclass: domain
dc: example

dn: ou=Group,dc=example,dc=com
```

```
objectclass: organizationalUnit
ou: Group

dn: ou=People,dc=example,dc=com
objectclass: organizationalUnit
ou: People

dn: uid=test1,ou=People,dc=example,dc=com
objectclass: account
uid: test1

dn: cn=testgroup,ou=Group,dc=example,dc=com
objectclass: groupOfNames
cn: testgroup
member: uid=test1,ou=People,dc=example,dc=com
```

Результат поискового запроса для пользователя test1:

```
# ldapsearch -LL -Y EXTERNAL -H ldapi:/// "(uid=test1)" -b dc=example,dc=com memberOf
SASL/EXTERNAL authentication started
SASL username: gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth
SASL SSF: 0
version: 1

dn: uid=test1,ou=People,dc=example,dc=com
memberOf: cn=testgroup,ou=Group,dc=example,dc=com
```

Обратите внимание, что атрибут **memberOf** - это операционный атрибут, поэтому он должен быть запрошен явно.

12.8.3. Дополнительная информация

slapo-memberof(5)

12.9. Кэширующий прокси-сервер

Серверы LDAP обычно содержат одно или несколько поддеревьев DIT. Серверы-реплики (или теневые серверы) содержат теневые копии записей с одного или нескольких основных серверов. Изменения распространяются с основного сервера на подчинённые серверы (реплики) посредством LDAP Sync-репликаций. LDAP-кэш - это специальный тип реплики, содержащий записи, соответствующие поисковым фильтрам, а не поддеревьям.

12.9.1. Обзор

Расширение slapd кэширующий прокси-сервер предназначено для улучшения быстродействия механизмов манипуляции данными ldap и meta. Оно обрабатывает поисковые запросы и сначала определяет, соответствуют ли его поисковые фильтры тем, записи по которым уже закэшированы. Если записи по запросу содержатся в локальной базе данных прокси-кэша, они извлекаются оттуда и возвращаются в ответ на запрос. Остальные запросы пропускаются дальше в нужный механизм манипуляции данными ldap или meta, где обрабатываются обычным способом.

Например, записи по запросу `(shoesize>=9)` уже содержатся в записях по запросу `(shoesize>=8)`, а записи по запросу `(sn=Richardson)` уже содержатся в записях по запросу `(sn=Richards*)`

При сравнении утверждений, для принятия решения о сохранении результата запроса в кэше, используются установленные правила соответствия и синтаксисы. Для упрощения проблемы сохранения информации в кэше, во время конфигурации определяется список "шаблонов" запросов, которые могут быть закэшированы (смотрите описание ниже). Результаты запроса помещаются в кэш и извлекаются оттуда только если запрос соответствует одному из таких шаблонов. Записи, связанные с закэшированными запросами, хранятся в локальной базе данных прокси-кэша, а ассоциированная с ними мета-информация (база и диапазон поиска, фильтры, атрибуты) содержится непосредственно в оперативной памяти.

Шаблон - это прототип для образования поискового запроса LDAP. Шаблоны задаются прототипом поискового фильтра и списком атрибутов, которые потребуются в запросах, образовываемых из шаблона. Представление прототипа поискового фильтра аналогично RFC4515, за исключением того, что конкретно

заданные значения атрибутов опускаются. Примеры прототипов поисковых фильтров: (sn=),(&(sn=)(givenname=)), соответствующие им экземпляры поисковых фильтров: (sn=Doe) и (&(sn=Doe)(givenname=John)).

Политика обновления кэша удаляет запросы, которые давно не запрашивались повторно, и соответствующие им записи. Кроме того, закэшированным запросам назначается максимальное время жизни (TTL), чтобы, по возможности, избежать несоответствия данных из кэша реальным данным целевого сервера. Фоновый процесс периодически проверяет кэш на наличие устаревших записей и удаляет их.

На странице Кэширующего Прокси-сервера (http://www.openldap.org/pub/kapurva/proxycaching.pdf) представлены детали проектирования и реализации данной технологии.

12.9.2. Настройка наложения кэширующего прокси-сервера

Описанные ниже директивы, специфичные для кэширующего прокси, должны указываться после директивы `overlay pcache` в разделе `database meta` или `database ldap` файла *slapd.conf*(5).

12.9.2.1. Задание параметров кэша

```
pcache <DB> <maxentries> <nattrsets> <entrylimit> <period>
```

Данная директива активирует прокси-кэширование и задаёт основные параметры кэша. Параметр <DB> указывает тип базы данных, которая будет использоваться для хранения кэшированных записей. Можно задать либо `bdb`, либо `hdb`. Параметр <maxentries> указывает общее количество записей, которые могут содержаться в кэше. Параметр The <nattrsets> указывает общее количество наборов атрибутов (задаваемых директивой `pcacheAttrSet`), которые можно определить. Параметр <entrylimit> указывает максимальное количество записей в запросе, который может быть закэширован. Наконец, параметр <period> указывает период проверки целостности (в секундах). Всякий раз по истечении этого периода, запросы с устарешими TTL удаляются.

12.9.2.2. Определение наборов атрибутов

```
pcacheAttrset <index> <attrs...>
```

Данная директива нужна для ассоциации наборов атрибутов с индексами. Каждый набор атрибутов ассоциируется с числовым индексом от 0 до <numattrsets>-1. Эти индексы используются директивой proxyTemplate для определения шаблонов запросов, которые могут быть закэшированы.

12.9.2.3. Определение шаблонов запросов, которые могут быть закэшированы

```
pcacheTemplate <prototype_string> <attrset_index> <TTL>
```

Данная директива задаёт шаблон запросов, которые могут быть закэшированы и "время жизни" в секундах (параметр <TTL>) тех запросов, которые попадают под этот шаблон. Шаблон описывается строкой прототипа поискового фильтра и набором запрашиваемых атрибутов, определяемым индексом в параметре <attrset_index>.

12.9.2.4. Пример с использованием slapd.conf

Вот пример раздела database файла *slapd.conf*(5) для организации кэширующего прокси-сервера для поддерева `"dc=example,dc=com"`, содержащегося на сервере `ldap.example.com`.

```
database        ldap
suffix          "dc=example,dc=com"
rootdn          "dc=example,dc=com"
uri             ldap://ldap.example.com/
overlay pcache
pcache          hdb 100000 1 1000 100
pcacheAttrset   0 mail postaladdress telephonenumber
pcacheTemplate (sn=) 0 3600
pcacheTemplate (&(sn=)(givenName=)) 0 3600
pcacheTemplate (&(departmentNumber=)(secretary=*)) 0 3600

cachesize 20
directory ./testrun/db.2.a
```

```
index       objectClass eq
index       cn,sn,uid,mail  pres,eq,sub
```

12.9.2.5. Пример с использованием slapd-config

Пример аналогичного LDIF-файла для back-config, настраивающий кэширующий прокси сервер для поддерева `"dc=example,dc=com"`, хранящегося на сервере `ldap.example.com`.

```
dn: olcDatabase={2}ldap,cn=config
objectClass: olcDatabaseConfig
objectClass: olcLDAPConfig
olcDatabase: {2}ldap
olcSuffix: dc=example,dc=com
olcRootDN: dc=example,dc=com
olcDbURI: "ldap://ldap.example.com"

dn: olcOverlay={0}pcache,olcDatabase={2}ldap,cn=config
objectClass: olcOverlayConfig
objectClass: olcPcacheConfig
olcOverlay: {0}pcache
olcPcache: hdb 100000 1 1000 100
olcPcacheAttrset: 0 mail postalAddress telephoneNumber
olcPcacheTemplate: "(sn=)" 0 3600 0 0 0
olcPcacheTemplate: "(&(sn=)(givenName=))" 0 3600 0 0 0
olcPcacheTemplate: "(&(departmentNumber=)(secretary=))" 0 3600

dn: olcDatabase={0}hdb,olcOverlay={0}pcache,olcDatabase={2}ldap,cn=config
objectClass: olcHdbConfig
objectClass: olcPcacheDatabase
olcDatabase: {0}hdb
olcDbDirectory: ./testrun/db.2.a
olcDbCacheSize: 20
olcDbIndex: objectClass eq
olcDbIndex: cn,sn,uid,mail  pres,eq,sub
```

12.9.2.4.1. Запросы, которые могут быть закэшированы

Поисковый запрос LDAP может быть закэширован, если его фильтр соответствует одному из шаблонов, определённых директивами pcacheTemplate, и если запрашиваемые в нём атрибуты указаны в соответствующем наборе атрибутов. В приведённом выше примере набор атрибутов номер 0 определяет, что только атрибуты `mail postaladdress telephonenumber` могут быть закэшированы в идущих следом pcacheTemplate.

12.9.2.4.2. Примеры:

```
Filter: (&(sn=Richard*)(givenName=jack))
Attrs: mail telephoneNumber
```

может быть закэширован, поскольку он соответствует шаблону `(&(sn=)(givenName=))` и его атрибуты содержатся в pcacheAttrset 0.

```
Filter: (&(sn=Richard*)(telephoneNumber))
Attrs: givenName
```

не может быть закэширован, поскольку фильтр не соответствует шаблону, и атрибут givenName не задан для хранения в кэше.

```
Filter: (|(sn=Richard*)(givenName=jack))
Attrs: mail telephoneNumber
```

не может быть закэширован, поскольку фильтр не соответствует шаблону (условие логического ИЛИ "|" вместо логического И "&").

12.9.3. Дополнительная информация

slapo-pcache(5)

12.10. Политики паролей

12.10.1. Обзор

Цель данного наложения - исполнение требований проектного RFC под названием draft-behera-ldap-password-policy-09. Несмотря на то, что данный проект не получил официального статуса, он был реализован в нескольких серверах службы каталогов, в том числе и в slapd. Тем не менее, важно отметить, что это всего-лишь проектный RFC, работы по нему еще ведутся и, возможно, будут внесены какие-либо изменения.

Ключевые возможности наложения политик паролей следующие:

- Установка минимально допустимой длины при создании пароля
- Предотвращение слишком частой смены пароля
- Механизм устаревания паролей, предупреждающий пользователей о необходимости смены пароля до определённого срока, и предоставляющий им фиксированное количество подключений "последнего шанса", чтобы они могли сменить пароль после того, как он устареет
- Ведение истории паролей для исключения их повторного использования
- Предотвращение подбора пароля путём его блокировки на определённый период времени после определённого количества неверных попыток аутентификации
- Требование принудительной смены пароля при следующей аутентификации
- Установка административной блокировки на учётную запись
- Поддержка нескольких политик паролей отдельно для каждого объекта или по умолчанию
- Выполнение произвольной проверки качества паролей с использованием внешних загружаемых модулей. Это нестандартное расширение проектного RFC.

12.10.2. Настройка наложения политик паролей

Перед подключением наложения к той базе данных, где оно будет использоваться, нужно добавить новый набор схемы ppolicy и загрузить модуль ppolicy. В следующем примере демонстрируется подключение наложения ppolicy к базе данных, обслуживающей пространство имён "dc=example,dc=com". В этом примере мы также указываем DN объекта политики по умолчанию, которая применяется, если для объекта учётной записи пользователя не была определена никакая другая политика.

```
database bdb
suffix "dc=example,dc=com"
[... здесь указываются другие директивы конфигурации базы данных ...]

overlay ppolicy
ppolicy_default "cn=default,ou=policies,dc=example,dc=com"
```

Теперь мы должны создать контейнер для объектов политик. В нашем примере объекты политик паролей будут размещаться в ветви, называемой "ou=policies,dc=example,dc=com":

```
dn: ou=policies,dc=example,dc=com
objectClass: organizationalUnit
objectClass: top
ou: policies
```

Создадим объект политики по умолчанию и определим в нём следующие политики:

- Пользователю разрешено менять свой пароль (pwdAllowUserChange: TRUE). Обратите внимание, что ACL для атрибута userPassword также может влиять на эту способность.
- Имя атрибута пароля - "userPassword" (pwdAttribute: userPassword). Обратите внимание, что это единственное значение для данного атрибута, которое принимает OpenLDAP.
- Сервер будет проверять синтаксис пароля. Если сервер не может проверить синтаксис (например, он был захэширован или иным способом закодирован клиентом) он вернёт an error refusing the password (pwdCheckQuality: 2).
- Когда клиент при запросе на подключение использует элемент управления Password Policy Request (запрос политики пароля), то, если до момента устаревания пароля остаётся менее 10 минут, сервер отправит клиенту предупреждение о грядущем устаревании пароля (pwdExpireWarning: 600). Сами предупреждения отправляются в элементе управления Password Policy Response (отклик политики пароля).
- Когда срок действия пароля для DN истечёт, сервер позволит сделать пять дополнительных попыток подключений "последнего шанса" (pwdGraceAuthNLimit: 5).
- Сервер будет вести историю последних пяти паролей, которые были использованы для DN (pwdInHistory: 5).

- Сервер будет блокировать учётную запись после превышения максимального количества попыток подключения (pwdLockout: TRUE).
- Если сервер заблокировал учётную запись, она будет заблокирована им до тех пор, пока администратор не разблокирует её (pwdLockoutDuration: 0).
- Сервер будет сбрасывать счётчик неудачных попыток подключений после промежутка в 30 секунд (pwdFailureCountInterval: 30).
- Устаревание паролей не отслеживается (pwdMaxAge: 0).
- Пароли могут быть изменены как угодно часто (pwdMinAge: 0).
- Длина пароля не менее 5 символов (pwdMinLength: 5).
- Не требуется изменять пароль после первого подключения или после сброса пароля администратором (pwdMustChange: FALSE)
- Не требуется запрашивать текущий пароль при запросе на изменение пароля (pwdSafeModify: FALSE)
- Разрешено не более пяти неудачных попыток подключения для конкретного DN (pwdMaxFailure: 5).

Итак, наш объект политики получился такой:

```
dn: cn=default,ou=policies,dc=example,dc=com
cn: default
objectClass: pwdPolicy
objectClass: person
objectClass: top
pwdAllowUserChange: TRUE
pwdAttribute: userPassword
pwdCheckQuality: 2
pwdExpireWarning: 600
pwdFailureCountInterval: 30
pwdGraceAuthNLimit: 5
pwdInHistory: 5
pwdLockout: TRUE
pwdLockoutDuration: 0
pwdMaxAge: 0
pwdMaxFailure: 5
pwdMinAge: 0
pwdMinLength: 5
pwdMustChange: FALSE
pwdSafeModify: FALSE
sn: dummy value
```

При необходимости можно создать и другие объекты политик.

Есть 2 пути применения политик паролей к индивидуальным объектам:

1. Атрибут pwdPolicySubentry в объекте учётной записи пользователя. Если этот атрибут присутствует и в нём указан DN объекта политики, то данная политика применяется к целевому объекту учётной записи.

2. Политика паролей по умолчанию. Если в учётной записи не задан атрибут pwdPolicySubentry, и наложение политик паролей было сконфигурировано с указанием DN объекта политики по умолчанию, а также если такой объект присутствует, тогда указанная в этом объекте политика применяется к учетной записи пользователя.

Полностью возможности данного наложения описаны в *slapo-ppolicy(5)*. Также посмотрите дискуссию "Вопросы управления паролями" по адресу http://www.symas.com/blog/?page_id=66

12.10.3. Дополнительная информация

slapo-ppolicy(5)

12.11. Обеспечение целостности ссылок

12.11.1. Обзор

Данное наложение может использоваться поверх механизмов манипуляции данными, таких как slapd-bdb(5), для поддержания целостности схемы данных, использующей атрибуты-ссылки.

При выполнении операций *modrdn* или *delete*, то есть при переименовании DN записи или удалении записи, сервер будет искать ссылки на данный DN в записях каталога (в указанных атрибутах, смотрите описание

ниже) и обновлять их соответственно. Если выполнялась операция *delete*, ссылка будет удалена. Если выполнялась операция *modrdn*, то в ссылочный атрибут будет записан новый DN.

Примером может служить очень распространённая административная задача ведения списка членов группы во время манипуляций с учётными записями пользователей. Когда учётная запись пользователя удаляется или переименовывается, все записи групп, где этот пользователь состоит членом, должны быть обновлены соответственно. Для решения этой задачи администраторы LDAP обычно пишут свои скрипты. Но мы можем использовать наложение `refint` для автоматизации этой задачи. В этом примере, если пользователь удаляется из каталога, наложение позаботится об удалении пользователя из группы, где он состоял. И не надо никаких скриптов.

12.11.2. Настройка наложения обеспечения целостности ссылок

Настройки этого наложения следующие:

```
overlay refint
refint_attributes <attribute [attribute ...]>
refint_nothing <string>
```

- Параметр `refint_attributes` задаёт разделённый пробелами список атрибутов, которые будут проверяться на целостность ссылок. Когда запись удаляется или её DN переименовывается, сервер выполнит внутренний поиск по атрибутам `refint_attributes`, которые указывают удаляемый/изменяемый DN и соответственно изменит значения этих атрибутов. ВАЖНОЕ ЗАМЕЧАНИЕ: перечисленные в этом параметре атрибуты должны иметь синтаксис `distinguishedName`, то есть их значениями должны быть DN.
- Параметр `refint_nothing` требуется в тех случаях, когда при попытке обеспечения целостности ссылок сервер должен удалить из записи последний содержащий ссылку атрибут, перечисленный в `refint_attributes`. Это может быть запрещено правилами схемы: например, объектный класс `groupOfNames` требует наличия хотя бы одного атрибута `member`. В этом случае сервер добавит к записи недостающий атрибут со значением, указанным в `refint_nothing`.

Проиллюстрируем работу наложения описанным выше сценарием обеспечения целостности членства в группах.

В файле `slapd.conf` укажем:

```
overlay refint
refint_attributes member
refint_nothing "cn=admin,dc=example,dc=com"
```

Данная конфигурация указывает наложению поддерживать целостность ссылок в атрибуте `member`. Этот атрибут используется в объектном классе `groupOfNames`, требующем наличия хотя бы одного члена, поэтому мы добавляем директиву `refint_nothing`, чтобы дополнить группу стандартным членом, если все её члены будут удалены.

При такой конфигурации членства в группах, наложение refint автоматически удалит `john` из группы, если его запись будет удалена из каталога:

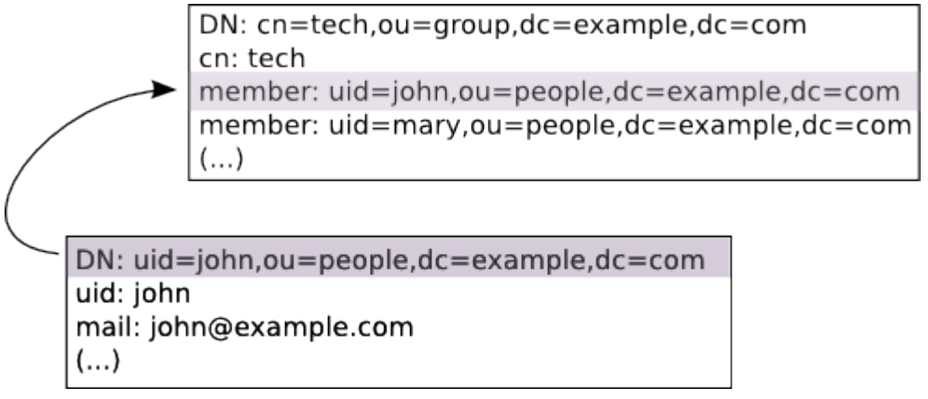

Рисунок 12.3: Поддержание целостности ссылок в группах

Обратите внимание, что если мы переименуем (`modrdn`) запись `john` в, скажем, `jsmith`, наложение refint также

переименует ссылку в атрибуте `member`, таким образом членство в группе останется верным.

При удалении всех пользователей-членов данной группы из каталога, в конечном итоге в группе останется один член: `cn=admin,dc=example,dc=com`. Он будет принудительно помещён туда из параметра `refint_nothing` для обеспечения требований корректности набора схемы.

Для базы данных должен быть задан *rootdn*, поскольку `refint` работает от имени *rootdn*, чтобы получить соответствующие права для выполнения требуемых обновлений. Задавать *rootpw* не обязательно.

12.11.3. Дополнительная информация

slapo-refint(5)

12.12. Настраиваемые коды возврата

12.12.1. Обзор

Данное наложение полезно для тестирования поведения клиентов при получении от сервера ошибочных или необычных ответов, например кодов ошибок, отсылок, превышение времени отклика и т.п.

Его можно квалифицировать как инструмент отладки в процессе разработки клиентских программ или дополнительных наложений.

Для получения детальной информации обращайтесь к man-странице *slapo-retcode(5)*.

12.12.2. Настройка кодов возврата

Наложение retcode использует набор схемы "return code", описанный в man-странице. Этот набор схемы специально разработан для использования с данным наложением и не предназначен для использования в иных случаях.

Замечание: Необходимый набор схемы загружается наложением автоматически.

Примерная конфигурация может быть такой:

```
overlay         retcode
retcode-parent  "ou=RetCodes,dc=example,dc=com"
include         ./retcode.conf

retcode-item    "cn=Unsolicited"                0x00 unsolicited="0"
retcode-item    "cn=Notice of Disconnect"       0x00 unsolicited="1.3.6.1.4.1.1466.20036"
retcode-item    "cn=Pre-disconnect"             0x34 flags="pre-disconnect"
retcode-item    "cn=Post-disconnect"            0x34 flags="post-disconnect"
```

Замечание: *retcode.conf* можно найти в исходных кодах openldap в: `tests/data/retcode.conf`

Вот отрывок из `retcode.conf`:

```
retcode-item    "cn=success"                            0x00

retcode-item    "cn=success w/ delay"                   0x00    sleeptime=2

retcode-item    "cn=operationsError"                    0x01
retcode-item    "cn=protocolError"                      0x02
retcode-item    "cn=timeLimitExceeded"                  0x03    op=search
retcode-item    "cn=sizeLimitExceeded"                  0x04    op=search
retcode-item    "cn=compareFalse"                       0x05    op=compare
retcode-item    "cn=compareTrue"                        0x06    op=compare
retcode-item    "cn=authMethodNotSupported"             0x07
retcode-item    "cn=strongAuthNotSupported"             0x07    text="same as
authMethodNotSupported"
retcode-item    "cn=strongAuthRequired"                 0x08
retcode-item    "cn=strongerAuthRequired"               0x08    text="same as strongAuthRequired"
```

Посмотреть полный `retcode.conf` можно в `tests/data/retcode.conf`.

12.12.3. Дополнительная информация

slapo-retcode(5)

12.13. Перезапись/Переназначение (Rewrite/Remap)

12.13.1. Обзор

Наложение выполняет простую перезапись DN/данных и переназначение одних объектных классов/типов атрибутов другими. В основном его используют для виртуального представления существующей информации, причём как удалённой, в сочетании с механизмом манипуляции данными ldap, описанном в *slapd-ldap(5)*, так и локальной, в сочетании с механизмом манипуляции данными relay, описанном в *slapd-relay(5)*.

Это наложение чрезвычайно настраиваемое и продвинутое, поэтому рекомендуем обратиться к man-странице *slapo-rwm(5)*.

12.13.2. Настройка наложения перезаписи/переназначения

12.13.3. Дополнительная информация

slapo-rwm(5)

12.14. Сервер-поставщик синхронизации

12.14.1. Обзор

Данное наложение реализует поддержку syncrepl-репликации, то есть репликации с помощью LDAP Content Synchronization (RFC4533 (рус.) (RFC4533 (ориг.)), синхронизация содержимого LDAP), на стороне поставщика, включая связанные с ней функции поиска.

12.14.2. Настройка наложения сервера-поставщика синхронизации

Этому наложению не требуется большого количества настроек, фактически, для многих ситуаций достаточно его просто загрузить.

Однако, оно создаёт атрибут contextCSN в корневой записи базы данных, который обновляется при каждой операции записи в базу данных. Поскольку данный атрибут во время работы сервера обновляется только в оперативной памяти, рекомендуется настроить контрольную точку для сохранения contextCSN в целевую базу данных, чтобы минимизировать время восстановления после аварийного завершения:

```
overlay syncprov
syncprov-checkpoint 100 10
```

Итак, после выполнения каждых 100 операций или по прошествии 10 минут (в зависимости от того, что произойдёт раньше), contextCSN будет сохранён в базу данных.

Всего доступны четыре директивы конфигурации: **syncprov-checkpoint**, **syncprov-sessionlog**, **syncprov-nopresent** и **syncprov-reloadhint**, все они рассмотрены в man-странице, где обсуждаются несколько разных сценариев применения этого наложения.

12.14.3. Дополнительная информация

Man-страница *slapo-syncprov(5)* и раздел Настройка различных типов репликации.

12.15. Прозрачный прокси (Translucent Proxy)

12.15.1. Обзор

Данное наложение используется совместно с базой данных одного из механизмов манипуляции данными, таких как *slapd-bdb*(5) для создания "прозрачного прокси".

Перед передачей клиенту записей, получаемых с удалённого LDAP-сервера, у них могут быть подменены некоторые или все атрибуты, или добавлены новые с помощью соответствующих записей, хранящихся в локальной базе данных.

При операции поиска сначала извлекаются записи с удалённого сервера, затем их атрибуты подменяются атрибутами из локальной базы данных (если таковые имеются). В локальных переопределениях могут применяться операции add, modify, и modrdn, если их использование разрешено root-пользователю локальной базы данных прозрачного прокси.

При операциях сравнения сначала происходит сравнение с атрибутами записи из локальной базы данных (если таковые имеются), а затем с данными из удалённой базы данных.

12.15.2. Настройка наложения прозрачного прокси

Для этого наложения существуют различные варианты настройки, но в нашем примере мы продемонстрируем добавление новых атрибутов к удалённой записи, а также поиск по этим новым добавленным локальным атрибутам. Чтобы узнать больше о переопределении удалённых записей и настройке поисковых операций, посмотрите *slapo-translucent(5)*

Замечание: Наложение прозрачного прокси отключает проверку корректности набора схемы локальной базы данных, поэтому на записи, содержащие переопределяемые атрибуты, не накладывается требование соблюдения полноты схемы данных.

Сначала настроим наложение в нормальной манере:

```
include     /usr/local/etc/openldap/schema/core.schema
include     /usr/local/etc/openldap/schema/cosine.schema
include     /usr/local/etc/openldap/schema/nis.schema
include     /usr/local/etc/openldap/schema/inetorgperson.schema

pidfile     ./slapd.pid
argsfile    ./slapd.args

database    bdb
suffix      "dc=suretecsystems,dc=com"
rootdn      "cn=trans,dc=suretecsystems,dc=com"
rootpw      secret
directory   ./openldap-data

index       objectClass eq

overlay     translucent
translucent_local carLicense

uri         ldap://192.168.X.X:389
lastmod     off
acl-bind    binddn="cn=admin,dc=suretecsystems,dc=com" credentials="blahblah"
```

Обратите внимание на директиву overlay и следующую за ней директиву, определяющую, по каким атрибутам мы хотим осуществлять поиск в локальной базе данных. Для обращения к удалённому серверу каталогов мы должны также подключить механизм манипуляции данными ldap.

Ну, а теперь возьмём для примера группу LDAP на удалённом сервере:

```
# itsupport, Groups, suretecsystems.com
dn: cn=itsupport,ou=Groups,dc=suretecsystems,dc=com
```

```
objectClass: posixGroup
objectClass: sambaGroupMapping
cn: itsupport
gidNumber: 1000
sambaSID: S-1-5-21-XXX
sambaGroupType: 2
displayName: itsupport
memberUid: ghenry
memberUid: joebloggs
```

и создадим файл LDIF, чтобы добавить в локальную базу данных запись с несколько странным набором новых атрибутов (в целях демонстрации):

```
[ghenry@suretec test_configs]$ cat test-translucent-add.ldif
dn: cn=itsupport,ou=Groups,dc=suretecsystems,dc=com
businessCategory: frontend-override
carLicense: LIVID
employeeType: special
departmentNumber: 9999999
roomNumber: 41L-535
```

Поиск по нашему прокси выдаёт следующий результат:

```
[ghenry@suretec test_configs]$ ldapsearch -x -H ldap://127.0.0.1:9001 "(cn=itsupport)"
# itsupport, Groups, OxObjects, suretecsystems.com
dn: cn=itsupport,ou=Groups,ou=OxObjects,dc=suretecsystems,dc=com
objectClass: posixGroup
objectClass: sambaGroupMapping
cn: itsupport
gidNumber: 1003
SAMBASID: S-1-5-21-XXX
SAMBAGROUPTYPE: 2
displayName: itsupport
memberUid: ghenry
memberUid: joebloggs
roomNumber: 41L-535
departmentNumber: 9999999
employeeType: special
carLicense: LIVID
businessCategory: frontend-override
```

Мы видим 5 новых атрибутов, добавленных к удалённой записи перед тем, как выдать её нашему клиенту.

Поскольку мы определили поиск по локальному атрибуту:

```
overlay     translucent
translucent_local carLicense
```

мы можем осуществлять поиск по нему в уже полностью сформированной записи:

```
ldapsearch -x -H ldap://127.0.0.1:9001 (carLicense=LIVID)
```

Это весьма интересная функция, поскольку мы не только можем локально "расширить" удалённый сервер каталогов, но и осуществлять поиск по локальным записям.

Замечание: Поскольку наложение translucent не выполняет никаких переписываний DN, записи в локальной и удалённой базах данных должны иметь одинаковый суффикс. В противном случае сервер может аварийно завершить работу с ошибками типа No Such Object (нет такого объекта) или другими.

12.15.3. Дополнительная информация

slapo-translucent(5)

12.16. Проверка уникальности атрибутов

12.16.1. Обзор

Данное наложение может использоваться поверх механизмов манипуляции данными, таких как *slapd-bdb(5)*,

для принудительного поддержания уникальности одного или нескольких атрибутов в пределах поддерева.

12.16.2. Настройка наложения проверки уникальности атрибутов

Данное наложение воздействует только на те записи, которые были созданы или изменены после того, как наложение было применено к базе данных. Чтобы проверить уникальность уже имеющихся данных, их можно экспортировать и снова импортировать с помощью LDAP-операции add, (для большого объёма данных такой способ будет неэффективным, в отличии от **slapcat**).

В следующем примере, при установке проверки уникальности для атрибута **mail**, в ходе выполнения операций **add**, **modify** или **modrdn** поддерево будет сканировано (в указанном при настройке диапазоне поиска) на наличие записей с тем же значением атрибута **mail**, что и присутствует в запросе. Если найдётся хотя бы одна такая запись, запрос будет отвергнут.

Для проверки уникальности атрибута **mail** с поиском по всему поддереву, начиная от базового DN текущей базы данных, просто добавим следующие строки конфигурации:

```
overlay unique
unique_uri ldap:///?mail?sub?
```

Замечание: Если в URI не было указано никаких атрибутов, например **ldap:///??sub?**, проверка уникальности будет распространена на все неоперационные атрибуты. Можно, однако, исключить некоторые из неоперационных атрибутов с помощью ключевого слова **ignore**.

Итак, в нашем каталоге имеется запись:

```
dn: cn=gavin,dc=suretecsystems,dc=com
objectClass: top
objectClass: inetorgperson
cn: gavin
sn: henry
mail: ghenry@suretecsystems.com
```

При добавлении такой записи:

```
dn: cn=robert,dc=suretecsystems,dc=com
objectClass: top
objectClass: inetorgperson
cn: robert
sn: jones
mail: ghenry@suretecsystems.com
```

мы получим следующую ошибку:

```
adding new entry "cn=robert,dc=example,dc=com"
ldap_add: Constraint violation (19)
        additional info: some attributes not unique
```

Чтобы осуществлять более комплексный отбор записей, в наложении можно указать несколько URI для одного домена. Также можно указать несколько директив **unique_uri** или атрибутов **olcUniqueURI** для поддержки независимых доменов.

За дополнительной информацией и подробностями использования ключевых слов **strict** и **ignore**, обращайтесь к man-странице *slapo-unique(5)*.

12.16.3. Дополнительная информация

slapo-unique(5)

12.17. Сортировка значений атрибутов

12.17.1. Обзор

Наложение сортировки значений атрибутов может использоваться поверх механизмов манипуляции данными в пределах поддерева базы данных для сортировки значений указанных атрибутов, которые могут иметь несколько значений в одной записи. Сортировка производится только когда данные атрибуты возвращаются в ответ на запрос клиента.

12.17.2. Настройка наложения сортировки значений атрибутов

Сортировка может быть определена как в порядке убывания, так и возрастания, с использованием числового или символьного метода сортировки. Кроме того, может быть определена сортировка "по весу" с использованием числового значения веса, ассоциированного со значением атрибута.

Сортировка по весу всегда производится в порядке возрастания, но она может быть скомбинирована с другими методами для значений, имеющих одинаковый вес. Вес указывается подстановкой целочисленного значения {<вес>} перед собственно значением атрибута, с которым этот вес ассоциируется. При поисковых запросах данное значение веса отбрасывается и клиенту передаётся "чистое" значение атрибута.

Приведём несколько примеров. Для сортировки по возрастанию:

```
loglevel    sync stats

database    hdb
suffix      "dc=suretecsystems,dc=com"
directory   /usr/local/var/openldap-data

......

overlay valsort
valsort-attr memberUid ou=Groups,dc=suretecsystems,dc=com alpha-ascend
```

Получаем:

```
# sharedemail, Groups, suretecsystems.com
dn: cn=sharedemail,ou=Groups,dc=suretecsystems,dc=com
objectClass: posixGroup
objectClass: top
cn: sharedemail
gidNumber: 517
memberUid: admin
memberUid: dovecot
memberUid: laura
memberUid: suretec
```

Для сортировки по весу внесём изменения в нашу запись:

```
# sharedemail, Groups, suretecsystems.com
dn: cn=sharedemail,ou=Groups,dc=suretecsystems,dc=com
objectClass: posixGroup
objectClass: top
cn: sharedemail
gidNumber: 517
memberUid: {4}admin
memberUid: {2}dovecot
memberUid: {1}laura
memberUid: {3}suretec
```

и поменяем конфигурацию:

```
overlay valsort
valsort-attr memberUid ou=Groups,dc=suretecsystems,dc=com weighted
```

В этом случае получаем:

```
# sharedemail, Groups, OxObjects, suretecsystems.com
dn: cn=sharedemail,ou=Groups,ou=OxObjects,dc=suretecsystems,dc=com
objectClass: posixGroup
objectClass: top
cn: sharedemail
gidNumber: 517
memberUid: laura
memberUid: dovecot
memberUid: suretec
memberUid: admin
```

12.17.3. Дополнительная информация

slapo-valsort(5)

12.18. Объединение наложений в стек

12.18.1. Обзор

Наложения могут быть объединены в стек. Это означает, что более одного наложения сразу может быть применено к той или иной базе данных или на глобальном уровне `frontend`. При последовательном вызове наложений исполняется каждая из функций работающего в данный момент наложения (если настроено её исполнение). Несколько наложений исполняются в обратном порядке (как и положено стеку) по отношению к их определению в slapd.conf (5), или по отношению к установленному для них порядку в конфигурационной базе данных (назначение порядка сортировки описано в slapd-config (5)).

12.18.2. Возможные сценарии применения

12.18.2.1. Samba

К содержанию

13. Спецификация схемы

В этом разделе описывается как расширить пользовательскую (т.е. не системную, жёстко скомпилированную в slapd) схему, используемую *slapd*(8). Предполагается, что читатель знаком с информационной моделью LDAP/X.500.

В первом подразделе, Файлы наборов схемы, распространяемые с дистрибутивом, даются некоторые детали наборов определения схемы, поставляемых с дистрибутивом, а также информация о том, где можно найти другие наборы определения схемы. Во втором подразделе, Расширение схемы, описывается, как определить новые элементы схемы.

В данном разделе не обсуждается расширение системной схемы, используемой *slapd*(8), поскольку это требует модификации исходного кода. Системная схема включает в себя все операционные типы атрибутов, а также объектные классы, которые могут или должны использовать данные атрибуты (прямо или косвенно).

13.1. Файлы наборов схемы, распространяемые с дистрибутивом

OpenLDAP Software распространяется с наборами спецификаций схемы для Ваших нужд. Каждый набор описан в файле, который можно подключить (с помощью директивы `include`) в Вашем файле *slapd.conf*(5). Эти файлы схемы по умолчанию установлены в директорию `/usr/local/etc/openldap/schema`.

Таблица 8.1: Спецификации схемы, распространяемые с дистрибутивом

Файл	**Описание**
`core.schema`	OpenLDAP *core* (обязательная)
`cosine.schema`	Cosine and Internet X.500 (полезная)
`inetorgperson.schema`	InetOrgPerson (полезная)
`misc.schema`	Разное (экспериментальная)
`nis.schema`	Network Information Services (FYI)
`openldap.schema`	OpenLDAP Project (экспериментальная)

Чтобы использовать любой из этих файлов схемы, Вам нужно только подключить нужный файл в разделе глобальных определений вашего файла *slapd.conf*(5). Например:

```
# Подключаем схему
include /usr/local/etc/openldap/schema/core.schema
include /usr/local/etc/openldap/schema/cosine.schema
include /usr/local/etc/openldap/schema/inetorgperson.schema
```

Доступны также дополнительные файлы, смотрите OpenLDAP FAQ (http://www.openldap.org/faq/).

Замечание: Не следует изменять какие-либо элементы схемы, определённые в распространяемых файлах схемы.

13.2. Расширение схемы

Используемая *slapd*(8) схема может быть расширена для поддержки дополнительных синтаксисов, правил соответствия, типов атрибутов и объектных классов. В этом подразделе детально описано как добавить новые типы атрибутов и объектные классы, требуемые в пользовательских приложениях, с использованием уже поддерживаемых slapd синтаксисов и правил соответствия. Для slapd также можно определить дополнительные синтаксисы, правила соответствия и наборы системной схемы, но это требует написания программного кода и в данном подразделе не рассматривается.

Пять шагов к определению нового набора схемы:

1. получить идентификатор объекта (Object Identifier)
2. выбрать префикс имени
3. создать локальный файл набора схемы
4. определить пользовательские типы атрибутов (если необходимо)
5. определить пользовательские объектные классы

13.2.1. Идентификаторы объектов

Каждый элемент схемы идентифицируется глобально-уникальным Object Identifier (OID, идентификатор объекта). OID также используются для идентификации других объектов. Обычно их можно найти в протоколах, описанных в ASN.1. В частности, они активно используются в Simple Network Management Protocol (SNMP). Поскольку OID иерархичны, Ваша организация может получить один OID и делать от него ответвления по мере необходимости. Например, если Вашей организации был присвоен OID `1.1`, Вы можете сделать такие ответвления:

Таблица 8.2: Примерная иерархия OID

OID	Assignment
`1.1`	OID организации
`1.1.1`	Элементы SNMP
`1.1.2`	Элементы LDAP
`1.1.2.1`	Типы атрибутов
`1.1.2.1.1`	x-my-Attribute (один из атрибутов)
`1.1.2.2`	Объектные классы
`1.1.2.2.1`	x-my-ObjectClass (один из объектных классов)

Вы, конечно, можете выстраивать иерархию в пределах своего OID, ориентируясь на потребности Вашей организации. Независимо от того, какую иерархию Вы избрали, Вам нужно вести реестр выделенных Вами идентификаторов и присвоения их объектам. Это может быть простой текстовый файл или что-то более продвинутое, типа *OID-реестра OpenLDAP* (http://www.openldap.org/faq/index.cgi?file=197).

Дополнительную информацию о идентификаторах объектов (и сервисе регистрации) можно посмотреть

здесь: http://www.alvestrand.no/objectid/.

Ни при каких обстоятельствах Вы не должны заниматься самозахватом ветви идентификаторов OID!

Для *бесплатного* получения зарегистрированного OID, можно запросить OID из пространства *Private Enterprise (частное предприятие)*, которое поддерживается организацией Internet Assigned Numbers Authority (ORG:IANA). Любое частное предприятие (организация) может запросить присвоить ему в пределах этого пространства Private Enterprise Number (PEN, номер частного предприятия). Просто заполните форму IANA на странице http://pen.iana.org/pen/PenApplication.page и в течение нескольких дней Вам вышлют Ваш официальный PEN. Ваш основной OID будет примерно такой: `1.3.6.1.4.1.X`, где `X` - целое число.

Замечание: Полученные с помощью заполнения данной формы PEN могут быть использованы в любых целях, в том числе и идентификации элементов схемы LDAP.

Кроме того, можно получить подпространство OID от национального органа (например, ANSI, BSI).

13.2.2. Именование элементов

В дополнение к присвоению каждому элементу схемы уникального идентификатора, Вы должны предоставить ему как минимум одно текстовое имя. Имена должны быть зарегистрированы в IANA или иметь префикс "x-" для помещения элемента в пространство имён "private use" ("для частного использования").

Имена должны быть одновременно и описательными и не пересекаться с именами других элементов схемы. В частности, любое имя, которое Вы выберите, не должно пересекаться с действующими и зарезервированными именами из Standard Track (Стандартного Ряда) (это гарантировано, если Ваши имена зарегистрированы или начинаются с "x-").

Следует отметить, что Вы можете получить свой собственный зарегистрированный префикс имен, чтобы не нужно было бы регистрировать каждое имя отдельно. За деталями обращайтесь к RFC4520.

В приведённых ниже примерах мы использовали короткий префикс '`x-my-`'. Подобные короткие префиксы могут применяться только в очень больших глобальных организациях. В общем случае мы рекомендуем что-то вроде '`x-ru-Firm-`' (Российская компания) или '`x-com-Example`' (элементы, ассоциированные с организацией `example.com`).

13.2.3. Файл локального набора схемы

Чтобы определить правила схемы для создания записей в каталоге, можно использовать директивы конфигурационного файла `objectclass` и `attributetype`. Обычной практикой является создание файла, содержащего определения Ваших пользовательских элементов схемы. Мы рекомендуем Вам создать файл `local.schema` в `/usr/local/etc/openldap/schema/local.schema`, а затем подключать его в Ваш файл *slapd.conf*(5) сразу после других директив подключения наборов схемы `include`.

```
# Подключаем наборы схемы
include /usr/local/etc/openldap/schema/core.schema
include /usr/local/etc/openldap/schema/cosine.schema
include /usr/local/etc/openldap/schema/inetorgperson.schema
# Подключаем локальный набор схемы
include /usr/local/etc/openldap/schema/local.schema
```

13.2.4. Спецификация типа атрибута

Для определения нового типа атрибута используется директива *attributetype*. Эта директива использует такое же Attribute Type Description (описание типа атрибута, определённое в RFC4512), как и атрибут attributeTypes из subschema subentry (служебное понятие для описания DIT каталога, смотрите RFC4512):

```
attributetype <RFC4512 Attribute Type Description>
```

где Attribute Type Description определяется следующей ABNF (Расширенная спецификация синтаксиса Бэкуса-Наура):

```
AttributeTypeDescription = "(" whsp
```

```
      numericoid whsp              ; идентификатор типа атрибута
    [ "NAME" qdescrs ]             ; имя используемое в типе атрибута
    [ "DESC" qdstring ]            ; описание
    [ "OBSOLETE" whsp ]
    [ "SUP" woid ]                 ; унаследован от этого AttributeType
    [ "EQUALITY" woid              ; имя правила соответствия
    [ "ORDERING" woid              ; имя правила соответствия
    [ "SUBSTR" woid ]              ; имя правила соответствия
    [ "SYNTAX" whsp noidlen whsp ] ; OID синтаксиса
    [ "SINGLE-VALUE" whsp ]        ; по умолчанию может иметь несколько значений
    [ "COLLECTIVE" whsp ]          ; по умолчанию не общий
    [ "NO-USER-MODIFICATION" whsp ]; по умолчанию изменяемый пользователем
    [ "USAGE" whsp AttributeUsage ]; по умолчанию userApplications
    whsp ")"

AttributeUsage =
    "userApplications"     /
    "directoryOperation"   /
    "distributedOperation" / ; значения, общие для DSA в распределённой системе
    "dSAOperation"           ; значения, специфичные для этого сервера DSA
```

где whsp - это пробельный символ (' '), numericoid - глобально-уникальный OID в точечно-цифровой форме (например, `1.1.0`), qdescrs - одно или более имён, woid - это или имя или OID, за которым может следовать необязательный указатель длины (например, `{10}`).

Например, типы атрибутов `name` и `cn` определены в `core.schema` так:

```
  attributeType ( 2.5.4.41 NAME 'name'
          DESC 'name(s) associated with the object'
          EQUALITY caseIgnoreMatch
          SUBSTR caseIgnoreSubstringsMatch
          SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{32768} )
  attributeType ( 2.5.4.3 NAME ( 'cn' 'commonName' )
          DESC 'common name(s) asscіated with the object'
          SUP name )
```

Обратите внимание, что в каждом из них определены OID атрибута, короткое имя и краткое описание. Каждое имя - это псевдоним для OID. В ответ на запрос *slapd*(8) возвращает первое из перечисленных имён.

Первый атрибут, `name`, может содержать значения с синтаксисом `directoryString` (в кодировке Unicode UTF-8). Синтаксис указан с помощью его OID (1.3.6.1.4.1.1466.115.121.1.15 идентифицирует синтаксис directoryString). Также указана рекомендуемая длина значения в 32768 символов. Серверы должны поддерживать хранения значений с этой длиной, однако могут поддерживать и более длинные значения. Данное поле НЕ ограничивает длины значения, поэтому оно игнорируется серверами (в том числе и slapd), которые не накладывают таких ограничений размера. Кроме того, полное значение и его подстроки (equality и substring) используют правило соответствия case ignore (игнорирование регистра символов). Ниже приведены таблицы с часто используемыми синтаксисами и правилами соответствия (*slapd*(8) поддерживает все из перечисленных и гораздо больше).

Таблица 8.3: Часто используемые синтаксисы

Название	**OID**	**Описание**
`boolean`	`1.3.6.1.4.1.1466.115.121.1.7`	логическое значение
`directoryString`	`1.3.6.1.4.1.1466.115.121.1.15`	строка Unicode (UTF-8)
`distinguishedName`	`1.3.6.1.4.1.1466.115.121.1.12`	LDAP DN
`integer`	`1.3.6.1.4.1.1466.115.121.1.27`	целое число
`numericString`	`1.3.6.1.4.1.1466.115.121.1.36`	числовая строка
`OID`	`1.3.6.1.4.1.1466.115.121.1.38`	идентификатор объекта
`octetString`	`1.3.6.1.4.1.1466.115.121.1.40`	произвольная последовательность октетов (байт)

Таблица 8.4: Часто используемые правила соответствия

Название	**Тип**	**Описание**
`booleanMatch`	equality	логическое значение
`caseIgnoreMatch`	equality	нечувствительное к регистру, нечувствительное к количеству пробельных символов значение

caseIgnoreOrderingMatch	ordering	нечувствительная к регистру, нечувствительная к количеству пробельных символов сортировка
caseIgnoreSubstringsMatch	substrings	нечувствительное к регистру, нечувствительное к количеству пробельных символов значение подстроки
caseExactMatch	equality	чувствительное к регистру, нечувствительное к количеству пробельных символов значение
caseExactOrderingMatch	ordering	чувствительная к регистру, нечувствительная к количеству пробельных символов сортировка
caseExactSubstringsMatch	substrings	чувствительное к регистру, нечувствительное к количеству пробельных символов значение подстроки
distinguishedNameMatch	equality	уникальное имя (DN)
integerMatch	equality	целое число
integerOrderingMatch	ordering	целочисленная сортировка
numericStringMatch	equality	числовое значение
numericStringOrderingMatch	ordering	числовая сортировка
numericStringSubstringsMatch	substrings	числовое значение подстроки
octetStringMatch	equality	строка байт
octetStringOrderingMatch	ordering	побайтная сортировка
octetStringSubstringsMatch	ordering	подстрока строки байт
objectIdentiferMatch	equality	идентификатор объекта

Второй атрибут, `cn`, - это подтип `name`, следовательно, он наследует синтаксис, правила соответствия и правила использования типа атрибута `name`. `commonName` - это его альтернативное имя.

Ни один из атрибутов не ограничен единственным значением. Оба предназначены для использования в пользовательских приложениях. Ни один из них не устарел и не является коллективным.

В следующих подразделах даётся несколько примеров.

13.2.4.1. x-my-UniqueName

Во многих организациях стремятся поставить в соответствие каждому пользователю одно уникальное имя. Хотя для этого можно было бы использовать атрибут `displayName` (RFC2798), в действительности этот атрибут предназначен для того, чтобы контролироваться пользователем, а не организацией. Мы можем просто скопировать определение `displayName` из набора `inetorgperson.schema` и исправить OID, имя, описание, и т.д.:

```
attributetype ( 1.1.2.1.1 NAME 'x-my-UniqueName'
        DESC 'unique name with my organization'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
        SINGLE-VALUE )
```

Однако, если мы хотим, чтобы это имя использовалось в операциях с обращением к атрибуту `name`, таких как `(name=*Jane*)`, мы можем альтернативно определить его как подтип от типа атрибута `name`, например:

```
attributetype ( 1.1.2.1.1 NAME 'x-my-UniqueName'
        DESC 'unique name with my organization'
        SUP name )
```

13.2.4.2. x-my-Photo

Во многих организациях собираются фотографии всех сотрудников. Для хранения фотографий можно определить тип атрибута `x-my-Photo`. Конечно, для этой задачи можно просто использовать тип `jpegPhoto` (RFC2798) (или его подтип), но в этом случае все фотографии должны быть в формате *JPEG File Interchange Format*. А мы можем определить универсальный тип атрибута для хранения изображений с использованием синтаксиса *octetString*:

```
attributetype ( 1.1.2.1.2 NAME 'x-my-Photo'
        DESC 'a photo (application defined format)'
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.40
        SINGLE-VALUE )
```

В этом случае в синтаксисе не определяется формат фотографии. Предполагается (возможно, ошибочно), что все приложения знают, как обрабатывать значения этого атрибута при их получении.

Если вы хотите поддерживать несколько форматов фотографий, можно определить разные типы атрибутов для каждого формата, добавив перед фотографией некоторую порцию информации о её типе, или описать значение атрибута с помощью ASN.1 и использовать опцию передачи `;binary`.

Альтернативный путь - хранить в атрибуте URI, указывающий на фотографию. Вы можете смоделировать подобный атрибут по аналогии с `labeledURI` (RFC2079) или просто унаследовать от него подтип:

```
attributetype ( 1.1.2.1.3 NAME 'x-my-PhotoURI'
        DESC 'URI and optional label referring to a photo'
        SUP labeledURI )
```

13.2.5. Спецификация объектного класса

Для определения нового объектного класса используется директива *objectclass*. Эта директива использует такое же Object Class Description (описание объектного класса, определённое в RFC4512), как и атрибут objectClasses из subschema subentry (служебное понятие для описания DIT каталога, смотрите RFC4512):

```
objectclass <RFC4512 Object Class Description>
```

где Object Class Description определяется следующей ABNF:

```
ObjectClassDescription = "(" whsp
        numericoid whsp      ; идентификатор объектного класса
        [ "NAME" qdescrs ]
        [ "DESC" qdstring ]
        [ "OBSOLETE" whsp ]
        [ "SUP" oids ]       ; родительский объектный класс
        [ ( "ABSTRACT" / "STRUCTURAL" / "AUXILIARY" ) whsp ]
                ; по умолчанию STRUCTURAL
        [ "MUST" oids ]      ; типы атрибутов
        [ "MAY" oids ]       ; типы атрибутов
        whsp ")"
```

где whsp - это пробельный символ (' '), numericoid - глобально-уникальный OID в точечно-цифровой форме (например, `1.1.0`), qdescrs - одно или более имён, oids - одно или более имён и/или OID.

13.2.5.1. x-my-PhotoObject

Определим *auxiliary (вспомогательный)* объектный класс, который позволит добавить атрибут x-my-Photo к любой существующей записи.

```
objectclass ( 1.1.2.2.1 NAME 'x-my-PhotoObject'
        DESC 'mixin x-my-Photo'
        AUXILIARY
        MAY x-my-Photo )
```

13.2.5.2. x-my-Person

Если Вашей организации нужен собственный внутренний *structural (структурный)* объектный класс для представления пользователей, Вы можете сделать подкласс одного из существующих классов для описания людей, таких как `inetOrgPerson` (RFC2798), и добавить к нему любые дополнительные атрибуты, которые Вы разработали.

```
objectclass ( 1.1.2.2.2 NAME 'x-my-Person'
        DESC 'my person'
        SUP inetOrgPerson
        MUST ( x-my-UniqueName $ givenName )
        MAY x-my-Photo )
```

Данный объектный класс наследует требуемые/разрешённые типы атрибутов от `inetOrgPerson`, но также требует наличия типов атрибутов `x-my-UniqueName` и `givenName` и разрешает наличие `x-my-Photo`.

13.2.6. Макросы OID

Чтобы облегчить управление и использование OID, *slapd*(8) поддерживает макросы *идентификаторов объектов*. Директива `objectIdentifier` используется для сопоставления макроса (имени) с OID. OID также может быть производным от ранее определенного макроса OID. Синтаксис файла *slapd.conf*(5) для задания макроса:

```
objectIdentifier <имя> { <oid> | <имя>[:<суффикс>] }
```

Следующий пример демонстрирует задание набора макросов OID и их использование в определении элементов схемы:

```
objectIdentifier myOID  1.1
objectIdentifier mySNMP myOID:1
objectIdentifier myLDAP myOID:2
objectIdentifier myAttributeType        myLDAP:1
objectIdentifier myObjectClass  myLDAP:2
attributetype ( myAttributeType:3 NAME 'x-my-PhotoURI'
        DESC 'URI and optional label referring to a photo'
        SUP labeledURI )
objectclass ( myObjectClass:1 NAME 'x-my-PhotoObject'
        DESC 'mixin x-my-Photo'
        AUXILIARY
        MAY x-my-Photo )
```

К содержанию

14. Вопросы безопасности

Программное обеспечение OpenLDAP разработано для функционирования в самых различных вычислительных средах, от тотально-контролируемых закрытых сетей до глобального Интернет. Поэтому программное обеспечение OpenLDAP поддерживает множество различных механизмов безопасности. В этом разделе описываются данные механизмы и обсуждаются вопросы безопасности использования OpenLDAP.

14.1. Сетевая безопасность

14.1.1. Выборочное обслуживание запросов

По умолчанию, *slapd*(8) будет обслуживать запросы на "любых" адресах как IPv4, так и IPv6. Чаще всего предпочтительно, чтобы *slapd* обслуживал запросы на определённых адресах/портах. Например, если указать для обслуживания только IPv4 адрес `127.0.0.1`, то все внешние запросы к серверу службы каталогов будут отвергнуты:

```
slapd -h ldap://127.0.0.1
```

Хотя сервер может быть настроен на обслуживание запросов на конкретном адресе интерфейса, это не говорит о том, что он будет разрешать доступ к серверу только из тех сетей, которые доступны через данный интерфейс. Для осуществления выборочного ограничения удалённых подключений рекомендуется использовать IP-фаервол.

Дополнительная информация в подразделе Параметры командной строки и *slapd*(8).

14.1.2. IP-фаервол

Для ограничения доступа на основе клиентских IP-адресов и/или сетевых интерфейсов, использующихся для соединения с клиентами, можно использовать возможности IP-фаервола серверной системы.

Обычно, *slapd*(8) слушает 389/tcp для сессий ldap:// и порт 636/tcp для сессий ldaps://. Его также можно сконфигурировать для прослушивания других портов.

Поскольку настройки IP-фаевола зависят от конкретного типа IP-фаервола, здесь не приводится никаких примеров. Смотрите документацию к Вашему IP-фаерволу.

14.1.3. TCP Wrappers

slapd(8) поддерживает TCP Wrappers. TCP Wrappers представляет собой основанную на правилах систему разграничения доступа для контроля доступа к серверу по TCP/IP. Например, правило *host_options*(5):

```
slapd: 10.0.0.0/255.0.0.0 127.0.0.1 : ALLOW
slapd: ALL : DENY
```

разрешает для доступа к службе каталогов только входящие соединения из частной сети `10.0.0.0` и из localhost (`127.0.0.1`).

Примечание: В правилах для *slapd*(8) обычно указываются такие IP-адреса, чтобы по ним нельзя было выполнить обратный поиск.

Имейте в виду, что для TCP wrappers требуется указывать соединения, которые будут приниматься. Поскольку в большинстве случаев, наоборот, требуется запрещать соединения, предпочтительнее использовать защиту IP-фаерволом, нежели TCP wrappers.

Дополнительная информация о правилах TCP wrapper в *hosts_access*(5).

14.2. Целостность данных и защита конфиденциальности

Для обеспечения целостности данных и защиты конфиденциальности можно использовать Transport Layer Security (TLS). OpenLDAP поддерживает согласование TLS (SSL) как через StartTLS, так и через ldaps://. Для получения дополнительной информации смотрите раздел Использование TLS. StartTLS -стандартный механизм установки соединения.

Также, в рамках обеспечения целостности данных и защиты конфиденциальности, поддерживается несколько механизмов Simple Authentication and Security Layer (SASL), таких как DIGEST-MD5 и GSSAPI. Для получения дополнительной информации смотрите раздел Использование SASL.

14.2.1. Факторы силы безопасности

Сервер использует факторы силы безопасности (Security Strength Factors, SSF) для указания относительной силы защиты. SSF равный 0, указывает на то, что защиты не требуется. SSF равный 1, указывает на то, что требуется защита целостности. SSF больший единицы (>1), примерно коррелирует с эффективной длиной ключа шифрования. Например, DES составляет 56, 3DES составляет 112, и AES - 128, 192 или 256.

В рамках сессии LDAP количество административного контроля, налагаемого с помощью SSF, ассоциируется со степенью защиты средствами TLS и SASL.

Элементы управления `security` не позволяют выполнение операции, если не соблюдается указанная степень защиты. Например:

```
security ssf=1 update_ssf=112
```

требует защиты целостности для всех операций, а для операций обновления (таких как добавление, удаление, изменение) требует шифрования, эквивалентного 3DES. Детальное описание можно посмотреть в *slapd.conf*(5).

SSF можно использовать при контроле доступа с целью установки более точного контроля. Для получения дополнительной информации смотрите раздел Контроль доступа.

14.3. Методы аутентификации

14.3.1. Простой ("simple") метод

У простого метода аутентификации LDAP три режима работы:

- предоставление доступа анонимному пользователю,
- предоставление доступа пользователю без прохождения проверки подлинности, и
- предоставление доступа пользователю с проверкой подлинности по имени пользователя и паролю.

Запрос анонимного доступа происходит, когда при выполнении операции подключения по методу "simple" не передаётся ни имени пользователя, ни пароля. Запрос на предоставление доступа пользователю без прохождения проверки подлинности происходит, когда при подключении передаётся только имя пользователя, а пароль не передаётся. Наконец, запрос на предоставление доступа пользователю с проверкой подлинности происходит, когда при подключении передаются корректные имя пользователя и пароль.

В результате анонимного подключения возникает авторизационная ассоциация *anonymous*. Механизм анонимного подключения по умолчанию включен, но его можно отключить, указав "`disallow bind_anon`" в *slapd.conf*(5).

Примечание: Отключение механизма анонимного подключения не предотвращает анонимного доступа к каталогу. Чтобы для доступа к базе всегда требовалась аутентификация, вместо приведённой выше директивы следует указать "`require authc`".

В результате подключения пользователя без прохождения проверки подлинности также возникает авторизационная ассоциация *anonymous*. Механизм подключения пользователя без прохождения проверки подлинности отключен по умолчанию, но его можно включить, указав "`allow bind_anon_cred`" в *slapd.conf*(5). Поскольку некоторые приложения LDAP ошибочно генерируют запросы на подключение без прохождения проверки подлинности вместо запросов с проверкой подлинности (то есть, не удостоверившись, ввёл ли пользователь пароль), включение данного механизма нежелательно.

В результате подключения с успешной аутентификацией по имени пользователя и паролю возникает авторизационная идентифицированная сущность пользователя, то есть определённое имя, которое будет ассоциировано с сессией. Подключение с проверкой подлинности по имени пользователя и паролю включено по умолчанию. Однако, поскольку данный механизм сам по себе не обеспечивает защиту от прослушивания (например, в случае хранения пароля в открытом виде), рекомендуется использовать его только в жёстко контролируемых системах, либо когда сессии LDAP защищаются другими средствами (например, TLS, IPsec). Если администратор использует TLS для защиты пароля, рекомендуется не допускать незащищённой аутентификации. Это можно сделать с помощью параметра `simple_bind` директивы `security`, которая предоставляет достаточно тонкий контроль над уровнем конфиденциальной защиты, требуемой при аутентификации по имени пользователя и паролю по методу *simple*. Например, с помощью директивы `security simple_bind=56` задается необходимость использования при *simple*-подключениях шифрования, эквивалентного DES или более стойкого.

Механизм подключения с проверкой подлинности по имени пользователя и паролю может быть полностью отключен при помощи директивы "`disallow bind_simple`".

Примечание: В результате подключения с проверкой подлинности, окончившейся неудачно, всегда возникает авторизационная ассоциация *anonymous*.

14.3.2. Метод SASL

Метод аутентификации LDAP SASL позволяет использовать любой аутентификационный механизм SASL. Применение данного метода обсуждается в разделе Использование SASL.

14.4. Хранилище паролей

Пароли LDAP обычно хранятся в атрибуте *userPassword*. В RFC4519 указано, что пароли не должны храниться в зашифрованной (или хэшированной) форме. Тем самым позволяется использование широкого круга механизмов аутентификации на основе паролей, таких как `DIGEST-MD5`. Такая схема хранения является также наиболее совместимой с реализациями системы аутентификации в различных клиентских программах.

Однако, на практике, чаще всего желательно всё же хранить хэш пароля. *slapd*(8) поддерживает несколько различных схем хранения на выбор системного администратора.

Примечание: Значения атрибутов пароля, независимо от используемой схемы хранения, должны защищаться также, как если бы они хранились в открытом виде. Хэшированные пароли подвержены *атакам по словарю* и *атакам методом полного перебора*.

Атрибут *userPassword* может иметь более одного значения, и все они могут храниться в различной форме. В процессе аутентификации *slapd* будет последовательно проходить по значениям, пока не найдёт такого, которое соответствует предложенному паролю, или пока значения не закончатся. Схема хранения указывается как префикс к значению, то есть пароль, захэшированный по схеме Salted SHA1 (`SSHA`) выглядит так:

```
userPassword: {SSHA}DkMTwBl+a/3DQTxCYEApdUtNXGgdUac3
```

Преимущество хэшированных паролей состоит в том, что взломщик, получающий доступ к хэшу, не имеет прямого доступа к настоящему паролю. К сожалению, они довольно просто подбираются с помощью атак по словарю или методом полного перебора, поэтому такое преимущество довольно призрачно (вот почему все современные Unix-системы используют парольный файл shadow).

Недостаток хэшированного хранения паролей состоит в том, что они не являются стандартными и могут вызвать проблемы с совместимостью. Это может привести к тому, что использование любых парольных механизмов аутентификации, сильнее Simple (или SASL/PLAIN), вообще исключается.

14.4.1. Схема хранения паролей SSHA

Это версия схемы SHA с "солью" . Она считается наиболее безопасной схемой хранения паролей, поддерживаемой *slapd*.

Эти значения представляют собой один и тот же пароль:

```
userPassword: {SSHA}DkMTwBl+a/3DQTxCYEApdUtNXGgdUac3
userPassword: {SSHA}d0Q0626PSH9VUld7yWpR0k6BlpQmtczb
```

14.4.2. Схема хранения паролей CRYPT

Эта схема использует функцию хэширования *crypt(3)* операционной системы. Как правило, в этой схеме генерируются традиционные 13-символьные хэши в стиле Unix, но на системах с `glibc2` также могут генерироваться более безопасные 34-байтные хэши MD5.

```
userPassword: {CRYPT}aUihad99hmev6
userPassword: {CRYPT}$1$czBJdDqS$TmkzUAb836oMxg/BmIwN.1
```

Преимущество схемы CRYPT состоит в том, что пароли могут быть перенесены из существующего парольного файла Unix без необходимости иметь пароли в открытом виде. Обе формы *crypt* используют "соль", поэтому они имеют определенную стойкость к атакам по словарю.

Примечание: Поскольку эта схема использует функцию хэширования *crypt(3)* операционной системы, она является специфичной для каждой операционной системы.

14.4.3. Схема хранения паролей MD5

Эта схема просто берёт MD5-хэш от пароля и хранит его в base64-закодированной форме:

```
userPassword: {MD5}Xr4ilOzQ4PCOq3aQ0qbuaQ==
```

Это не очень безопасная схема, хотя и безопаснее хранения в открытом виде. MD5 - быстрый алгоритм, и поскольку он не использует "соль", данная схема хранения уязвима для атак по словарю.

14.4.4. Схема хранения паролей SMD5

Данная схема усиливает основную схему MD5 путём добавления "соли" (то есть, случайных данных), что означает, что одному и тому же паролю в открытом виде может соответствовать множество форм представления хэша. Например, оба этих значения представляют собою один и тот же пароль:

```
userPassword: {SMD5}4QWGWZpj9GCmfuqEvm8HtZhZS6E=
userPassword: {SMD5}g2/J/7D5EO6+oPdklp5p8YtNFk4=
```

14.4.5. Схема хранения паролей SHA

Как и схема MD5, эта схема просто пропускает пароль через процесс хэширования SHA. SHA считается более безопасным, чем MD5, но отсутствие "соли" делает её уязвимой к атакам по словарю.

```
userPassword: {SHA}5en6G6MezRroT3XKqkdPOmY/BfQ=
```

14.4.6. Схема хранения паролей SASL

На самом деле это вовсе не схема хранения паролей. Она использует значение атрибута *userPassword*, чтобы делегировать проверку пароля другому процессу. Подробности смотрите ниже.

Примечание: Это не то же самое, что и использование SASL для аутентификации сессии LDAP.

14.5. Сквозная аутентификация

Начиная с OpenLDAP 2.0 у *slapd* появилась возможность делегировать проверку пароля отдельному процессу. При этом используется функция *sasl_checkpass(3)*, таким образом в процессе проверки может быть задействован любой механизм, который поддерживается Cyrus SASL для проверки пароля. Выбор механизмов очень широк, одна из возможностей - использовать *saslauthd(8)*, который настраивается на использование локальных файлов, Kerberos, сервера IMAP, другого сервера LDAP, или чего-либо еще, поддерживаемого механизмом PAM.

Для включения сквозной аутентификации сервер должен быть собран с опцией конфигурации `--enable-spasswd`.

Замечание: Это не то же самое, что и использование SASL для аутентификации сессии LDAP.

Сквозная аутентификация работает только с паролями в открытом виде, такими, которые используются в механизмах аутентификации "simple bind" и "SASL PLAIN".

Сквозная аутентификация происходит выборочно: она применяется только к тем пользователям, у которых значение атрибута *userPassword* имеет префикс схемы "{SASL}". Формат этого атрибута:

```
userPassword: {SASL}username@realm
```

username и *realm* передаются механизму аутентификации SASL и используются для идентификации аккаунта, пароль которого проверяется. Это позоляет произвольно связывать записи в OpenLDAP с аккаунтами, известными сторонним механизмам аутентификации.

Пользователям, у которых включена сквозная аутентификация, целесообразно, посредством контроля доступа, запретить менять свои пароли через LDAP.

14.5.1. Настройка slapd на использование поставщика аутентификации

Для записей, имеющих значение атрибута пароля со схемой "{SASL}", OpenLDAP полностью делегирует

процесс проверки пароля записи Cyrus SASL. Таким образом, все настройки делаются в конфигурационных файлах SASL.

Первый файл, который нужно рассмотреть, имеет запутанное название *slapd.conf* и обычно находится в директории библиотек SASL, часто расположенной в `/usr/lib/sasl2/slapd.conf`. Этот файл управляет работой SASL при их совместном использовании со *slapd*, а также применяется при использовании механизмов SASL для сквозной аутентификации. Детальную информацию можно найти на странице `options.html` в документации Cyrus SASL. Вот простой пример для сервера, который будет использовать *saslauthd* для проверки паролей:

```
mech_list: plain
pwcheck_method: saslauthd
saslauthd_path: /var/run/sasl2/mux
```

14.5.2. Настройка saslauthd

saslauthd способен использовать много разных аутентификационных сервисов. За деталями обращайтесь к *saslauthd(8)*. Распространённая практика - делегировать аутентификацию, частично или полностью, другому серверу LDAP. Вот пример файла `saslauthd.conf` для аутентификации через Microsoft Active Directory (AD):

```
ldap_servers: ldap://dc1.example.com/ ldap://dc2.example.com/

ldap_search_base: cn=Users,DC=ad,DC=example,DC=com
ldap_filter: (userPrincipalName=%u)

ldap_bind_dn: cn=saslauthd,cn=Users,DC=ad,DC=example,DC=com
ldap_password: secret
```

В этом случае *saslauthd* запускается с аутентификационным механизмом `ldap` и с указанием объединять SASL realm с именем пользователя:

```
saslauthd -a ldap -r
```

Это означает, что строка "username@realm", которой заканчивается значение атрибута *userPassword*, будет использоваться для поиска в AD на совпадение с фильтром "userPrincipalName=username@realm". После этого осуществляется проверка пароля путём попытки подключения к AD с использованием записи, найденной в процессе поиска, и пароля, предоставленного клиентом LDAP.

14.5.3. Тестирование сквозной аутентификации

Как правило, тестирование лучше проводить сверху вниз, начиная от конечного поставщика аутентификации, и спускаясь через *saslauthd* и *slapd* к клиентам LDAP.

В приведённом выше примере с AD сначала проверяется возможность подключения к AD с теми DN и паролем, которые будет использовать *saslauthd* при подключении:

```
ldapsearch -x -H ldap://dc1.example.com/ \
     -D cn=saslauthd,cn=Users,DC=ad,DC=example,DC=com \
     -w secret \
     -b '' \
     -s base
```

Затем проверяется, может ли быть найден тестируемый AD-пользователь:

```
ldapsearch -x -H ldap://dc1.example.com/ \
     -D cn=saslauthd,cn=Users,DC=ad,DC=example,DC=com \
     -w secret \
     -b cn=Users,DC=ad,DC=example,DC=com \
     "(userPrincipalName=user@ad.example.com)"
```

Затем проверяется, может ли этот пользователь подключиться к AD:

```
ldapsearch -x -H ldap://dc1.example.com/ \
     -D cn=user,cn=Users,DC=ad,DC=example,DC=com \
     -w userpassword \
     -b cn=user,cn=Users,DC=ad,DC=example,DC=com \
     -s base \
       "(objectclass=*)"
```

Если всё это работает, тогда и *saslauthd* сможет сделать то же самое:

```
testsaslauthd -u user@ad.example.com -p userpassword
testsaslauthd -u user@ad.example.com -p wrongpassword
```

Теперь поместите волшебную последовательность в нужную запись в OpenLDAP:

```
userPassword: {SASL}user@ad.example.com
```

Теперь можно создать соединение с OpenLDAP, используя DN этой записи и пароль AD-пользователя.

К содержанию

15. Использование SASL

Клиенты и серверы OpenLDAP способны производить аутентификацию с использованием системы Simple Authentication and Security Layer (SASL), описанной в RFC4422 (рус.) (RFC4422 (ориг.)). В данном разделе рассказано о том, как использовать SASL в OpenLDAP.

Существует несколько механизмов аутентификации, являющихся промышленными стандартами, которые могут быть использованы с SASL, в том числе GSSAPI для Kerberos V, DIGEST-MD5, а также PLAIN и EXTERNAL, которые используются с Transport Layer Security (TLS).

Стандартные клиентские утилиты, поставляемые с OpenLDAP, такие как *ldapsearch*(1) и *ldapmodify*(1), по умолчанию будут пытаться аутентифицировать пользователя на сервере службы каталогов LDAP, используя SASL. С помощью ряда несложных действий администраторы LDAP могут настроить сервис простой аутентификации SASL, чтобы позволить пользователям аутентифицироваться на сервере slapd от имени своей записи LDAP. С помощь нескольких дополнительных действий можно разрешить некоторым пользователям и службам использовать функцию прокси-авторизации SASL, что позволит им, после прохождения аутентификации, переключаться со своей идентификационной сущности на идентификационную сущность другого пользователя или службы.

В этом разделе предполагается, что Вы прочли руководство *Cyrus SASL для системных администраторов*, распространяемое с пакетом Cyrus SASL (в `doc/sysadmin.html`), и у Вас есть рабочая инсталляция Cyrus SASL. Вам следует протестировать Вашу инсталляцию SASL с помощью Cyrus SASL `sample_client` и `sample_server` перед тем, как Вы будете использовать её совместно с программным обеспечением OpenLDAP.

Обратите внимание, что в последующем тексте термин *пользователь* применяется для описания человека или сущности приложения, которые подключаются к серверу LDAP с помощью клиента LDAP, такого, как *ldapsearch*(1). Поэтому термин *пользователь* относится не только к человеку, использующему LDAP-клиент, но и к приложению, которое выполняет операции клиента LDAP без прямого контроля человека. Например, сущностью приложения является почтовый сервер, использующий операции LDAP для доступа к информации, содержащейся на сервере LDAP.

15.1. Вопросы безопасности при использовании SASL

SASL предлагает множество различных механизмов аутентификации. В этом подразделе кратко излагаются вопросы безопасности.

Некоторые механизмы, такие как PLAIN и LOGIN, предлагают уровень безопасности не выше, чем *простая (simple)* аутентификация LDAP. Как и *простая* аутентификация LDAP, такие механизмы не должны использоваться без принятия других адекватных мер защиты. Рекомендуется использовать эти механизмы только вместе с Transport Layer Security (TLS). Применение механизмов PLAIN и LOGIN далее в этом документе не обсуждается.

Механизм DIGEST-MD5 является обязательным механизмом аутентификации для реализации LDAPv3. Хотя DIGEST-MD5 не является механизмом строгой аутентификации по сравнению с доверенными системами аутентификации третьих сторон (такими как Kerberos или системы открытых ключей), он предлагает значительную защиту от некоторых атак. В отличие от механизма CRAM-MD5, он предотвращает chosen

plaintext attacks (атаки по избранному простому тексту). DIGEST-MD5 предпочтительнее, чем использование простых парольных механизмов. Механизм CRAM-MD5 считается устаревшим в пользу DIGEST-MD5. Применение DIGEST-MD5 обсуждается ниже.

Механизм GSSAPI использует GSS-API Kerberos V для обеспечения сервисов безопасной аутентификации. Механизм KERBEROS_V4 делает то же самое, используя Kerberos IV. Kerberos рассматривается как безопасная распределенная система аутентификации, подходящая как для малых, так и для крупных предприятий. Применение GSSAPI и KERBEROS_V4 обсуждается ниже.

Механизм EXTERNAL использует сервисы аутентификации, предоставляемые низкоуровневыми сетевыми службами, такими как Transport Layer Security (TLS). При использовании совместно с основанной на X.509 технологией открытых ключей TLS, механизм EXTERNAL предлагает строгую аутентификацию. TLS обсуждается в разделе Использование TLS.

Механизм EXTERNAL также может быть использован с транспортом `ldapi:///`, поскольку Unix-сокеты могут сообщать UID и GID клиентского процесса.

Можно выбрать и другие механизмы строгой аутентификации, в том числе OTP (one time passwords, одноразовые пароли) и SRP (secure remote passwords, безопасные удалённые пароли). Эти механизмы не обсуждаются в данном документе.

15.2. Аутентификация с помощью SASL

Чтобы простая аутентификация с помощью SASL заработала, нужно выполнить несколько шагов. На первом шаге настраивается окружение Вашего сервера slapd так, чтобы он мог взаимодействовать с клиентскими программами, используя систему безопасности, развёрнутую на Вашем сайте. Как правило, это предполагает создание сервисного ключа, открытого ключа или других форм создания зашифрованных соединений. На втором шаге настраивается отображение аутентификационных идентификационных сущностей в DN LDAP, это зависит от структуры размещения записей в Вашем каталоге. Объяснение первого шага дано в последующем подразделе на примере механизма Kerberos V4. Действия, требуемые для настройки аутентификационного механизма, используемого в Вашей системе, будут подобны приведённым в примере, однако руководство по каждому механизму, доступному с SASL, выходит за рамки данного раздела. Второй шаг описан в подразделе Отображение аутентификационных идентификационных сущностей.

15.2.1. GSSAPI

В этом подразделе описано использование механизма GSSAPI SASL и Kerberos V совместно с OpenLDAP. Предполагается, что у Вас уже развёрнут Kerberos V, Вы знакомы с работой этой системы, и что Ваши пользователи умеют ею пользоваться. Также предполагается, что Вы ознакомились с использованием механизма GSSAPI, прочитав *Configuring GSSAPI and Cyrus SASL* (поставляется с Cyrus SASL в файле `doc/gssapi`), и провели успешные эксперименты с предоставляемыми Cyrus приложениями `sample_server` и `sample_client`. Основную информацию по Kerberos можно найти на сайте http://web.mit.edu/kerberos/www/.

Для использования механизма GSSAPI совместно с *slapd*(8) сначала нужно создать сервисный ключ с принципалом (principal) для сервиса *ldap* в области (realm) для хоста, на котором запущена служба каталога. Например, если Ваш *slapd* запущен на `directory.example.com` и Ваш realm - `EXAMPLE.COM`, Вам нужно создать сервисный ключ с таким принципалом:

```
ldap/directory.example.com@EXAMPLE.COM
```

При запуске *slapd*(8) он должен иметь доступ к этому ключу. Обычно это достигается путём помещения ключа в файл keytab, `/etc/krb5.keytab`. Информацию о настройках местоположения файла keytab можно получить в документации по Kerberos и Cyrus SASL.

Чтобы использовать механизм GSSAPI для аутентификации при подключении к службе каталогов, пользователь получает "разрешение на получение разрешения" (Ticket Granting Ticket, TGT) перед тем, как запустить клиент LDAP. При использовании клиентских утилит OpenLDAP, пользователь может явно указать использование механизма GSSAPI, передав параметр командной строки `-Y GSSAPI`.

В целях аутентификации и авторизации, *slapd*(8) ассоциирует DN запроса аутентификации в форме:

```
uid=<primary[/instance]>,cn=<realm>,cn=gssapi,cn=auth
```

Продолжая наш пример, пользователь с принципалом Kerberos `kurt@EXAMPLE.COM` будет иметь ассоциированный DN:

```
uid=kurt,cn=example.com,cn=gssapi,cn=auth
```

а принципал `ursula/admin@FOREIGN.REALM` будет иметь ассоциированный DN:

```
uid=ursula/admin,cn=foreign.realm,cn=gssapi,cn=auth
```

Поскольку DN запроса аутентификации имеет правильный формат LDAP DN, он может быть непосредственно использован в ACL и атрибутах "member" записей с объектным классом `groupOfNames`. Или же, DN запроса аутентификации перед использованием может быть отображено в какое-нибудь другое DN. Подробности смотрите в подразделе Отображение аутентификационных идентификационных сущностей.

15.2.2. KERBEROS_V4

В этом подразделе описано использование механизма KERBEROS_V4 SASL совместно с OpenLDAP. Предполагается, что Вы знакомы с работой системы безопасности Kerberos IV и что у Вас уже развёрнут Kerberos IV. Ваши пользователи должны быть знакомы с политикой аутентификации, а также с тем, как получить учётные данные в кэше разрешений (ticket cache) Kerberos, и как обновить учётные данные, срок действия которых истёк.

Примечание: KERBEROS_V4 и Kerberos IV считаются устаревшими в пользу GSSAPI и Kerberos V.

Клиентские программы должны быть способны получить ключ сессии, чтобы использовать его при подключении к Вашему серверу LDAP. Это позволит серверу LDAP узнавать идентификационную сущность пользователя, а клиенту - удостовериться, что он подключается именно к тому серверу. Если используется шифрование, ключ сессии также применяется в процессе переговоров между клиентом и сервером при установке шифрованного соединения.

Сервер slapd запускает сервис под названием "*ldap*", и серверу требуется файл srvtab, содержащий сервисный ключ. Клиенты, которые осведомлены, что им нужно использовать SASL, получают разрешение (ticket) сервиса "ldap" с пользовательским "разрешением на получение разрешения" (TGT), в котором экземпляр разрешения совпадает с именем хоста сервера OpenLDAP. Например, если Ваш realm называется `EXAMPLE.COM` и сервер slapd запущен на хосте `directory.example.com`, файл `/etc/srvtab` на сервере будет содержать сервисный ключ

```
ldap.directory@EXAMPLE.COM
```

Когда клиент LDAP аутентифицирует пользователя при подключении к службе каталогов с использованием механизма KERBEROS_IV, он будет запрашивать ключ сессии для принципала, ассоциированного с этим пользователем, либо из кэша разрешений, либо получая новый от сервера Kerberos. Для этого требуется, чтобы в кэше было доступно правильное TGT. Если его там нет или срок его действия истёк, клиент может вывести следующее сообщение:

```
ldap_sasl_interactive_bind_s: Local error
```

Когда сервисное разрешение получено, оно будет передано серверу LDAP в качестве подтверждения идентификационной сущности пользователя. Сервер, с помощью библиотечных функций SASL, извлечёт из сервисного разрешения идентификационную сущность и realm и конвертирует их в *DN запроса аутентификации* в форме

```
uid=<имя пользователя>,cn=<realm>,cn=<механизм>,cn=auth
```

Таким образом, в приведённом выше примере, если имя пользователя было "adamson", DN запроса аутентификации будет:

```
uid=adamsom,cn=example.com,cn=kerberos_v4,cn=auth
```

Этот DN запроса аутентификации может быть непосредственно использован в ACL, или же, перед

использованием, может быть отображен в какой-нибудь другой. Подробности смотрите в подразделе Отображение аутентификационных идентификационных сущностей.

15.2.3. DIGEST-MD5

В этом подразделе описано использование механизма DIGEST-MD5 SASL, использующего секретные последовательности, хранящиеся либо в самом каталоге, либо в собственной базе данных Cyrus SASL. DIGEST-MD5 строит работу на "секретной последовательности" (как правило, пароле), известной и клиенту, и серверу. Сервер генерирует запрос, знает ли клиент секретную последовательность, и клиент отвечает, доказывая, что эта общая секретная последовательность ему известна. Это гораздо более безопасно, чем просто отправлять пароль по сети.

Cyrus SASL поддерживает несколько механизмов с общими секретными последовательностями. Для этого ему необходим доступ к паролю в открытом виде (в отличие от механизмов, пересылающих пароли в открытом виде по сети, где сервер может хранить захэшированную версию пароля).

Копия общих паролей, которой пользуется сервер, может храниться в собственной базе данных Cyrus SASL *sasldb*, во внешней системе, доступной через *saslauthd*, либо в самой базе данных LDAP. В любом случае, очень важно применять контроль доступа к файлам и контроль доступа к LDAP для предотвращения раскрытия паролей. Конфигурация и команды, обсуждаемые в этом подразделе, предполагают использование Cyrus SASL 2.1.

Чтобы использовать секретные последовательности, хранимые в *sasldb*, просто добавьте пользователей командой *saslpasswd2*:

```
saslpasswd2 -c <имя пользователя>
```

Управление паролями таких пользователей должно осуществляться с помощью команды *saslpasswd2*.

Чтобы использовать секретные последовательности, хранимые в каталоге LDAP, поместите пароли в открытом виде в атрибут `userPassword`. Чтобы убедиться, что пароли, устанавливаемые с помощью операции изменения пароля LDAP (LDAP Password Modify Operation) сохраняются в открытом виде, необходимо добавить в `slapd.conf` опцию:

```
password-hash   {CLEARTEXT}
```

Управление паролями, хранимыми таким способом, может осуществляться либо с помощью утилиты *ldappasswd*(1), либо просто изменением атрибута `userPassword`. Независимо от того, где хранятся пароли, требуется отображение DN запроса аутентификации в DN пользователя.

Механизм DIGEST-MD5 создаёт аутентификационные идентификаторы в форме:

```
uid=<имя пользователя>,cn=<realm>,cn=digest-md5,cn=auth
```

Если используется realm по умолчанию, то в идентификаторе имя realm опускается, и получается:

```
uid=<имя пользователя>,cn=digest-md5,cn=auth
```

Информацию о дополнительном отображении идентификационных сущностей смотрите ниже в подразделе Отображение аутентификационных идентификационных сущностей.

Если подходящее отображение настроено, пользователи могут указывать идентификаторы SASL при выполнении операций LDAP, и для проверки подлинности будут использованы пароли, хранящиеся в *sasldb* или непосредственно в каталоге. Например, пользователь, идентифицируемый записью каталога:

```
dn: cn=Andrew Findlay+uid=u000997,dc=example,dc=com
objectclass: inetOrgPerson
objectclass: person
sn: Findlay
uid: u000997
userPassword: secret
```

может выполнять команды в такой форме:

```
ldapsearch -Y DIGEST-MD5 -U u000997 ...
```

Примечание: в каждом из приведённых выше случаев не было предоставлено авторизационной идентификационной сущности (например, с помощью `-x`). Если вы не пытались использовать прокси-авторизацию SASL, то авторизационная идентификационная сущность опять же не будет определена. Сервер будет вычислять авторизационную идентификационную сущность на основании аутентификационной идентификационной сущности (как описано ниже).

15.2.4. EXTERNAL

Механизм EXTERNAL SASL позволяет использовать аутентификацию, уже выполненную протоколом низкого уровня, как правило TLS или Unix IPC.

Каждый транспортный протокол возвращает аутентификационные идентификационные сущности в своём собственном формате.

15.2.4.1. Формат аутентификационной идентификационной сущности TLS

Это DN Субъекта из клиентского сертификата. Обратите внимание, что DN в LDAP и X.509 отображаются по разному, поэтому сертификат, выданный

```
C=gb, O=The Example Organisation, CN=A Person
```

будет создавать аутентификационную идентификационную сущность:

```
cn=A Person,o=The Example Organisation,c=gb
```

Также обратите внимание, что Вы должны установить подходящее значение параметра TLSVerifyClient, чтобы сервер запрашивал использование клиентского сертификата. Если этого не сделать, сервер не будет пытаться использовать механизм EXTERNAL SASL. Подробности смотрите в подразделе Использование TLS.

15.2.4.2. Формат идентификационной сущности IPC (ldapi:///)

Он формируется из Unix UID и GID клиентского процесса:

```
gidNumber=<номер>+uidNumber=<номер>,cn=peercred,cn=external,cn=auth
```

Таким образом, идентификационная сущность клиентского процесса, запущенного от `root`, будет:

```
gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth
```

15.2.5. Отображение аутентификационных идентификационных сущностей

Механизм проверки подлинности на сервере slapd будет использовать библиотечные вызовы SASL для получения "имени пользователя" аутентифицированного пользователя, основываясь на том, какой механизм аутентификации применялся. Это имя пользователя располагается в пространстве имён механизма аутентификации, а не в нормальном пространстве имён LDAP. Как показано в предыдущих подразделах, такие имена пользователей включены в DN запроса аутентификации, который имеет следующую форму:

```
uid=<имя пользователя>,cn=<realm>,cn=<механизм>,cn=auth
```

или

```
uid=<имя пользователя>,cn=<механизм>,cn=auth
```

в зависимости от того, применяется ли в <механизме> концепция "realm". Также обратите внимание, что часть <realm> будет опущена в случае применения при аутентификации realm по умолчанию.

Чтобы определить идентификационную сущность, ассоциированную с пользователем, можно использовать утилиту *ldapwhoami*(1). Она очень полезна и при проверке правильности работы функции отображения.

В общем случае не подразумевается, что Вы будете заполнять свою базу данных LDAP записями LDAP в

приведённой выше форме. Скорее всего, в Вашем дереве каталога будут записи LDAP для каждого человека, которому нужно проходить аутентификацию при подключении к серверу LDAP, и это дерево не будет начинаться с cn=auth. Однако, если в Вашей системе поддерживается чёткое соответствие между "именем пользователя" и записью LDAP для человека, можно будет настроить сервер LDAP так, чтобы он автоматически отображал DN запроса аутентификации в *аутентификационный DN* пользователя.

Примечание: не выдвигается требований, чтобы DN запроса аутентификации или аутентификационный DN пользователя, полученный в результате отображения, ссылались на запись, содержащуюся в каталоге. Однако в этом случае становятся доступными дополнительные возможности (смотрите ниже).

Администратор LDAP должен настроить сервер slapd так, чтобы тот знал, как отображать DN запроса аутентификации в аутентификационный DN пользователя. Это делается добавлением одной или более директив `authz-regexp` в файл *slapd.conf*(5). Эта директива принимает два аргумента:

```
authz-regexp   <шаблон поиска>   <шаблон замены>
```

DN запроса аутентификации сравнивается с шаблоном поиска с помощью функций регулярных выражений *regcomp*() и *regexec*(), и если совпадение найдено, он переписывается согласно шаблону замены. Если присутствует несколько директив `authz-regexp`, используется только та из них, у которой первым найдено совпадение шаблона поиска с аутентификационной идентификационной сущностью. Строка, образованная после применения шаблона замены, должна быть либо аутентификационным DN пользователя, либо LDAP URL. Если полученная после замены строка означает DN, запись, именем которой служит этот DN, не обязательно должна содержаться на этом сервере. Если полученная после замены строка означает LDAP URL, то этот LDAP URL должен соответствовать одной и только одной записи, содержащейся на этом сервере.

Шаблон поиска может содержать любой из символов, применяемых в регулярных выражениях, которые перечислены в *regexec*(3C). Основные из этих символов - точка ".", звёздочка "*", открывающиеся и закрывающиеся круглые скобки "(" и ")". Если коротко, точка соответствует любому символу, звездочка говорит о возможности нуля или более повторов непосредственно предшествующего ей символа, а совпадения с выражениями в скобках запоминаются для использования в шаблоне замены.

В результате применения шаблона замены создаётся либо DN, либо URL, указывающий на пользователя. Любая часть DN запроса аутентификации, совпавшая с выражением в скобках в шаблоне поиска, хранится в переменной "$1". Эта переменная "$1" может быть указана в шаблоне замены, и будет заменена на совпавшую строку из DN запроса аутентификации. Если в шаблоне поиска было указано несколько выражений в скобках, то совпадения с ними доступны в переменных $2, $3, и так далее.

15.2.6. Прямое отображение

Там, где это возможно, рекомендуется в первую очередь использовать прямое отображение DN запроса аутентификации в аутентификационный DN пользователя. Помимо снижения нагрузки на сервер, связанной с выполнением операций поиска DN пользователя, это позволяет делать отображение в DN, которые указывают на записи, не содержащиеся на этом сервере.

Предположим, DN запроса аутентификации представлен в виде:

```
uid=adamson,cn=example.com,cn=gssapi,cn=auth
```

, а актуальная запись LDAP пользователя:

```
uid=adamson,ou=people,dc=example,dc=com
```

В этом случае прямое отображение будет выполняться следующей директивой `authz-regexp` в *slapd.conf*(5):

```
authz-regexp
  uid=([^,]*),cn=example.com,cn=gssapi,cn=auth
  uid=$1,ou=people,dc=example,dc=com
```

Можно указать даже более общие правила:

```
authz-regexp
  uid=([^,]*),cn=[^,]*,cn=auth
  uid=$1,ou=people,dc=example,dc=com
```

Однако будьте внимательны, настраивая слишком общие поисковые шаблоны. Это может привести к тому, что человек будет ошибочно аутентифицирован как DN, к которому у него не должно быть доступа. Лучше уж написать несколько конкретных директив, чем одну общую директиву, открывающую бреши в системе безопасности. Если в Вашей системе используется только один механизм аутентификации и один realm (или его вообще нет), то можно написать всего одну директиву `authz-regexp`, которая позволит Вам отображать аутентификационные идентификационные сущности в LDAP DN.

Не забудьте учесть случай, когда realm пропущен, также, как и случай, когда realm указан явно. Это может потребовать написания отдельных директив `authz-regexp` для каждого случая, причём директива с явным указанием realm должна следовать первой.

15.2.7. Отображения, основанные на поиске

В ряде случаев целесообразно использовать отображение в LDAP URL. Например, в некоторых каталогах объекты, описывающие людей, могут располагаться в нескольких областях дерева LDAP, к примеру, в поддеревьях `ou=accounting` и `ou=engineering`, причём RDN записей в них могут пересекаться между собой. Либо искомое отображение должно основываться не на DN, а на информации, содержащейся в самой записи пользователя. Рассмотрим случай, когда необходимо отобразить DN запроса аутентификации на пользователя со следующей записью:

```
dn: cn=Mark Adamson,ou=People,dc=Example,dc=COM
objectclass: person
cn: Mark Adamson
uid: adamson
```

Информации, содержащейся в DN запроса аутентификации, недостаточно, чтобы напрямую получить отображение в DN пользователя, поэтому здесь требуется осуществить поиск DN пользователя. В подобных ситуациях как раз нужно использовать директивы `authz-regexp`, шаблон замены которых формирует LDAP URL. Этот URL затем будет использоваться для выполнения внутреннего поиска в базе данных LDAP, чтобы найти аутентификационный DN пользователя.

Как и другие URL, LDAP URL имеет вид

```
ldap://<host>/<base>?<attrs>?<scope>?<filter>
```

Здесь содержатся все элементы, необходимые для выполнения поиска LDAP: <host> - имя хоста сервера, <base> - базовый LDAP DN для поиска, <attrs> - атрибуты LDAP, которые требуется получить, <scope> - диапазон поиска, имеет одно из трёх значений: "base", "one", и "sub", наконец, <filter> - поисковый фильтр LDAP. Поскольку поиск LDAP DN выполняется на текущем сервере, часть <host>можно оставить пустой. Поле <attrs> также игнорируется, так как нас интересует только DN. Эти два элемента были оставлены нами при описании формата URL, чтобы было понятно, какая информация располагается в каждой из частей строки URL.

Предположим, что человек из примера выше при проверке подлинности имеет фактическое имя пользователя "adamson", и что информация об этом содержится в атрибуте "uid" его записи LDAP. Тогда можно написать такую директиву `authz-regexp`:

```
authz-regexp
  uid=([^,]*),cn=example.com,cn=gssapi,cn=auth
  ldap:///ou=people,dc=example,dc=com??one?(uid=$1)
```

В результате выполнения этой директивы будет инициирован внутренний поиск в базе данных LDAP этого же самого сервера slapd. Если поиск возвратит ровно одну строку, она будет принята за DN данного пользователя. Если будет возвращено более одной строки, либо не будет возвращено ни одной строки, аутентификация окончится неудачей и подключение пользователя останется связанным с DN запроса аутентификации.

Для ускорения поиска необходимо проиндексировать атрибуты, используемые в поисковом фильтре <filter> в URL. Если этого не было сделано, этап аутентификации может занять неприемлемо долгое время, и у пользователей сложится впечатление, что сервер недоступен.

Более комплексные системы могут использовать несколько realm, каждый из которых соответствует разным поддеревьям каталога. Данная ситуация может быть обработана с помощью определений следующего вида:

```
# совпадение с realm Engineering
authz-regexp
```

```
      uid=([^,]*),cn=engineering.example.com,cn=digest-md5,cn=auth
      ldap:///dc=eng,dc=example,dc=com??one?(&(uid=$1)(objectClass=person))

   # совпадение с realm Accounting
   authz-regexp
      uid=([^,].*),cn=accounting.example.com,cn=digest-md5,cn=auth
      ldap:///dc=accounting,dc=example,dc=com??one?(&(uid=$1)(objectClass=person))

   # realm по умолчанию - customers.example.com
   authz-regexp
      uid=([^,]*),cn=digest-md5,cn=auth
      ldap:///dc=customers,dc=example,dc=com??one?(&(uid=$1)(objectClass=person))
```

Обратите внимание, что сначала обрабатываются явно заданные realm, чтобы избежать ситуации, когда имя realm становится частью UID. Также обратите внимание, что в данном примере использованы диапазоны и фильтры, чтобы ограничить совпадения только теми записями, которые действительно нужны.

Учтите также, что на внутренний поиск `authz-regexp` распространяется действие контроля доступа. В частности, аутентификационные идентификационные сущности должны иметь уровень доступа `auth`.

Более подробную информацию смотрите в *slapd.conf*(5).

15.3. Прокси-авторизация SASL

SASL предлагает функцию, известную как *прокси-авторизация*, которая позволяет пользователям, прошедшим аутентификацию, запрашивать возможность выполнения действий от имени другого пользователя. Данный шаг происходит после того, как пользователь получил аутентификационный DN, и включает в себя отправку авторизационной идентификационной сущности на сервер. Затем сервер принимает решение, следует ли позволить, чтобы данная авторизация произошла. Если принято положительное решение, данное соединение пользователя с LDAP будет переподключено от имени связующего DN, производного от авторизационной идентификационной сущности, и сессия LDAP продолжит выполняться с уровнем доступа нового авторизационного DN.

Решение запретить или разрешить осуществление авторизации зависит от правил и политики системы, в которой функционирует LDAP, и оно не может быть принято одной только SASL. Библиотека SASL возлагает принятие такого решения на сервер. Администратор LDAP устанавливает руководящие принципы о том, кто может авторизоваться под какой идентификационной сущностью, добавляя информацию об этом в записи базы данных LDAP. Данная функция авторизации отключена по умолчанию и, перед использованием, должна быть явно настроена администратором LDAP.

15.3.1. Применение прокси-авторизации

Такой тип сервиса полезен, когда одной сущности необходимо выполнять действия от имени многих других пользователей. Например, пользователям может быть предоставлена web-страница для изменения своей персональной информации в записи LDAP. Для подтверждения своей идентификационной сущности такие пользователи проходят аутентификацию на web-сервере, но CGI-скрипт на web-сервере не может пройти аутентификацию на сервере LDAP по учётным данным того же самого пользователя, чтобы затем внести изменения в его запись. Вместо этого, web-сервер аутентифицирует себя на сервере LDAP как служебная идентификационная сущность, скажем,

```
cn=WebUpdate,dc=example,dc=com
```

а затем с помощью SASL авторизуется как DN требуемого пользователя. Как только такая авторизация пройдена, CGI-скрипт вносит изменения в запись LDAP пользователя, поскольку сервер slapd сообщает своим ACL, что на том конце соединения находится именно этот пользователь. Конечно, пользователи могут напрямую подключаться к серверу LDAP и проходить аутентификацию самостоятельно, но это требует от них умения работать с клиентами LDAP, а с помощью web-страницы всё гораздо проще.

Прокси-авторизация также может применяться для ограничения доступа учётным записям, имеющим повышенные привилегии доступа к базе данных. Для такой учётной записи, возможно даже для root DN, определённого в *slapd.conf*(5), может быть создан ограниченный список людей, которые могут авторизоваться от имени этого DN. Можно разрешить вносить изменения в базу данных LDAP только этому DN, а чтобы стать таким DN, пользователю нужно сначала пройти аутентификацию под именем одного из тех, кто присутствует в

этом списке. Это позволяет лучше контролировать, кто же конкретно вносил изменения в базу данных LDAP. Если же пользователям будет разрешено напрямую аутентифицироваться от имени привилегированной учётной записи с помощью пароля, полученного из директивы `rootpw` файла *slapd.conf*(5), либо из атрибута `userPassword`, то контроль изменений становится более затруднительным.

Обратите внимание, что после успешной прокси-авторизации, оригинальный аутентификационный DN соединения LDAP перезаписывается новым DN из запроса авторизации. Если служебная программа способна пройти аутентификацию от своего собственного аутентификационного DN, а затем авторизоваться от другого DN, то, если в течение одной сессии LDAP она планирует переключиться на какие-нибудь другие идентификационные сущности, то ей потребуется проходить повторную аутентификацию от своего имени каждый раз перед прохождением авторизации от имени другого DN (либо использовать другой механизм прокси-авторизации). Сервер slapd не учитывает способности служебной программы переключаться на другие DN. Поэтому, при использовании механизмов аутентификации типа Kerberos, прокси-авторизация не потребует установки нескольких соединений с сервером Kerberos, поскольку TGT пользователя, от которого устанавливает соединение служебная программа, и ключ сессии "ldap" действительны для многократных обращений в течение нескольких часов срока действия разрешения.

15.3.2. Авторизационные идентификационные сущности SASL

Авторизационная идентификационная сущность SASL посылается серверу LDAP через переключатель `-X` для *ldapsearch*(1) и других утилит, или с помощью параметра `*authzid` при вызовах *lutil_sasl_defaults*(). Идентификационная сущность может быть в одной из двух форм:

```
u:<имя пользователя>
```

или

```
dn:<dn>
```

В первой форме <имя пользователя> берётся из того же самого пространства имён, что и аутентификационные идентификационные сущности выше. Это то самое имя пользователя, на которое происходит ссылка при использовании какого-либо конкретного механизма аутентификации. Авторизационные идентификационные сущности переводятся в формат DN в помощью тех же самых функций, которые используются и в процессе аутентификации, в результате чего получается *DN запроса авторизации* в форме

```
uid=<имя пользователя>,cn=<realm>,cn=<механизм>,cn=auth
```

Этот DN запроса авторизации затем пропускается через тот же самый процесс `authz-regexp` для преобразования в правильный авторизационный DN из базы данных. Если он не может быть преобразован вследствие неудачного завершения поиска по LDAP URL, запрос авторизации также завершается неудачей и выдаёт "inappropriate access" ("несанкционированный доступ"). В противном случае, строка DN становится правильным авторизационным DN, готовым пройти санкционирование.

Если авторизационная идентификационная сущность представлена во второй форме, с префиксом `"dn:"`, строка после префикса уже находится в форме авторизационного DN и готова пройти санкционирование.

15.3.3. Правила прокси-авторизации

Когда slapd получает авторизационный DN, начинается сам процесс санкционирования. Для разрешения авторизации администратор LDAP может поместить в записи LDAP два атрибута:

```
authzTo
authzFrom
```

Оба они могут иметь несколько значений. Атрибут `authzTo` - это правило источника, он помещается в запись, ассоциируемую с аутентификационным DN, чтобы указать, какие авторизационные DN может принять на себя этот аутентификационный DN. Второй атрибут - это правило назначения, он помещается в запись, ассоциируемую с запрашиваемым авторизационным DN, чтобы указать какие аутентификационные DN могут принимать его на себя.

Выбор, какой атрибут политики авторизации следует использовать, остаётся за системным администратором. Сначала проверяются правила источника в записи человека с аутентификационным DN, и, если там нет

правила authzTo, указывающего на разрешение запрашиваемой авторизации, то проверяются правила authzFrom в записи с авторизационным DN. Если и в данном случае нет указаний на легальность данного запроса авторизации, запрос запрещается. Поскольку по умолчанию запросы авторизации запрещены, правилами определяется только возможность разрешить такой запрос; обратных правил, запрещающих выполнение авторизации, не существует.

Значение (или значения) этих двух атрибутов указываются в той же форме, что и строка, получаемая на выходе шаблона замены в директиве authz-regexp: либо DN, либо LDAP URL. Например, если значение authzTo - это DN, то под этим DN может авторизоваться пользователь, прошедший аутентификацию. С другой стороны, если значение authzTo - это LDAP URL, данный URL используется для внутреннего поиска в базе данных LDAP, и аутентифицированный пользователь может стать ЛЮБЫМ ИЗ DN, которые будут возвращены в результате поиска. Если запись LDAP выглядит так:

```
dn: cn=WebUpdate,dc=example,dc=com
authzTo: ldap:///dc=example,dc=com??sub?(objectclass=person)
```

то любой пользователь, прошедший аутентификацию как cn=WebUpdate,dc=example,dc=com, может авторизоваться от имени любой другой записи LDAP в поддереве dc=example,dc=com, которая имеет объектный класс person.

15.3.3.1. Замечания по правилам прокси-авторизации

LDAP URL в атрибутах authzTo или authzFrom будут возвращать набор DN. Каждый возвращаемый DN будет подвергнут проверке. Если в результате поиска будет возвращаться большой набор DN, процесс авторизации может занять неприемлемо долгое время. Кроме того, необходимо проиндексировать атрибуты, по которым будет осуществляться поиск.

Для задания в authzFrom и authzTo более широких правил, разрешено использовать в значениях этих атрибутов DN, содержащие символы регулярных выражений. Это означает, что правило источника вида

```
authzTo: dn.regex:^uid=[^,]*,dc=example,dc=com$
```

разрешит пользователям, прошедшим аутентификацию, авторизоваться от имени любого DN, который совпадёт с шаблоном указанного регулярного выражения. Подобные сравнения с регулярными выражениями могут выполняться значительно быстрее, чем поиски LDAP с фильтром (uid=*).

Также имейте ввиду, что значения в правилах авторизации должны быть в одной из двух форм: LDAP URL или DN (с символами регулярных выражений или без них). Любая строка, в начале которой нет "ldap://", воспринимается как DN. Нельзя использовать в качестве правила авторизации другую авторизационную идентификационную сущность в форме "u:<username>".

15.3.3.2. Настройка политики

Решение о том, правила какого типа, authzFrom или authzTo, нужно использовать, будет зависеть от ситуации, сложившейся в системе. Например, если список людей, которые могут авторизоваться под данной идентификационной сущностью, может быть легко описан одним поисковым фильтром, то можно написать одно правило назначения. Если такой список нельзя легко задать с помощью поискового фильтра, и этот список не велик, возможно лучше написать в каждой из записей тех людей, которым будет разрешено выполнять прокси-авторизацию, правило источника.

По умолчанию, обработка правил прокси-авторизации отключена. Для включения авторизации нужно установить в файле *slapd.conf*(5) директиву authz-policy. Эта директива может быть установлена в none - не обрабатывать правила (по умолчанию), to - для обработки правил источника, from - для обработки правил назначения, или both - для обработки обоих типов правил.

Правила источника являются чрезвычайно мощными. Если у обычных пользователей есть право изменять атрибут authzTo своих собственных записей, они могут написать правила, которые позволят им авторизоваться под кем бы то ни было. Поэтому, при использовании правил источника, необходимо защищать атрибут authzTo с помощью ACL, которые будут позволять изменять значения данного атрибута только привилегированным пользователям.

К содержанию

16. Использование TLS

Клиенты и серверы OpenLDAP способны использовать технологию Transport Layer Security (TLS) для обеспечения защиты целостности и конфиденциальности данных, а также для поддержки аутентификации LDAP с использованием механизма EXTERNAL SASL. TLS определён в RFC4346.

Примечание: О том, как генерировать сертификаты, смотрите http://www.openldap.org/faq/data/cache/185.html

---

16.1. Сертификаты TLS

Для представления идентификационных сущностей клиента и сервера TLS использует сертификаты X.509. Всем серверам необходимо иметь корректные действующие сертификаты, тогда как клиентские сертификаты не являются обязательными. Клиентам необходимо иметь действующий сертификат в случае прохождения аутентификации с помощью SASL EXTERNAL. Дополнительную информацию о том, как создавать сертификаты и управлять ими, смотрите в документации по OpenSSL, GnuTLS или MozNSS, в зависимости от того, какую реализацию библиотек TLS Вы используете.

16.1.1. Сертификаты сервера

DN сертификата сервера должен использовать атрибут `CN` для именования сервера, и в этом `CN` должно содержаться полное доменное имя сервера. Дополнительные псевдонимы и шаблоны имени могут присутствовать в расширении сертификата `subjectAltName`. Подробности об именах в сертификате сервера можно найти в RFC4513.

16.1.2. Сертификаты клиента

DN сертификата клиента можно напрямую использовать в качестве аутентификационного DN. Поскольку X.509 - часть стандарта X.500, а LDAP также основан на X.500, оба они используют одинаковый формат DN; и в общем случае DN в сертификатах X.509 пользователей должны быть идентичны DN из записей LDAP. Однако, иногда данные DN могут не быть совершенно одинаковыми, в этом случае к ним может применяться функция отображения, описанная в подразделе Отображение аутентификационных идентификационных сущностей.

16.2. Настройка TLS

После получения необходимых сертификатов, и на клиенте и на сервере нужно произвести некоторое количество настроек, чтобы включить TLS и задействовать эти сертификаты. Как минимум, на клиентах должно быть настроено имя файла, содержащего все сертификаты Центров сертификации (или Удостоверяющих центров, Certificate Authority, CA), которым они будут доверять. На сервере должен быть настроен список сертификатов CA, а также его собственный сертификат сервера и закрытый ключ.

Обычно сертификат сервера и все доверенные сертификаты клиентов будут изданы одним CA, таким образом, серверу требуется доверять только этому единственному CA. Однако, клиент может соединятся с несколькими защищёнными серверами, управляемыми разными организациями и имеющими сертификаты серверов, сгенерированные многими разными CA. В таком случае, при конфигурации клиента понадобится список многих разных доверенных CA.

16.2.1. Конфигурация сервера

Директивы конфигурации slapd для настройки TLS располагаются в секции глобальных директив *slapd.conf*(5).

16.2.1.1. TLSCACertificateFile <имя файла>

Данная директива указывает файл в формате PEM, содержащий сертификаты для CA, которым slapd будет доверять. Среди них должен быть сертификат для CA, который подписал сертификат сервера. Если подписавший CA не является CA верхнего (корневого) уровня, тогда должны присутствовать сертификаты для всей цепочки CA, начиная от подписавшего CA до CA верхнего уровня. Несколько сертификатов просто добавляются в этот файл; порядок значения не имеет.

16.2.1.2. TLSCACertificatePath <путь>

Данная директива указывает путь к директории, содержащей индивидуальные сертификаты CA в отдельных файлах. Кроме того, данная директория должна специальным образом управляться с помощью утилиты OpenSSL *c_rehash*. При использовании этой функции библиотека OpenSSL будет пытаться искать файлы сертификатов, основываясь на хэше их имени и серийного номера. Утилита *c_rehash* используется для создания символических ссылок с хэшированными именами, указывающих на актуальные файлы сертификатов. Таким образом, эта опция может быть использована только с файловыми системами, поддерживающими символические ссылки. В общем случае, вместо этого проще использовать директиву `TLSCACertificateFile`.

При использовании Mozilla NSS, данная директива указывает путь к директории, содержащей сертификат NSS и файлы базы данных ключей. Для добавления сертификата CA может быть использована команда *certutil*:

```
certutil -d <путь> -A -n "имя сертификата CA" -t CT,, -a -i /path/to/cacertfile.pem
```

Эта команда добавит сертификат CA, хранящийся в файле /path/to/cacertfile.pem в формате PEM (ASCII). `-t CT,,` означает, что этот сертификат является доверенным CA для выдачи сертификатов, которые будут использоваться клиентами и серверами при работе с TLS.

16.2.1.3. TLSCertificateFile <имя файла>

Данная директива указывает файл, содержащий сертификат сервера slapd. Обычно сертификаты являются открытой информацией и не требуют специальной защиты.

При использовании Mozilla NSS, в случае применения базы данных сертификатов/ключей (задаваемой с помощью `TLSCACertificatePath`), данная директива указывает имя сертификата, который нужно использовать:

```
TLSCertificateFile Server-Cert
```

При использовании маркера (token), отличного от встроенного, сначала укажите имя маркера и двоеточие:

```
TLSCertificateFile my hardware device:Server-Cert
```

Для того, чтобы узнать имена сертификатов, используйте `certutil -L`:

```
certutil -d /path/to/certdbdir -L
```

16.2.1.4. TLSCertificateKeyFile <имя файла>

Данная директива указывает файл, содержащий закрытый ключ, соответствующий сертификату, хранящемуся в файле `TLSCertificateFile`. Закрытые ключи сами по себе являются информацией ограниченного распространения и обычно шифруются с помощью пароля в целях защиты. Однако, текущая реализация не поддерживает зашифрованных ключей, поэтому ключи должны быть незашифрованы, а сами файлы тщательно защищены.

При использовании Mozilla NSS данная директива указывает имя файла, содержащего пароль для ключа сертификата, заданного в `TLSCertificateFile`. Команда modutil может быть использована для отключения парольной защиты базы данных сертификатов/ключей. Например, если в `TLSCACertificatePath` в качестве расположения базы данных сертификатов/ключей задана /etc/openldap/certdb, используйте такую команду modutil, чтобы поменять пароль на пустую строку:

```
modutil -dbdir /etc/openldap/certdb -changepw 'NSS Certificate DB'
```

Вы должны знать предыдущий пароль, если он задавался. Проигнорируйте ПРЕДУПРЕЖДЕНИЕ о запуске браузера.
Нажмите 'Enter' на приглашение ввести новый пароль.

16.2.1.5. TLSCipherSuite <cipher-suite-spec>

Данная директива настраивает то, какие шифры будут приниматься и порядок их предпочтения. `<cipher-suite-spec>` - спецификации шифров для OpenSSL. С помощью команды

```
openssl ciphers -v ALL
```

можно посмотреть подробный список доступных спецификаций шифров.

Кроме отдельных названий шифров, можно применять спецификаторы `HIGH`, `MEDIUM`, `LOW`, `EXPORT` и `EXPORT40`, наряду с `TLSv1`, `SSLv3` и `SSLv2`.

Чтобы посмотреть список шифров в GnuTLS, выполните:

```
gnutls-cli -l
```

При использовании Mozilla NSS применяется набор спецификаций шифров OpenSSL, который переводится во внутренний формат Mozilla NSS. Простого пути посмотреть список шифров из командной строки не существует. Официальный список находится в исходном коде Mozilla NSS в файле sslinfo.c в структуре

```
static const SSLCipherSuiteInfo suiteInfo[]
```

16.2.1.6. TLSRandFile <имя файла>

Данная директива указывает файл, из которого можно получить случайную последовательность бит, когда `/dev/urandom` недоступен. Если система предоставляет `/dev/urandom`, то данная директива не нужна, в противном случае необходимо сконфигурировать источник случайных данных. Некоторые системы (например, Linux) предоставляют `/dev/urandom` по умолчанию, на других (например, Solaris) для этого требуется инсталляция патча, а третьи могут вообще его не поддерживать. В последнем случае нужно установить EGD или PRNGD и задать данную директиву для указания имени сокета EGD/PRNGD. Переменная окружения `RANDFILE` также может использоваться для указания имени этого файла. Кроме того, при отсутствии этих вариантов, может быть использован файл `.rnd` в домашней директории пользователя slapd, если таковой существует. Чтобы использовать файл `.rnd`, просто создайте его и скопируйте туда несколько сотен байт произвольных данных. Этот файл используется только для предоставления начальной последовательности для генератора псевдо-случайных чисел и ему не нужно очень много данных, чтобы работать.

При использовании GnuTLS и Mozilla NSS эта директива игнорируется.

16.2.1.7. TLSEphemeralDHParamFile <имя файла>

Данная директива указывает файл, содержащий параметры для обмена ключами по алгоритму Диффи-Хеллмана (Diffie-Hellman ephemeral key exchange). Это необходимо для того, чтобы использовать сертификат DSA на стороне сервера (то есть `TLSCertificateKeyFile` указывает на ключ DSA). В данный файл может быть включено несколько наборов параметров; все они будут обработаны. Параметры могут быть сгенерированы с помощью следующей команды:

```
openssl dhparam [-dsaparam] -out <filename> <numbits>
```

При использовании GnuTLS и Mozilla NSS эта директива игнорируется.

16.2.1.8. TLSVerifyClient { never | allow | try | demand }

Эта директива определяет, какие проверки будут выполняться над клиентскими сертификатами во входящих сессиях TLS, если вообще будут выполняться какие-либо. По умолчанию эта директива установлена в `never`, в этом случае сервер никогда не запрашивает сертификат у клиента. При установке в `allow` сервер будет запрашивать сертификат у клиента; если запрашиваемый сертификат не будет предоставлен, сессия будет нормально обработана. Если сертификат предоставлен, но сервер не может его проверить, сертификат будет

проигнорирован, и сессия будет обработана нормально, как и в случае, если сертификат не был предоставлен. При установке в `try` сертификат будет запрошен, и если он не будет предоставлен, сессия будет нормально обработана. Если сертификат предоставлен и не может быть проверен, сессия немедленно завершается. При установке в `demand` сертификат будет запрошен и действительный сертификат должен быть предоставлен, в противном случае сессия немедленно завершается.

Примечание: Сервер должен запрашивать сертификат клиента при использовании механизма аутентификации EXTERNAL SASL с сессией TLS. Таким образом, перед тем, как пытаться использовать аутентификацию SASL EXTERNAL, должны быть настроены отличные от значения по умолчанию установки `TLSVerifyClient`. Кроме того, аутентификация с помощью механизма EXTERNAL SASL будет предоставляться лишь тем клиентам, от которых получены действительные клиентские сертификаты.

16.2.2. Конфигурация клиента

Директивы конфигурации клиента в основном соответвуют директивам конфигурации сервера. Названия этих директив различаются и они указываются в *ldap.conf*(5) вместо *slapd.conf*(5), но их функциональность практически не отличается. Кроме того, хотя большинство этих параметров могут быть настроены на уровне системы в целом, все они могут быть переопределены отдельными пользователями в своих файлах *.ldaprc*.

Операция LDAP Start TLS используется в LDAP для инициации переговоров TLS. Все инструменты командной строки OpenLDAP поддерживают флаги `-Z` и `-ZZ`, указывающие, что требуется выполнить операцию Start TLS. Последний флаг указывает, что инструмент должен завершить работу, если сессия TLS не может быть установлена, в то время как первый флаг позволяет продолжить выполнение команды.

В среде LDAPv2 TLS, как правило, стартовал при использовании схемы LDAP Secure URI (`ldaps://`) вместо нормальной схемы LDAP URI (`ldap://`). Инструменты командной строки OpenLDAP позволяют использовать обе схемы с флагом `-H` и в опции `URI` файла *ldap.conf*(5).

16.2.2.1. TLS_CACERT <имя файла>

Это эквивалент серверной директивы `TLSCACertificateFile`. Как отмечалось в подразделе Настройка TLS, клиенту обычно требуется знать о большем количестве CA, нежели серверу, но в остальном применимы аналогичные соображения.

16.2.2.2. TLS_CACERTDIR <путь>

Это эквивалент серверной директивы `TLSCACertificatePath`. Указанная директория также должна управляться с помощью утилиты OpenSSL *c_rehash*. При использовании Mozilla NSS директория, указанная в параметре <путь>, может содержать базу данных сертификатов/ключей.

16.2.2.3. TLS_CERT <имя файла>

Данная директива указывает файл, содержащий сертификат клиента. Поскольку этот сертификат у каждого пользователя свой, данная директива может быть указана только в пользовательском файле *.ldaprc*.

При использовании Mozilla NSS, в случае применения базы данных сертификатов/ключей (указанной в `TLS_CACERTDIR`), данная директива определяет имя сертификата, который нужно использовать:

```
TLS_CERT Certificate for Sam Carter
```

При использовании маркера (token), отличного от встроенного, сначала укажите имя маркера и двоеточие:

```
TLS_CERT my hardware device:Certificate for Sam Carter
```

Для того, чтобы узнать имена сертификатов, используйте `certutil -L`:

```
certutil -d /path/to/certdbdir -L
```

16.2.2.4. TLS_KEY <имя файла>

Эта директива указывает файл, содержащий закрытый ключ, соответствующий сертификату, который хранится в файле `TLS_CERT`. Здесь применимы те же соображения, что и для серверной директивы `TLSCertificateKeyFile`. Данная директива также определяется отдельно для каждого пользователя.

16.2.2.5. TLS_RANDFILE <filename>

Данная директива аналогична серверной директиве `TLSRandFile`.

16.2.2.6. TLS_REQCERT { never | allow | try | demand }

Данная директива эквивалентна серверной директиве `TLSVerifyClient`. Однако, значение по умолчанию для клиентов - `demand`, и обычно нет оснований, чтобы изменять эту настройку.

К содержанию

17. Построение распределённой службы каталогов

Во многих системах достаточно запустить один или несколько *slapd*(8), содержащих полное поддерево данных. Однако часто бывает желательно, чтобы один *slapd* ссылался на другие службы каталогов, обслуживающие определённую часть дерева (эти службы также могут использовать *slapd*, а могут и не использовать).

slapd поддерживает взаимодействие с такими службами с помощью сведений о *нижестоящих* и *вышестоящих* частях дерева. Сведения о нижестоящих частях дерева хранятся в объектах `referral` (RFC3296).

17.1. Сведения о нижестоящих частях дерева

Сведения о нижестоящих частях дерева нужны для организации делегирования поддерева каталога. Они хранятся в самом каталоге в специальном объекте *referral*, который располагается в точке делегирования. Точнее объект referral выступает в качестве точки делегирования, склеивающей две службы друг с другом. Такой механизм позволяет строить иерархические службы каталогов.

Объект referral имеет структурный объектный класс `referral` и то же самое уникальное имя (Distinguished Name), что и делегируемое поддерево. Как правило, объект referral также будет иметь вспомогательный объектный класс `extensibleObject`. Это позволит записи хранить соответствующие значения относительного уникального имени (Relative Distinguished Name). Лучше всего продемонстрировать это на примере.

Если сервер `a.example.net` содержит дерево `dc=example,dc=net` и собирается делегировать поддерево `dc=subtree,dc=example,dc=net` другому серверу `b.example.net`, то на сервере `a.example.net` нужно добавить следующий именованный объект referral:

```
dn: dc=subtree,dc=example,dc=net
objectClass: referral
objectClass: extensibleObject
dc: subtree
ref: ldap://b.example.net/dc=subtree,dc=example,dc=net
```

Данный сервер использует эту информацию для генерации отсылок и продолжения поиска на нижестоящих серверах.

Для тех, кто знаком с X.500, именованный объект *referral* имеет сходство со сведениями о ссылках X.500, содержащимися в *subr* DSE.

17.2. Сведения о вышестоящих частях дерева

Сведения о вышестоящих частях дерева могут быть указаны с помощью директивы `referral`, значение которой - список URI, ссылающихся на вышестоящие службы каталогов. Если у какого-либо сервера нет непосредственного вышестоящего сервера (как у `a.example.net` из примера выше), он может быть настроен на использование службы каталогов с *глобальными сведениями*, такой, как *OpenLDAP Root Service* (http://www.openldap.org/faq/index.cgi?file=393).

```
referral        ldap://root.openldap.org/
```

Однако, поскольку `a.example.net` - *непосредственно вышестоящий* сервер для `b.example.net`, нужно сконфигурировать *b.example.net* следующим образом:

```
referral        ldap://a.example.net/
```

Сервер использует эту информацию, чтобы генерировать отсылки для операций, производимых над записями, не входящими в пространство имён, содержащееся на этом сервере или нижестоящих к нему серверах.

Для тех, кто знаком с X.500, данное использование атрибута `ref` имеет сходство со сведениями о ссылках X.500, содержащимися в *Supr* DSE.

17.3. Элемент управления ManageDsaIT

Как правило, добавление, изменение и удаление объектов referral выполняется с помощью *ldapmodify*(1) или подобных инструментов, поддерживающих элемент управления ManageDsaIT. Элемент управления ManageDsaIT сообщает серверу, что Вы будете производить манипуляции с объектом referral как с обычной записью. В этом случае сервер не будет возвращать отсылку в ответ на запросы получения или обновления объектов referral.

Не нужно указывать элемент управления ManageDsaIT при управлении обычными записями.

Параметр `-M` утилиты *ldapmodify*(1) (или другой) включает ManageDsaIT. Например:

```
ldapmodify -M -f referral.ldif -x -D "cn=Manager,dc=example,dc=net" -W
```

или с *ldapsearch*(1):

```
ldapsearch -M -b "dc=example,dc=net" -x "(objectclass=referral)" '*' ref
```

Примечание: атрибут `ref` является операционным и должен быть явно запрошен, если требуется его наличие в результатах поиска.

Примечание: использование отсылок для построения распределённой службы каталогов - чрезвычайно грубый метод, плохо поддерживаемый обычными клиентами. Если существующий каталог уже построен с использованием отсылок, применение наложения *сцепления* для сокрытия отсылок позволит значительно повысить удобство работы с таким каталогом. Лучшим подходом будет использование в конфигурациях с *нижестоящими* поддеревьями явно заданных локальных и прокси баз данных, обеспечивающих цельное представление распределённого каталога.

Примечание: как правило, операции LDAP, даже поиски по поддереву, получают доступ только к одной базе данных. Это можно изменить путём склеивания баз данных друг с другом с помощью ключевого слова **subordinate**/**olcSubordinate**. Подробности смотрите в *slapd.conf*(5) и *slapd-config*(5).

К содержанию

18. Репликация

Реплицируемая служба каталогов — фундаментальное требование для развёртывания устойчивой информационно-вычислительной системы уровня предприятия.

В OpenLDAP есть различные варианты настройки для создания реплицируемой службы каталогов. В предыдущих версиях репликация описывалась в терминах *основного (master)* сервера и некоторого количества *подчинённых (slave)* серверов. На основном сервере было разрешено выполнять операции обновления каталога, инициируемые любым клиентом, а на подчинённом было разрешено выполнять операции обновления каталога, инициируемые только единственным главным сервером. Структура репликации была жёстко определена, и конкретный сервер службы каталогов мог выполнять только одну роль: либо главного, либо подчинённого.

Поскольку сейчас OpenLDAP поддерживает разнообразные варианты топологий репликации, эти устаревшие термины заменены на *поставщика (provider)* и *потребителя (consumer)*: поставщик реплицирует обновления каталога потребителям; потребители получают реплицируемые обновления от поставщиков. В отличие от жёстко определённых отношений основной/подчинённый, роли поставщика/потребителя довольно расплывчаты: реплицируемые обновления, полученные потребителем, могут быть далее распространены от этого потребителя на другие серверы, таким образом потребитель может также выступать одновременно и в качестве поставщика. Кроме того, потребителю не обязательно быть реальным сервером LDAP; это может быть и LDAP-клиент.

В следующих подразделах описывается технология репликации и обсуждаются различные доступные варианты настройки репликации.

18.1. Технология репликации

18.1.1. Репликация на основе LDAP Sync

Механизм репликации на основе LDAP Sync, коротко syncrepl, — это механизм репликации на стороне потребителя, позволяющий LDAP-серверу-потребителю поддерживать теневую копию фрагмента DIT. Механизм syncrepl исполняется на стороне потребителя как один из потоков *slapd*(8). Он создаёт и поддерживает потребительскую реплику путём соединения с поставщиком репликаций для выполнения начальной загрузки содержимого DIT, а затем либо выполнения периодического опроса содержимого DIT поставщика, либо ожидания посылки ему обновлений по мере изменения этого содержимого.

Syncrepl использует LDAP Content Synchronization protocol (протокол синхронизации содержимого LDAP), или коротко LDAP Sync, в качестве протокола синхронизации реплик. LDAP Sync реализует динамическую репликацию с отслеживанием состояния, поддерживающую синхронизацию как на основе запроса (pull-based), так и на основе посылок (push-based), и не требующую использования хранилища истории операций. В репликации на основе запроса потребитель периодически запрашивает обновления у поставщика. В репликации на основе посылок потребитель ждёт прихода обновлений, которые посылает ему поставщик сразу же после изменения содержимого своего каталога. Поскольку протокол не требует наличия хранилища истории операций, поставщику не нужно вести каких-либо журналов полученных им обновлений (обратите внимание на то, что механизм syncrepl является расширяемым и в будущем возможна поддержка дополнительных протоколов репликации).

Syncrepl отслеживает статус реплицируемого содержимого путём поддержания и обмена синхронизационными куки. Поскольку и syncrepl-потребитель и поставщик поддерживают статус содержимого своих каталогов, потребитель может опрашивать содержимое каталога поставщика для выполнения инкрементной синхронизации, то есть запрашивая записи, необходимые для приведения реплики потребителя в соответствие с содержимым каталога поставщика. Syncrepl также предоставляет удобное управление репликами путём сохранения статуса реплики. Реплика потребителя может быть построена из резервной копии на стороне потребителя или на стороне поставщика при любом статусе синхронизации. Syncrepl может автоматически синхронизировать реплику потребителя в соответствии с текущим содержимым каталога поставщика.

Syncrepl поддерживает синхронизацию как на основе запросов, так и на основе посылок. В его базовом

режиме синхронизации refreshOnly (только обновление), поставщик использует синхронизацию на основе запросов, не требующей отслеживания серверов-потребителей и хранения истории операций. Информация, которая нужна поставщику для обработки периодических запросов на проверку содержимого каталога, находится в синхронизационных куки самих запросов. Для оптимизации синхронизации на основе запросов, syncrepl использует фазу наличия (present phase) и фазу удаления (delete phase) протокола LDAP Sync, вместо того, чтобы возвращаться к частым полным перезагрузкам содержимого каталога. Для дальнейшей оптимизации синхронизации на основе запросов поставщик может вести журнал в рамках сессии в качестве хранилища истории операций. В режиме синхронизации refreshAndPersist (обновление и непрерывность) поставщик использует синхронизацию на основе посылок. Поставщик отслеживает серверы-потребители, запросившие непрерывно-действующий поиск, и посылает им необходимые обновления по мере изменения реплицируемого содержимого каталога поставщика.

При использовании syncrepl, сервер-потребитель может создать реплику без изменения конфигурации сервера-поставщика и без его перезапуска, если сервер-потребитель имеет соответствующие привилегии доступа к реплицируемому фрагменту DIT. Сервер-потребитель также может прекратить репликацию без необходимости внесения изменений и перезапуска на стороне поставщика.

Syncrepl поддерживает частичные, разреженные и дробные репликации. Фрагмент теневого DIT определяется общими критериями поиска, состоящими из базы, диапазона, фильтра и списка атрибутов. Для идентификации в процессе соединений syncrepl, на реплицируемое содержимое каталога также требуется назначать привилегии доступа.

18.1.1.1. Протокол синхронизации содержимого LDAP

Протокол LDAP Sync позволяет клиенту поддерживать синхронизированную копию фрагмента DIT. Операции LDAP Sync определяются как набор элементов управления и других элементов протокола, расширяющих поисковые операции LDAP. В этом подразделе даётся очень краткое введение в протокол синхронизации содержимого LDAP. Полное описание в RFC4533 (рус.) (RFC4533 (ориг.)).

Протокол LDAP Sync поддерживает как опросы изменений, так и прослушивание посылок изменений путём определения двух соответствующих операций синхронизации: *refreshOnly* и *refreshAndPersist*. Опросы осуществляются с помощью операции *refreshOnly*. Потребитель опрашивает поставщика с использованием поисковых запросов LDAP с приложением к ним элементов управления LDAP Sync. Копия потребителя будет синхронизирована с копией поставщика на момент выполнения опроса с использованием информации, возвращённой в результате поиска. Когда поставщик заканчивает операцию поиска, он возвращает *SearchResultDone* в конце результатов поиска, как и при выполнении нормального поиска. Прослушивание осуществляется с помощью операции *refreshAndPersist*. Как следует из названия, оно начинается с поиска, как и refreshOnly. Вместо того, чтобы закончить поиск после возврата всех записей, соответствующих в настоящий момент критериям поиска, синхронизационный поиск продолжает непрерывно выполняться на стороне поставщика. Последующие обновления синхронизируемого содержимого каталога на стороне поставщика вызывают дополнительные посылки потребителю обновлённых записей.

Операция *refreshOnly* и стадия refresh операции *refreshAndPersist* могут быть представлены с помощью фазы наличия и фазы удаления.

В фазе наличия поставщик посылает потребителю записи, обновлённые после последней синхронизации, попадающие в поисковый диапазон. Поставщик отправляет все запрошенные атрибуты обновлённой записи, независимо от того, были ли они изменены, или нет. Для каждой неизменённой записи, попавшей в поисковый диапазон, поставщик отправляет сообщение наличия, состоящее только из имени записи и элемент управления синхронизации, определяющий состояние наличия. Сообщение наличия не содержит каких-либо атрибутов записи. После получения всех обновлённых записей, и записей, находящихся в наличии, потребитель может соответствующим образом определить новую копию своего каталога путём добавления записей, добавленных на стороне поставщика, заменой записей, изменённых на стороне поставщика, и удалением записей, которые не были обновлены и не были указаны, как находящиеся в наличии, на стороне поставщика.

В фазе удаления передача обновлённых записей происходит также, как и в фазе наличия. Поставщик посылает все запрашиваемые атрибуты записей из заданного поискового диапазона, которые были обновлены с момента последней синхронизации с потребителем. Однако, в фазе удаления поставщик, вместо того, чтобы посылать сообщения наличия, посылает сообщение об удалении для каждой записи из заданного поискового диапазона, которая была удалена. Сообщение об удалении состоит только из имени записи и элемента управления синхронизации, определяющего состояние удаления. Новая копия каталога потребителя будет

определена путём добавления, модификации и удаления записей в соответствии с элементами управления синхронизации, вложенными в сообщение *SearchResultEntry*.

Поставщик LDAP Sync может использовать фазу удаления в случае, если он ведёт хранилище истории операций, и может определить, какие записи стали отличными от копии потребителя со времени последней синхронизации. Если поставщик не ведёт никакого хранилища истории операций и не может с его помощью определить ставшие различными записи, либо хранилище не содержит достаточного количества операций с соответствующими сроками давности, чтобы можно было восстановить копию потребителя с момента последней синхронизации, поставщик должен использовать фазу наличия. Тем не менее, использование фазы наличия гораздо более эффективно с точки зрения синхронизационного трафика, чем перезаливка всего содержимого каталога. Для дальнейшего повышения эффективности протокол LDAP Sync также предусматривает некоторое количество оптимизаций, таких как передача нормализованных `entryUUID` и передача нескольких `entryUUIDs` в одном сообщении *syncIdSet*.

В концовке ответа синхронизации *refreshOnly* поставщик отправляет потребителю синхронизационное куки, как индикатор состояния, что копия потребителя после синхронизации полна. Потребитель, при отправке поставщику следующего запроса на инкрементную синхронизацию, будет предоставлять данное полученное куки.

При использовании синхронизации *refreshAndPersist* поставщик отправляет синхронизационное куки в конце стадии refresh путём посылки сообщения Sync Info с параметром refreshDone=TRUE. Он также отправляет синхронизационное куки путём присоединения его к сообщениям *SearchResultEntry*, генерируемым в стадии persist синхронизационного поиска. В течении стадии persist поставщик также может отправлять сообщения Sync Info, содержащие синхронизационные куки, в любой момент, когда поставщик захочет обновить индикатор состояния на стороне потребителя.

В протоколе LDAP Sync записи уникально идентифицируются значением атрибута `entryUUID`. Оно может выступать в качестве надежного идентификатора записи, в отличие от DN записи, которое может со временем измениться. `entryUUID` присоединяется к каждому сообщению *SearchResultEntry* или *SearchResultReference* как часть элементов управления синхронизацией.

18.1.1.2. Детали Syncrepl

Механизм syncrepl использует и *refreshOnly* и *refreshAndPersist* операции протокола LDAP Sync. Если спецификация syncrepl включена в раздел настроек какой-либо из баз данных (на стороне потребителя), *slapd*(8) запускает механизм syncrepl отдельным потоком *slapd*(8) и задаёт время его исполнения. Если при настройке указана операция *refreshOnly*, механизм syncrepl будет перезадавать время запуска на определённый интервал времени всякий раз по окончании операции синхронизации. Если при настройке была указана операция *refreshAndPersist*, поток *slapd*(8) механизма syncrepl будет оставаться активным и обрабатывать сообщения постоянной синхронизации от поставщика.

В механизме syncrepl может использоваться как фаза наличия, так и фаза удаления refresh-стадии синхронизации. На стороне поставщика возможно настроить журнал сессий, который будет хранить конечное число `entryUUID` удалённых из базы данных записей. В случае ведения нескольких реплик все они будут пользоваться одним и тем же журналом сессий. Механизм syncrepl использует фазу удаления, если есть журнал сессий и время последней синхронизации сервера-потребителя не раньше времени создания самой старой записи в журнале, оставшейся после его усечения до заданного числа записей. Механизм syncrepl использует фазу наличия, если для реплицируемого содержимого каталога не был сконфигурирован журнал сессий или если время последней синхронизации реплики потребителя не перекрывается временем создания записей журнала сессий. Текущая реализация хранилища журнала сессий размещает его записи в оперативной памяти, поэтому при перезапуске поставщика информация в журнале не сохраняется. Также в настоящий момент не поддерживается доступ к хранилищу журнала сессий посредством LDAP-операций и на него невозможно установить контроль доступа.

В качестве дальнейшей оптимизации, даже в случае, когда синхронизационный поиск не ассоциирован ни с каким журналом сессий, потребителю не будет передано никаких записей, если содержимое каталога поставщика не изменилось.

Механизм syncrepl, как механизм репликации на стороне потребителя, может работать с любым механизмом манипуляции данными. На стороне поставщика LDAP Sync может быть настроен как наложение на любой механизм манипуляции данными, но лучше всего оно работает поверх механизмов манипуляции данными *back-*

bdb или *back-hdb*.

Поставщик LDAP Sync устанавливает для каждой базы данных `contextCSN` как индикатор текущего состояния синхронизации содержимого каталога поставщика. Он равен самому большому `entryCSN` в ветке поставщика, такому, что нет ни одной записи с меньшим значением `entryCSN`, для которых не подтверждены завершения транзакций операций обновления. `contextCSN` не может быть просто приравнен самому большому из установленных `entryCSN`, поскольку `entryCSN` извлекаются до начала транзакции операции обновления и не существует определённого порядка посылки подтверждений окончания транзакций.

Поставщик хранит `contextCSN` ветке каталога в атрибуте `contextCSN` корневой записи этой ветки. Данный атрибут не записывается в базу данных сразу после каждой операции обновления, а сначала сохраняется в оперативной памяти. Во время запуска базы данных поставщик считывает последний сохранённый `contextCSN` в оперативную память и после этого проводит все операции с данной копией в памяти. По умолчанию, изменения `contextCSN` в результате обновлений базы данных не записываются в базу до тех пор, пока сервер не будет корректно остановлен. Существует возможность установки контрольных точек, чтобы, при необходимости, чаще записывать `contextCSN` в базу.

Обратите внимание, что если сервер-поставщик не может прочитать `contextCSN` из корневой записи во время загрузки, он будет сканировать всю базу данных, чтобы определить это значение, и в больших базах данных это сканирование может занять много времени. Даже если значение `contextCSN` прочитано из соответствующего атрибута, всё равно происходит сканирование базы данных на предмет нахождения значений `entryCSN`, больших указанного `contextCSN`, чтобы убедиться, что данное значение `contextCSN` действительно соответствует самому большому `entryCSN` с подтверждённой транзакцией обновления в базе данных. В базах данных с поддержкой эквивалентной индексации, установка индекса на атрибут `entryCSN` и настройка контрольных точек *contextCSN* значительно ускорит данный этап сканирования.

Если невозможно считать `contextCSN` и определить его путём сканирования базы данных, генерируется новое значение. Также, если при сканировании обнаружено значение `entryCSN`, большее, чем ранее записанное в атрибуте `contextCSN` корневой записи, найденное значение будет немедленно помещено в указанный атрибут.

Потребитель также хранит состояние своей реплики, которое является значением `contextCSN` поставщика, полученным в качестве синхронизационного куки, в атрибуте `contextCSN` корневой записи. Состояние реплики, установленное на сервере потребителя, используется в качестве индикатора состояния синхронизации, когда сервер-потребитель запрашивает инкрементную синхронизацию у сервера-поставщика. Оно также используется в качестве синхронизации на стороне поставщика, если данный сервер-потребитель выполняет роль вторичного поставщика в каскадной схеме репликации. Поскольку состояние информации поставщика и потребителя располагается в одних и тех же местах в рамках их соответствующих баз данных, любой потребитель может быть переквалифицирован в поставщика (и наоборот), без выполнения каких-либо специальных действий.

Поскольку при спецификации syncrepl могут использоваться обычные поисковые фильтры, некоторые записи в ветке могут не попасть в синхронизируемое содержимое каталога. Механизм syncrepl создаёт связующие записи для заполнения промежутков в содержимом реплики, если какая-либо часть содержимого реплики является подветкой от этих пропущенных записей. Связующие записи не будут возвращаться в результате поискового запроса, если при запросе не указан элемент управления *ManageDsaIT*.

Кроме того, в следствии применения поисковых фильтров при спецификации syncrepl, может возникнуть ситуация, когда в результате внесения изменений запись будет перемещена из одного места в другое и уже не будет попадать в диапазон реплицирования, хотя фактически запись на сервере-поставщике не была удалена. Логично было бы удалить эту запись на сервере-потребителе, но в режиме *refreshOnly* поставщик не сможет определить и передать данное изменение без использования журнала сессий.

Информация о настройке в разделе Syncrepl.

18.2. Варианты развёртывания

Если спецификация LDAP Sync определяет только узкий диапазон возможностей репликации, то реализация OpenLDAP чрезвычайно гибкая и поддерживает различные режимы работы для реализации сценариев репликации, в том числе такие, которые прямо не рассматриваются в спецификации.

18.2.1. Дельта-syncrepl репликация

- Недостатки репликации LDAP Sync:

Механизм репликации LDAP Sync основан на объектах. Когда на сервере-поставщике меняется значение любого атрибута реплицируемого объекта, каждый потребитель во время репликации извлекает и обрабатывает изменившийся объект целиком, включая **как изменившиеся, так и неизменившиеся значения атрибутов**. Одним из преимуществ такого подхода является то, что если у одного объекта происходит несколько изменений, не требуется сохранения точной последовательности этих изменений, значимым является только конечное состояние записи. Но такой подход может иметь недостатки, если требуется произвести по одному изменению у нескольких объектов.

Предположим, у Вас есть база данных, состоящая из 102 400 объектов по 1 KB каждый. Далее, предположим Вы периодически запускаете на выполнение задание, которое меняет значение размером в 2 байта одного-единственного атрибута у каждого из 102 400 объектов на главном сервере. Не считая заголовков протоколов LDAP и TCP/IP, каждый раз при запуске этого задания каждый из потребителей должен получить и обработать **100 MB** данных для внесения **200 KB** изменений!

99.98% данных, которые будут переданы и обработаны в подобных случаях, являются избыточными, поскольку представляют собой неизменившиеся значения. Подобная пустая трата ценных ресурсов передачи и обработки может, к тому же, привести к неприемлемым ситуациям, когда процесс репликации действительно изменившихся значений будет идти с большим отставанием. Конечно, приведённая выше ситуация является экстремальной, однако она служит для демонстрации абсолютно реальной проблемы, которая встречается в реальных развертываниях службы каталогов LDAP.

- Когда применима дельта-syncrepl:

Дельта-syncrepl, вариант syncrepl основанный на журнале изменений, разработан для разрешения ситуаций наподобие описанной выше. Дельта-syncrepl работает путём ведения журнала изменений настраиваемой глубины в отдельной базе данных на сервере поставщика. Потребители проверяют этот журнал изменений на наличие изменений, которые им необходимы, и, в случае, если журнал содержит требуемые изменения, потребитель извлекает эти изменения из журнала изменений и применяет их к своей базе данных. Однако, если реплика слишком долго не синхронизировалась (или она полностью пуста), то для приведения её в соответствие используется стандартный syncrepl, а затем происходит обратное переключение в режим дельта-syncrepl.

Примечание: поскольку состояние базы данных на сервере поставщика хранится и в базе данных журнала изменений, и в основной базе данных, важно производить резервирование обеих этих баз данных (с помощью slapcat), а при восстановлении или копировании на другую машину также восстанавливать обе эти базы данных (с помощью slapadd).

Информация о настройках в подразделе Дельта-syncrepl.

18.2.2. Разнонаправленная репликация с несколькими главными серверами (Multi-Master)

Репликация с несколькими главными серверами (Multi-Master) — это техника репликации с использованием Syncrepl для реплицирования данных на несколько серверов-поставщиков ("Master") службы каталогов.

18.2.2.1. Преимущества репликации с несколькими главными серверами

- При отказе любого поставщика, другой поставщик продолжит принимать обновления.
- Избежание ситуации, когда отказ одной точки приводит к нарушению работы всей системы.
- Серверы-поставщики могут располагаться в нескольких разных участках сети, в том числе глобально-разнесённых.
- Хорошо подходит для систем с требованиями автоматической отказоустойчивости и высокой доступности.

18.2.2.2. Развенчание некоторых мифов об использовании репликации с несколькими главными

серверами

(Они часто позиционируются в качестве преимуществ репликации с несколькими главными серверами, хотя в действительности таковыми не являются):

- Данный подход не делает **НИЧЕГО** для балансировки нагрузки на серверы.
- Каждый поставщик **обязан** распространять изменения на **все** остальные серверы. Это означает, что ситуация с сетевым трафиком и нагрузкой по обработке записей на всех серверах сохраняется такой же, как и в случае с единственным главным сервером.
- В лучшем случае, загрузка и производительность серверов при репликации с одним и несколькими главными серверами идентичны; а если брать не самый лучший случай, вариант с одним главным сервером предпочтительнее, поскольку можно по-разному настроить индексирование для оптимизации различных схем взаимодействия между поставщиком и потребителями.

18.2.2.3. Отрицательные моменты репликации с несколькими главными серверами

- Не гарантируется целостность данных модели данных каталога.
- http://www.openldap.org/faq/data/cache/1240.html.
- Если связь с поставщиком потеряна из-за разъединения участков сети, тогда попытка "автоматического переключения" на другой сервер может только усугубить проблемы.
- Как правило, одна машина, при потере соединения с другой, не может различить, произошло ли это потому, что другая машина перестала работать, или потому, что нарушено сетевое соединение.
- Если произошло разъединение сети на участки и несколько разных клиентов стали изменять содержимое разных серверов-поставщиков, то последующее приведение серверов-поставщиков в соответствие друг другу будет весьма болезненным; в таком случае, возможно, будет лучше просто запретить внесение изменений тем клиентам, которые не имеют доступа к какому-либо одному конкретному серверу-поставщику.

Информация о настройках в подразделе Разнонаправленная репликация с несколькими главными серверами ниже.

18.2.3. Репликация в режиме зеркала (MirrorMode)

Режим зеркала — это гибридная конфигурация, предоставляющая все гарантии целостности данных, как при репликации с одним главным сервером, одновременно предоставляя высокую доступность, как при репликации с несколькими главными серверами. В режиме зеркала два поставщика настраиваются на репликацию друг от друга (как в случае с несколькими главными серверами), однако используется некий внешний интерфейс, который направляет все запросы на обновление только на один из двух серверов. В случае отказа первого поставщика, указанный выше интерфейс переключит все запросы на обновление на второго поставщика, и он, в свою очередь будет принимать и обрабатывать их. При восстановлении и перезапуске отказавшего поставщика он автоматически запросит все изменения с функционирующего поставщика и пересинхронизируется.

18.2.3.1. Положительные моменты репликации в режиме зеркала

- Решение с высокой доступностью (high-availability, HA) как для операций обновления каталога, так и для последующих получений синхронизаций репликами потребителей.
- До тех пор, пока хотя бы один поставщик работоспособен, можно безопасно принимать операции обновления.
- Серверы-поставщики реплицируются друг от друга, таким образом они постоянно находятся в актуальном состоянии и постоянно готовы заменить друг друга (горячая замена).
- Syncrepl также позволяет северам-поставщикам пересихронизироваться после простоя любой продолжительности.

18.2.3.2. Отрицательные моменты репликации в режиме зеркала

- Репликация в режиме зеркала — это не то, что принято называть решением с несколькими главными серверами, поскольку операции обновления в определённый момент времени принимаются только одним из зеркальных серверов.

- Режим зеркала можно обозначить как два активных сервера с горячей заменой друг друга (Active-Active Hot-Standby), поэтому для принятия решения, какой из серверов-поставщиков сейчас является активным, требуется внешний сервер (slapd в режиме прокси) или устройство (аппаратный балансировщик нагрузки).
- Несколько иное управление резервным копированием:
  - Если создаётся резервная копия самой базы данных Berkeley и периодически создаются резервные копии файлов журнала транзакций, то, пока не будет сделана следующая резервная копия базы данных, требуется снимать копии с файлов журнала одного и того же сервера из пары зеркальных серверов.

Информация о настройках в подразделе Режим зеркала ниже.

18.2.4. Режим Syncrepl-прокси

Хотя протокол LDAP Sync поддерживает репликацию и на основе запросов и на основе посылок, режим репликации на основе посылок (refreshAndPersist) должен всё-таки быть инициирован со стороны потребителя, чтобы поставщик мог начать посылку изменений. В некоторых конфигурациях сети, в частности там, где фаервол ограничивает направления, в которых могут устанавливаться соединения, может потребоваться режим посылок, инициированных поставщиком.

Этот режим может быть сконфигурирован с помощью механизма манипуляции данными LDAP (смотрите раздел Механизмы манипуляции данными и *slapd-ldap(8)*). Вместо того, чтобы запускать механизм syncrepl на действительном сервере-потребителе, около сервера-поставщика (или прямо на нём) настраивается slapd-ldap прокси, указывающий на сервер-потребитель, и на нём запускается механизм syncrepl.

Информация о настройках в подразделе Syncrepl-прокси.

18.2.4.1. Замена Slurpd

Старый механизм *slurpd* работал только в режиме посылок, инициированных поставщиком. Репликация Slurpd устарела в пользу репликации Syncrepl, и была полностью удалена из OpenLDAP 2.4.

Демон slurpd был оригинальным механизмом репликации, унаследованным от UMich LDAP, работавшим в режиме посылок: основной сервер рассылал изменения на подчинённые серверы. Он был заменён по многим причинам, если коротко, то:

- Он не был надёжен:
  - Он был чрезвычайно зависим от порядка записей в журнале репликаций.
  - С ним было просто попасть в состояние рассинхронизации, после чего, для пересинхронизации базы данных подчинённого сервера с основным сервером, требовалось ручное вмешательство.
  - Он не проявлял особой терпимости к недоступным серверам. Если подчинённый сервер был недоступен долгое время, журнал репликаций мог разрастись до размера, который slurpd уже не был способен обработать.
- Он работал только в режиме посылок.
- Он требовал остановки и перезапуска основного сервера при добавлении новых подчинённых серверов.
- Он поддерживал только репликацию с одним главным сервером.

У Syncrepl нет ни одного из этих недостатков:

- Syncrepl является самосинхронизирующимся; Вы можете запустить сервер-потребитель с базой данных в любом состоянии, от полностью пустой до полностью синхронизированной, и он будет автоматически делать то, что нужно для достижения и поддержания синхронизации.
  - Он полностью независим от последовательности внесения изменений.
  - Он гарантирует соответствие между потребителем и поставщиком без ручного вмешательства.
  - Он способен пересинхронизироваться независимо от длительности простоя потребителя или потери связи с поставщиком.
- Syncrepl может работать в обоих направлениях.
- Потребители могут быть добавлены в любое время, без внесения изменений на поставщике.
- Поддерживается репликация с несколькими главными серверами.

18.3. Настройка различных типов репликации

18.3.1. Syncrepl

18.3.1.1. Настройка Syncrepl

Поскольку syncrepl — это механизм репликации на стороне потребителя, спецификация syncrepl определяется в файле *slapd.conf*(5) сервера-потребителя, а не в конфигурационном файле сервера-поставщика. Первоначальная загрузка содержимого реплики может быть выполнена как запуском механизма syncrepl без синхронизационного куки, так и наполнением реплики потребителя путём загрузки LDIF-файла, который был сгенерирован как резервная копия на поставщике.

При загрузке данных из резервной копии не требуется выполнения первоначальной загрузки из самой свежей (актуальной) резервной копии содержимого каталога поставщика. Механизм syncrepl автоматически синхронизирует первоначальную реплику потребителя с текущим содержимым каталога поставщика. Поэтому не требуется остановки сервера-поставщика для того, чтобы избежать неполноты реплики, которая может быть вызвана внесением в каталог поставщика изменений во время создания и последующей загрузки резервной копии.

При реплицировании каталога большого масштаба, особенно в условиях ограниченности пропускной способности, рекомендуется загрузить реплику потребителя из резервной копии, вместо того, чтобы выполнять полную первоначальную загрузку средствами syncrepl.

18.3.1.2. Настройка slapd поставщика

Часть протокола, выполняющаяся на стороне поставщика, реализована в виде наложения. Само наложение, перед тем, как его можно будет использовать, должно быть сконфигурировано в *slapd.conf*(5). На поставщике могут быть заданы только две директивы конфигурации, для установки контрольных точек `contextCSN` и для настройки журнала сессий. Поскольку поиск LDAP Sync — это сущность, над которой можно установить контроль доступа, для реплицируемого содержимого каталога должны быть настроены корректные привилегии контроля доступа.

Контрольная точка `contextCSN` конфигурируется следующей директивой:

```
syncprov-checkpoint <количество операций> <количество минут>
```

Контрольные точки проверяются только после успешной операции записи. Если с момента последней контрольной точки прошло указанное *<количество операций>* или больше заданного *<количества минут>*, срабатывает новая контрольная точка.

Журнал сессий конфигурируется директивой:

```
syncprov-sessionlog <размер>
```

Здесь *<размер>* — это максимальное число записей, которые могут быть записаны в журнал сессий. После того, как журнал сессий был сконфигурирован, он автоматически используется для всех поисков LDAP Sync по указанной базе данных.

Обратите внимание, что использование журнала сессий требует поиска по атрибуту *entryUUID*. Установка eq-индекса на этот атрибут увеличит производительность журнала сессий на сервер-поставщике.

Вот более полный пример содержимого файла *slapd.conf*(5) на стороне поставщика:

```
database bdb
suffix dc=Example,dc=com
rootdn dc=Example,dc=com
directory /var/ldap/db
index objectclass,entryCSN,entryUUID eq

overlay syncprov
syncprov-checkpoint 100 10
syncprov-sessionlog 100
```

18.3.1.3. Настройка slapd потребителя

Директива настройки репликации syncrepl указывается в разделе database, описывающем контекст реплики, в файле *slapd.conf*(5). Механизм syncrepl не зависит от используемого механизма манипуляции данными, и данная директива может настраиваться с любым типом базы данных.

```
database hdb
suffix dc=Example,dc=com
rootdn dc=Example,dc=com
directory /var/ldap/db
index objectclass,entryCSN,entryUUID eq

syncrepl rid=123
        provider=ldap://provider.example.com:389
        type=refreshOnly
        interval=01:00:00:00
        searchbase="dc=example,dc=com"
        filter="(objectClass=organizationalPerson)"
        scope=sub
        attrs="cn,sn,ou,telephoneNumber,title,l"
        schemachecking=off
        bindmethod=simple
        binddn="cn=syncuser,dc=example,dc=com"
        credentials=secret
```

В этом примере потребитель будет подключаться к slapd поставщика на порт 389 по адресу ldap://provider.example.com для выполнения синхронизации в режиме опроса (*refreshOnly*) один раз в день. Он будет подключаться как `cn=syncuser,dc=example,dc=com` с использованием простой аутентификации с паролем "secret". Обратите внимание, что привилегии контроля доступа на поставщике должны быть заданы таким образом, чтобы `cn=syncuser,dc=example,dc=com` мог получать нужное ему реплицируемое содержимое каталога. Также права на поиск на стороне поставщика должны быть достаточными для того, чтобы позволить syncuser получать полную копию запрашиваемого содержимого каталога. Потребитель использует rootdn для записи в свою базу данных, поэтому с его стороны нет ограничений на запись всего реплицируемого содержимого.

Синхронизационный поиск в примере выше ищет записи с объектным классом organizationalPerson, во всём поддереве каталога, начиная от `dc=example,dc=com`. Запрашиваемые атрибуты: `cn`, `sn`, `ou`, `telephoneNumber`, `title`, и `l`. Проверка целостности схемы данных отключена, поэтому *slapd*(8) потребителя не будет проверять соответствие записи схеме данных при обработке обновлений, получаемых от *slapd*(8) поставщика.

Более детальную информацию о директиве syncrepl смотрите в подразделе syncrepl раздела Конфигурационный файл slapd данного руководства администратора.

18.3.1.4. Запуск slapd поставщика и потребителя

Перезагрузки *slapd*(8) поставщика не требуется. *contextCSN* создаётся автоматически по мере необходимости: он может сразу содержаться в LDIF-файле, загружаемом с помощью *slapadd*(8), создаваться во время изменений содержимого каталога, или генерироваться при первом поисковом запросе LDAP Sync, поступившем поставщику. При загрузке LDIF-файла, не содержащего *contextCSN*, для его генерации нужно использовать *slapadd*(8) с опцией *-w*. Это позволит серверу произвести первый запуск немного быстрее.

При запуске *slapd*(8) потребителя можно с помощью параметра командной строки *-c cookie* задать синхронизационное куки, если Вы хотите начать синхронизацию с какого-то определённого состояния. Куки — это разделённый запятыми список пар имя=значение. Поддерживаемые в настоящий момент поля syncrepl куки: *csn=<csn>* и *rid=<rid>*. *<csn>* представляет собой текущее состояние синхронизации (current synchronization state) реплики потребителя. *<rid>* (replica identifier) идентифицирует реплику потребителя локально на сервере потребителя. Он используется, чтобы сопоставить куки определению syncrepl в файле *slapd.conf*(5), имеющему такой же идентификатор реплики. *<rid>* должен быть не более 3 десятичных цифр. Куки, задаваемое в командной строке, переопределяет синхронизационное куки, хранящееся в базе данных реплики потребителя.

18.3.2. Дельта-syncrepl

18.3.2.1. Настройка поставщика дельта-syncrepl

Настройка дельта-syncrepl требует внесения изменений в конфигурацию как основного сервера, так и

сервера-реплики:

```
# Задаём DN реплики неограниченные права на чтение.  Этот ACL нужно
# объединить с другими определениями ACL, и/или поместить в определение
# базы данных. Часть "by * break"  вызывает оценку
# последующих правил. Детали в slapd.access(5).
access to *
   by dn.base="cn=replicator,dc=symas,dc=com" read
   by * break

# Указываем местонахождение модулей
modulepath /opt/symas/lib/openldap

# Загружаем механизм hdb
moduleload back_hdb.la

# Загружаем наложение accesslog
moduleload accesslog.la

# Загружаем наложение syncprov
moduleload syncprov.la

# Определение базы данных журнала доступа (Accesslog)
database hdb
suffix cn=accesslog
directory /db/accesslog
rootdn cn=accesslog
index default eq
index entryCSN,objectClass,reqEnd,reqResult,reqStart

overlay syncprov
syncprov-nopresent TRUE
syncprov-reloadhint TRUE

# Разрешим DN реплики осуществлять поиск без ограничений (в рамках БД Accesslog)
limits dn.exact="cn=replicator,dc=symas,dc=com" time.soft=unlimited time.hard=unlimited
size.soft=unlimited size.hard=unlimited

# Определение основной базы данных
database hdb
suffix "dc=symas,dc=com"
rootdn "cn=manager,dc=symas,dc=com"

## Далее поместим остальные требуемые настройки БД

# Индексы, задаваемые для syncprov
index entryCSN eq
index entryUUID eq

# Определение Поставщика syncrepl для основной БД
overlay syncprov
syncprov-checkpoint 1000 60

# Определение наложения accesslog для основной БД
overlay accesslog
logdb cn=accesslog
logops writes
logsuccess TRUE
# Сканируем БД accesslog каждый день и удаляем записи старше 7-ми дней
logpurge 07+00:00 01+00:00

# Разрешим DN реплики осуществлять поиск без ограничений (в рамках основной БД)
limits dn.exact="cn=replicator,dc=symas,dc=com" time.soft=unlimited time.hard=unlimited
size.soft=unlimited size.hard=unlimited
```

Для получения дополнительной информации смотрите соответствующие man-страницы (*slapo-accesslog*(5) и *slapd.conf*(5))

18.3.2.2. Настройка потребителя дельта-syncrepl

```
# Конфигурация базы данных реплики
database hdb
suffix "dc=symas,dc=com"
rootdn "cn=manager,dc=symas,dc=com"

## Далее поместим остальные требуемые настройки для реплики,
## например нужное индексирование

# Индексы, задаваемые для syncrepl
index entryUUID eq

# Директивы syncrepl
syncrepl  rid=0
```

```
            provider=ldap://ldapmaster.symas.com:389
            bindmethod=simple
            binddn="cn=replicator,dc=symas,dc=com"
            credentials=secret
            searchbase="dc=symas,dc=com"
            logbase="cn=accesslog"
            logfilter="(&(objectClass=auditWriteObject)(reqResult=0))"
            schemachecking=on
            type=refreshAndPersist
            retry="60 +"
            syncdata=accesslog

    # Отсылка обновлений на главный сервер
    updateref               ldap://ldapmaster.symas.com
```

В этой конфигурации предполагается, что учётная запись для аутентификации на сервере поставщика определена в этой же самой реплицируемой базе данных. Кроме того, у всех баз данных (основная, реплика и база хранения журнала доступа) должны быть правильно настроенные файлы *DB_CONFIG*, удовлетворяющие Вашим потребностям.

18.3.3. Разнонаправленная репликация с несколькими главными серверами

В следующем примере мы будет использовать 3 основных сервера. Последуем тесту **test050-syncrepl-multimaster** из пакета тестов OpenLDAP, и настроим *slapd(8)* через **cn=config**

Сначала настроим базу данных конфигурации:

```
dn: cn=config
objectClass: olcGlobal
cn: config
olcServerID: 1

dn: olcDatabase={0}config,cn=config
objectClass: olcDatabaseConfig
olcDatabase: {0}config
olcRootPW: secret
```

Второй и третий серверы, очевидно, будут иметь отличные от первого сервера olcServerID:

```
dn: cn=config
objectClass: olcGlobal
cn: config
olcServerID: 2

dn: olcDatabase={0}config,cn=config
objectClass: olcDatabaseConfig
olcDatabase: {0}config
olcRootPW: secret
```

Затем настроим syncrepl в качестве сервера-поставщика (поскольку все они главные серверы):

```
dn: cn=module,cn=config
objectClass: olcModuleList
cn: module
olcModulePath: /usr/local/libexec/openldap
olcModuleLoad: syncprov.la
```

Теперь настроим первый главный сервер (замените $URI1, $URI2 и $URI3 Вашими актуальными ldap url):

```
dn: cn=config
changetype: modify
replace: olcServerID
olcServerID: 1 $URI1
olcServerID: 2 $URI2
olcServerID: 3 $URI3

dn: olcOverlay=syncprov,olcDatabase={0}config,cn=config
changetype: add
objectClass: olcOverlayConfig
objectClass: olcSyncProvConfig
olcOverlay: syncprov

dn: olcDatabase={0}config,cn=config
changetype: modify
add: olcSyncRepl
olcSyncRepl: rid=001 provider=$URI1 binddn="cn=config" bindmethod=simple
  credentials=secret searchbase="cn=config" type=refreshAndPersist
  retry="5 5 300 5" timeout=1
```

```
    olcSyncRepl: rid=002 provider=$URI2 binddn="cn=config" bindmethod=simple
      credentials=secret searchbase="cn=config" type=refreshAndPersist
      retry="5 5 300 5" timeout=1
    olcSyncRepl: rid=003 provider=$URI3 binddn="cn=config" bindmethod=simple
      credentials=secret searchbase="cn=config" type=refreshAndPersist
      retry="5 5 300 5" timeout=1
    -
    add: olcMirrorMode
    olcMirrorMode: TRUE
```

Теперь запустим этот главный сервер и потребителей, также добавляя приведённый выше LDIF на первый сервер-потребитель, второй сервер-потребитель и т.д., и при изменении **cn=config**, она будет реплицироваться. Итак, Вы получили систему разнонаправленной репликации с несколькими главными серверами, реплицирующую базу данных конфигурации.

Но, поскольку, помимо конфигурационных, требуется реплицировать еще и реальные данные, добавим следующий LDIF на один из главных серверов (все настроенные нами потребители/поставщики применят ту же конфигурацию, так как они все синхронизируются). Опять же, замените переменные *${}* на требуемые Вам значения:

```
    dn: olcDatabase={1}$BACKEND,cn=config
    objectClass: olcDatabaseConfig
    objectClass: olc${BACKEND}Config
    olcDatabase: {1}$BACKEND
    olcSuffix: $BASEDN
    olcDbDirectory: ./db
    olcRootDN: $MANAGERDN
    olcRootPW: $PASSWD
    olcLimits: dn.exact="$MANAGERDN" time.soft=unlimited time.hard=unlimited size.soft=unlimited
size.hard=unlimited
    olcSyncRepl: rid=004 provider=$URI1 binddn="$MANAGERDN" bindmethod=simple
      credentials=$PASSWD searchbase="$BASEDN" type=refreshOnly
      interval=00:00:00:10 retry="5 5 300 5" timeout=1
    olcSyncRepl: rid=005 provider=$URI2 binddn="$MANAGERDN" bindmethod=simple
      credentials=$PASSWD searchbase="$BASEDN" type=refreshOnly
      interval=00:00:00:10 retry="5 5 300 5" timeout=1
    olcSyncRepl: rid=006 provider=$URI3 binddn="$MANAGERDN" bindmethod=simple
      credentials=$PASSWD searchbase="$BASEDN" type=refreshOnly
      interval=00:00:00:10 retry="5 5 300 5" timeout=1
    olcMirrorMode: TRUE

    dn: olcOverlay=syncprov,olcDatabase={1}${BACKEND},cn=config
    changetype: add
    objectClass: olcOverlayConfig
    objectClass: olcSyncProvConfig
    olcOverlay: syncprov
```

Замечание: Системное время на всех Ваших серверах должно быть чётко синхронизировано с помощью NTP http://www.ntp.org/, атомных часов или любой другой технологии синхронизации времени.

Замечание: Как сказано в *slapd-config*(5), указанные в директиве *olcSyncRepl* URL — это URL серверов, с которых нужно производить репликацию. Они должны точно соответствовать URL, на которых слушают *slapd* (согласно параметру командной строки *-h*). В противном случае slapd может попытаться произвести репликацию с самого себя, что приведёт к образованию петли.

18.3.4. Режим зеркала

Настройка режима зеркала, на самом деле, очень проста. Если Ваш slapd уже настроен в нормальный режим поставщика syncrepl, то все изменения будут заключаться в следующих двух директивах:

```
    mirrormode  on
    serverID    1
```

Замечание: Необходимо убедиться, что *serverID* каждого сервера-зеркала различаются, и добавить эту директиву в раздел глобальных директив конфигурации.

18.3.4.1. Настройка зеркальных серверов

Сначала нужно настроить поставщика syncrepl как описано в подразделе Настройка slapd поставщика.

Вот отрывок примера использования LDAP Sync Replication в режиме *refreshAndPersist*, касаемый репликации в режиме зеркала:

Зеркальный сервер 1:

```
# Глобальный раздел
serverID    1
# Раздел базы данных

# Директива syncrepl
syncrepl      rid=001
              provider=ldap://ldap-sid2.example.com
              bindmethod=simple
              binddn="cn=mirrormode,dc=example,dc=com"
              credentials=mirrormode
              searchbase="dc=example,dc=com"
              schemachecking=on
              type=refreshAndPersist
              retry="60 +"

mirrormode on
```

Зеркальный сервер 2:

```
# Глобальный раздел
serverID    2
# Раздел базы данных

# Директива syncrepl
syncrepl      rid=001
              provider=ldap://ldap-sid1.example.com
              bindmethod=simple
              binddn="cn=mirrormode,dc=example,dc=com"
              credentials=mirrormode
              searchbase="dc=example,dc=com"
              schemachecking=on
              type=refreshAndPersist
              retry="60 +"

mirrormode on
```

Как видите, всё очень просто; оба зеркальных сервера настроены **совершенно** одинаково, за исключением уникального *serverID*, и того, что выступая в роли потребителя, каждый из серверов указывает на другой сервер.

18.3.4.1.1. Настройка отказоустойчивости

Для этого есть 2 основных способа: 1. использование аппаратного устройства прокси/балансировщика нагрузки или специального программного прокси; 2. использование Back-LDAP прокси в качестве поставщика syncrepl.

Типичный пример каталога уровня предприятия:

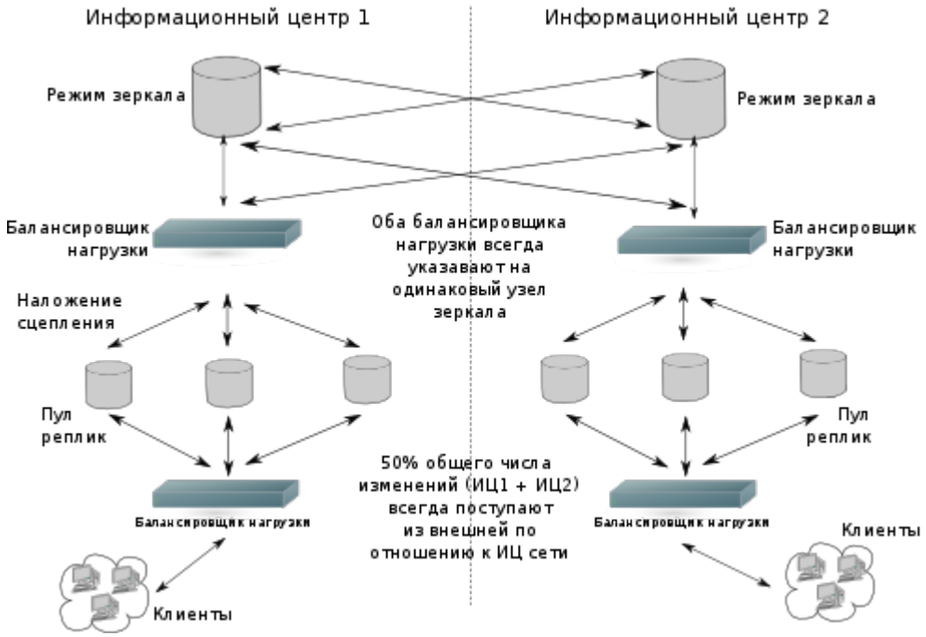

Рисунок 18.1: Конфигурация режима зеркала в двух центрах обработки информации

18.3.4.1.2. Настройка обычного потребителя

Настройка производится точно также, как описано в подразделе Настройка slapd потребителя. Можно произвести настройку репликации как в нормальном режиме syncrepl, так и в дельта-syncrepl режиме.

18.3.4.2. Резюме по зеркальному режиму

Теперь у Вас есть архитектура каталога, предоставляющая одновременно все гарантии целостности репликации с одним главным сервером и высокую доступность репликации с несколькими главными серверами.

18.3.5. Syncrepl прокси

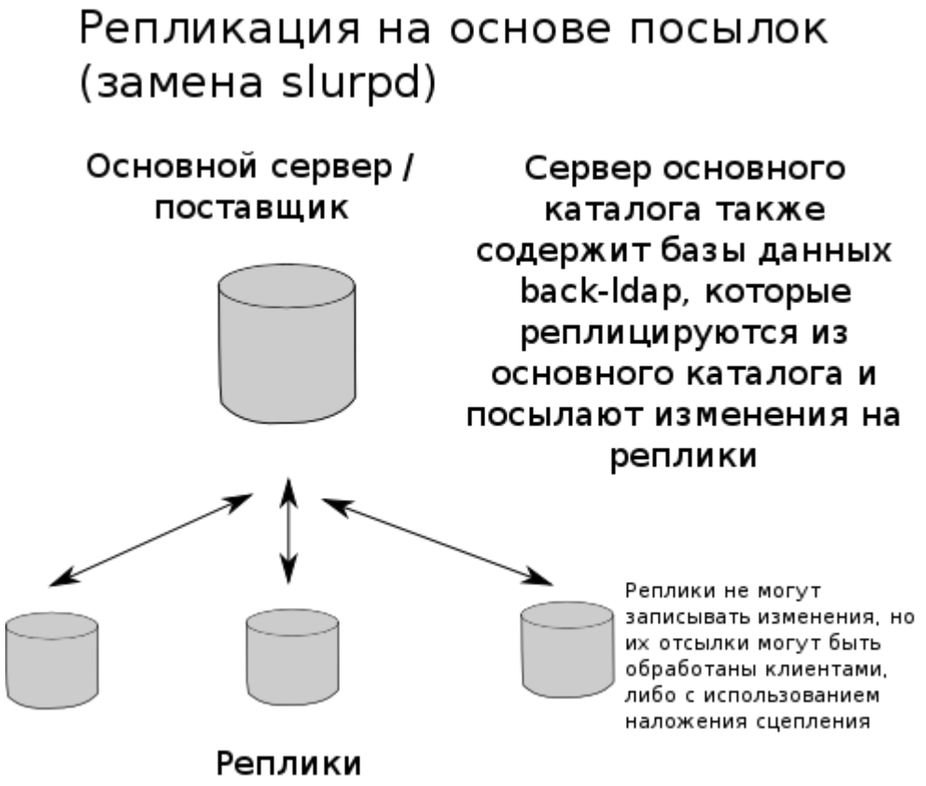

Рисунок 18.2: Замена slurpd

В следующем примере показано автономное решение репликации, основанной на посылках (поставщик и прокси на одном сервере):

```
        ########################################################################
        # Стандартный поставщик (основной сервер) OpenLDAP
        ########################################################################

        include     /usr/local/etc/openldap/schema/core.schema
        include     /usr/local/etc/openldap/schema/cosine.schema
        include     /usr/local/etc/openldap/schema/nis.schema
        include     /usr/local/etc/openldap/schema/inetorgperson.schema

        include     /usr/local/etc/openldap/slapd.acl

        modulepath  /usr/local/libexec/openldap
        moduleload  back_hdb.la
        moduleload  syncprov.la
        moduleload  back_monitor.la
        moduleload  back_ldap.la

        pidfile     /usr/local/var/slapd.pid
        argsfile    /usr/local/var/slapd.args

        loglevel    sync stats

        database    hdb
        suffix      "dc=suretecsystems,dc=com"
        directory   /usr/local/var/openldap-data

        checkpoint      1024 5
        cachesize       10000
        idlcachesize    10000

        index       objectClass eq
        # остальные индексы
        index       default     sub

        rootdn          "cn=admin,dc=suretecsystems,dc=com"
        rootpw          testing

        # индексы, специфичные для syncprov
        index entryCSN eq
        index entryUUID eq

        # настройка поставщика syncrepl для основной БД
        overlay syncprov
        syncprov-checkpoint 1000 60

        # Разрешим DN репликатора производить поиск без ограничений
        limits dn.exact="cn=replicator,dc=suretecsystems,dc=com" time.soft=unlimited time.hard=unlimited
size.soft=unlimited size.hard=unlimited

        database    monitor

        database    config
        rootpw          testing

        ########################################################################################
        # Прокси-потребитель, получающий данные через Syncrepl и отправляющий их через slapd-ldap
        ########################################################################################

        database        ldap
        # проигнорируем конфликты с другими БД, поскольку
        # нам требуется отправлять данные с тем же суффиксом
        hidden              on
        suffix          "dc=suretecsystems,dc=com"
        rootdn          "cn=slapd-ldap"
        uri             ldap://localhost:9012/

        lastmod         on

        # Нам не нужно никаких прав на доступ к данному DSA
        restrict        all

        acl-bind        bindmethod=simple
                        binddn="cn=replicator,dc=suretecsystems,dc=com"
                        credentials=testing

        syncrepl        rid=001
                        provider=ldap://localhost:9011/
                        binddn="cn=replicator,dc=suretecsystems,dc=com"
                        bindmethod=simple
```

```
                    credentials=testing
                    searchbase="dc=suretecsystems,dc=com"
                    type=refreshAndPersist
                    retry="5 5 300 5"

        overlay         syncprov
```

Настройка реплики для подобных случаев может быть такой:

```
        ###############################################################################
        # Стандартный подчинённый сервер OpenLDAP без использования Syncrepl
        ###############################################################################

        include     /usr/local/etc/openldap/schema/core.schema
        include     /usr/local/etc/openldap/schema/cosine.schema
        include     /usr/local/etc/openldap/schema/nis.schema
        include     /usr/local/etc/openldap/schema/inetorgperson.schema

        include     /usr/local/etc/openldap/slapd.acl

        modulepath  /usr/local/libexec/openldap
        moduleload  back_hdb.la
        moduleload  syncprov.la
        moduleload  back_monitor.la
        moduleload  back_ldap.la

        pidfile     /usr/local/var/slapd.pid
        argsfile    /usr/local/var/slapd.args

        loglevel    sync stats

        database    hdb
        suffix      "dc=suretecsystems,dc=com"
        directory   /usr/local/var/openldap-slave/data

        checkpoint      1024 5
        cachesize       10000
        idlcachesize    10000

        index       objectClass eq
        # rest of indexes
        index       default     sub

        rootdn          "cn=admin,dc=suretecsystems,dc=com"
        rootpw          testing

        # Разрешим DN репликатора производить поиск без ограничений
        limits dn.exact="cn=replicator,dc=suretecsystems,dc=com" time.soft=unlimited time.hard=unlimited
size.soft=unlimited size.hard=unlimited

        updatedn "cn=replicator,dc=suretecsystems,dc=com"

        # Отсылаем попытки внесения изменений на главный сервер
        updateref   ldap://localhost:9011

        database    monitor

        database    config
        rootpw          testing
```

Обратите внимание на директиву *updatedn* в приведённых настройках. Примерные ACL (`usr/local/etc/openldap/slapd.acl`) для данного случая могут быть такими:

```
        # Дадим DN репликатора неограниченный доступ на чтение.
        # Данный ACL нужно совместить с другими настройками ACL.

        access to *
             by dn.base="cn=replicator,dc=suretecsystems,dc=com" write
             by * break

        access to dn.base=""
                by * read

        access to dn.base="cn=Subschema"
                by * read

        access to dn.subtree="cn=Monitor"
            by dn.exact="uid=admin,dc=suretecsystems,dc=com" write
            by users read
            by * none

        access to *
                by self write
                by * read
```

Для поддержки большего числа реплик, просто добавьте секции *database ldap* в настройки прокси и увеличьте номер *syncrepl rid* соответственно.

Замечание: В отличие от использования нормального Syncrepl, Вы должны предварительно наполнить каталоги основного и подчинённых сервером одинаковыми данными

Если у Вас нет доступа для изменения настроек основного сервера, можно настроить отдельностоящий LDAP-прокси как показано на рисунке:

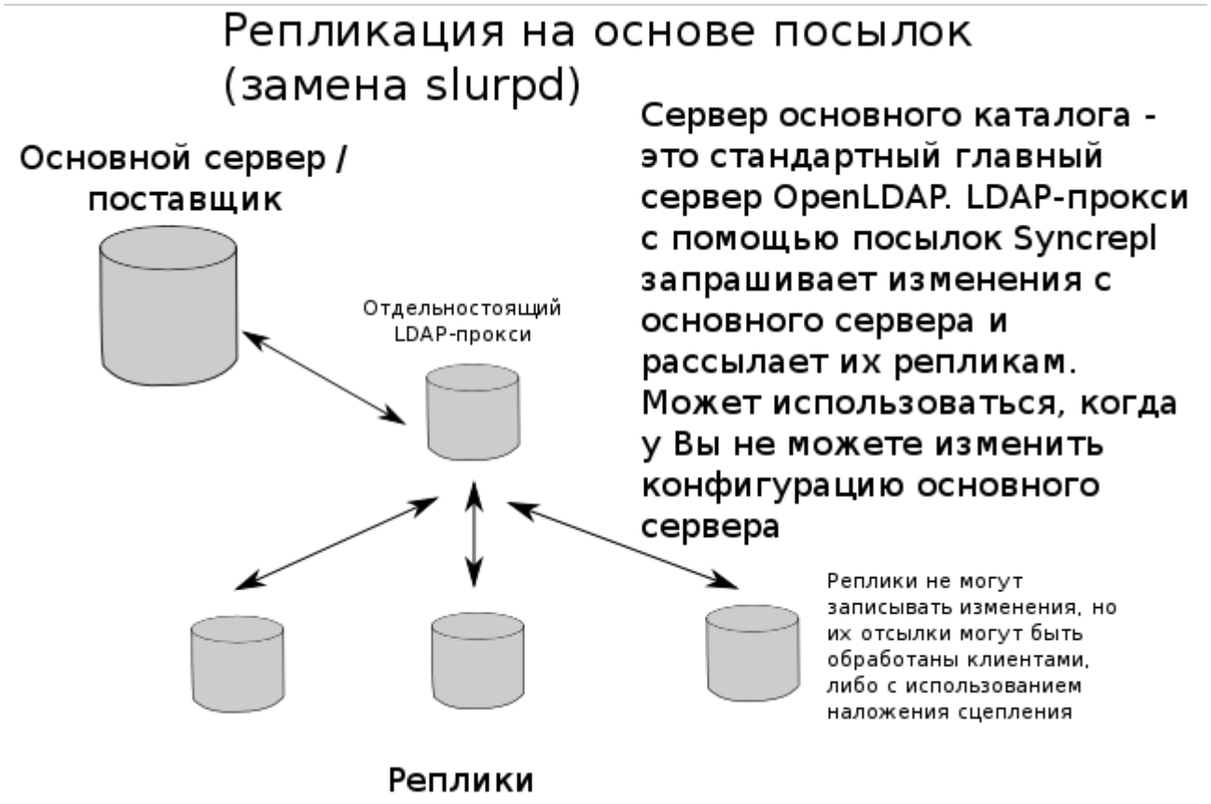

Рисунок 18.3: Версия замены slurpd с отдельностоящим LDAP-прокси

Пример настройки отдельностоящего LDAP-прокси:

```
include     /usr/local/etc/openldap/schema/core.schema
include     /usr/local/etc/openldap/schema/cosine.schema
include     /usr/local/etc/openldap/schema/nis.schema
include     /usr/local/etc/openldap/schema/inetorgperson.schema

include     /usr/local/etc/openldap/slapd.acl

modulepath  /usr/local/libexec/openldap
moduleload  syncprov.la
moduleload  back_ldap.la

########################################################################################
# Прокси-потребитель, получающий данные через Syncrepl и отправляющий их через slapd-ldap
########################################################################################

database        ldap
# проигнорируем конфликты с другими БД, поскольку
# нам требуется отправлять данные с тем же суффиксом
hidden              on
suffix          "dc=suretecsystems,dc=com"
rootdn          "cn=slapd-ldap"
uri             ldap://localhost:9012/

lastmod         on

# Нам не нужно никаких прав на доступ к данному DSA
restrict        all

acl-bind        bindmethod=simple
                binddn="cn=replicator,dc=suretecsystems,dc=com"
```

```
                credentials=testing

syncrepl        rid=001
                provider=ldap://localhost:9011/
                binddn="cn=replicator,dc=suretecsystems,dc=com"
                bindmethod=simple
                credentials=testing
                searchbase="dc=suretecsystems,dc=com"
                type=refreshAndPersist
                retry="5 5 300 5"

overlay         syncprov
```

Как видите, в использовании Syncrepl и *slapd-ldap(8)* Вы можете позволить разгуляться своему воображению, адаптируя технологии репликации к своей конкретной сетевой топологии.

К содержанию

19. Обслуживание

Системное администрирование - это и есть обслуживание; таким образом, наши рассуждения о том, как правильно обслуживать системы OpenLDAP, вполне закономерны.

19.1. Резервное копирование каталога

Стратегия резервного копирования во многом определяется количеством изменений базы данных, а также тем, насколько большую часть этих изменений администратор готов потерять в случае катастрофического краха системы. Существует два основных подхода:

1. Резервное копирование самой базы данных Berkeley и периодическое резервное копирование файлов журналов транзакций:

Berkeley DB ведёт журналы транзакций, которые могут быть использованы для реконструкции изменений, начиная с заданной точки во времени. Например, если администратора не устраивает потеря изменений больше чем за один час, он может останавливать сервер в ночное время, копировать файлы базы данных Berkeley в надёжное место, и вновь запускать сервер. Затем ежечасно он может принудительно вызывать сброс базы данных в контрольной точке, собирать файлы журнала, созданные за последний час, и копировать их в надёжной место. С помощью инструмента db_recover можно, на основании собранных файлов журнала в сочетании с предыдущей резервной копией базы данных, реконструировать базу данных на момент последнего сбора файлов журнала. Этот подход обеспечивает хорошую сохранность данных с минимальными затратами дискового пространства.

2. Периодический запуск slapcat и резервное копирование LDIF-файла:

Slapcat может быть запущен во время работы slapd. Однако, есть риск возникновения нарушений целостности базы данных, причём не с точки зрения slapd, а с точки зрения приложений, использующих LDAP. Например, если пополняющее каталог приложение выполняет задачи, состоящие из нескольких операций LDAP, и slapcat выполнялся одновременно с этими операциями, то могут произойти нарушения целостности в базе данных LDAP с точки зрения этого приложения и приложений, зависящих от него. Следовательно, нужно убедиться, чтобы подобное не происходило. Один из путей решения - перевод базы данных в режим только для чтения на время выполнения slapcat. Другой недостаток этого подхода состоит в том, что созданные файлы LDIF могут быть довольно большими и накопление ежедневных резервных копий может занять значительный объём дискового пространства.

Можно использовать *slapcat*(8) для создания отдельных файлов LDIF для каждой из Ваших баз данных back-bdb или back-hdb.

```
slapcat -f slapd.conf -b "dc=example,dc=com"
```

Для back-bdb и back-hdb данная команда может быть выполнена при запущенном slapd(8).

Дополнительная информация о резервном копировании Berkeley DB, включая db_recover и другое, будет позже.

19.2. Журналы Berkeley DB

Файлы журнала Berkeley DB имеют тенденцию разрастаться, и администратору приходится что-то с этим делать. Данная процедура известна как архивирование или ротация файлов журнала.

Примечание: Реальная ротация файлов журнала производится механизмом Berkeley DB.

Журналы текущих транзакций должны сохраняться в файлах для того, чтобы база данных могла быть восстановлена в случае сбоя приложения. Администраторы могут изменить ограничения на размер одного файла журнала (по умолчанию 10MB), и настроить автоматическое удаление старых файлов журнала путём установки окружения базы данных (смотрите ниже). Причиной того, что Berkeley DB по умолчанию никогда не удаляет никаких файлов журнала, является то, что администратор может захотеть сделать резервные копии файлов журнала перед удалением, чтобы была возможность восстановления базы данных даже после катастрофического отказа, такого, как повреждение файловой системы.

Имена файлов журнала - `log.XXXXXXXXXX` (X - это цифра). По умолчанию файлы журнала размещаются в директории с базой данных. Инструмент `db_archive` знает, какие файлы журнала используются в текущих транзакциях, а какие нет. Администраторы могут копировать неиспользуемые файлы в надёжное место и удалять их. Чтобы они удалялись автоматически, поместите директиву set_flags *DB_LOG_AUTOREMOVE* в файл `DB_CONFIG`.

Примечание: Если файлы журнала удаляются автоматически, восстановление после катастрофического сбоя скорее всего будет невозможно.

Файлы с именами `__db.001`, `__db.002` и т.д. - это просто разделяемые участки памяти (или что-либо ещё). Это НЕ ЖУРНАЛЫ, и они должны быть оставлены в покое. Не переживайте за них, они не разрастаются так, как файлы журналов.

Чтобы понять, как работает интерфейс `db_archive`, изучите главу 9 руководства Berkeley DB. В частности, рекомендуется ознакомиться со следующими подглавами:

- Архивирование баз данных и файлов журнала - http://www.oracle.com/technology/documentation/berkeley-db/db/ref/transapp/archival.html;
- Удаление файлов журнала - http://www.oracle.com/technology/documentation/berkeley-db/db/ref/transapp/logfile.html;
- Процедура восстановления - http://www.oracle.com/technology/documentation/berkeley-db/db/ref/transapp/recovery.html;
- Оперативное устранение отказов - http://www.oracle.com/technology/documentation/berkeley-db/db/ref/transapp/hotfail.html
- Полный список флагов Berkeley DB - http://www.oracle.com/technology/documentation/berkeley-db/db/api_c/env_set_flags.html.

Продвинутые инсталляции могут использовать специальные установки окружения для тонкой настройки некоторых опций Berkeley DB (изменения ограничения размера файлов журнала и т.п.). Это может быть сделано с помощью файла `DB_CONFIG`. Этот волшебный файл может быть создан в директории с базой данных механизма манипуляции данными BDB, определённой в *slapd.conf*(5). Более подробную информацию об этом файле можно найти в главе, называемой 'File', руководства Berkeley DB. Конкретные директивы можно найти в интерфейсе C, ищите вызовы *DB_ENV->set_XXXX* .

Примечание: опции, задаваемые в файле `DB_CONFIG`, переопределяют опции, заданные в OpenLDAP. Используйте их с большой осторожностью. Если Вы не знаете, что делают эти опции, не используйте их.

Применяя `DB_CONFIG`, можно получить следующие преимущества:

- размещение файлов с данными и файлов журнала на разных носителях (дисках) для повышения производительности и/или надёжности;
- тонкая настройка некоторых специфических опций (таких, как размеры разделяемых участков памяти);
- настройка ограничений размера файлов журнала (перед тем, как это делать, прочитайте, пожалуйста, главу 'Log file limits' руководства Berkeley DB).

Для выяснения передовой практики резервного копирования BDB настоятельно рекомендуется полностью прочитать главу 9 руководства Berkeley DB: 'Berkeley DB Transactional Data Store Applications' ('Приложения хранения данных с транзакциями Berkeley DB'). Эта глава состоит из набора маленьких страниц с примерами на языке C. Люди, далёкие от программирования, могут пропустить эти примеры без потери необходимых знаний.

19.3. Срабатывание контрольных точек

Если Вы поместили в slapd.conf "checkpoint 1024 5" (то есть, срабатывание контрольной точки после 1024Kb или 5 минут), это не означает, что контрольная точка будет срабатывать каждые пять минут, как Вы могли бы подумать. Howard дал следующие пояснения:

'В OpenLDAP 2.1 и 2.2 директива checkpoint работает следующим образом - *если была операция записи* и с момента срабатывания последней контрольной точки прошло более чем <check> минут, выполняется срабатывание контрольной точки. Если после записи прошло более чем <check> минут без выполнения какой-либо другой операции записи, срабатывание контрольной точки не выполняется; таким образом, существует возможность потерять последнюю произошедшую операцию записи'.

Другими словами, операция записи, произошедшая менее чем через "check" минут после последнего срабатывания контрольной точки, не будет сброшена на диск в контрольной точке, пока следующая операция записи не произойдёт более чем через "check" минут после последнего срабатывания контрольной точки.

В версии 2.3 это было изменено таким образом, чтобы срабатывание действительно происходило так часто, как оно настроено; вариант решения для предыдущих версий - выполнять "db_checkpoint" из cron так часто, как это требуется, скажем, каждые 5 минут.

19.4. Миграция

В зависимости от типа развёртывания Вашего каталога, простейшие шаги, которые необходимо выполнить для миграции между версиями или обновления, могут быть такими:

1. **Перед началом процедуры остановите сервер текущей версии;**
2. **Экспортируйте данные каталога с помощью slapcat;**
3. **Очистите текущую директорию с данными (/usr/local/var/openldap-data/), оставив DB_CONFIG на месте;**
4. **Выполните обновление программного обеспечения;**
5. **Импортируйте экспортированные данные обратно в каталог с помощью slapadd;**
6. **Запустите сервер.**

Очевидно, это не совсем подходит при сложных типах развёртываний, таких как Режим зеркала или Разнонаправленная репликация с несколькими главными серверами, в таких случаях может помочь изучение предыдущих разделов. Кроме того, Вы всегда можете воспользоваться коммерческой поддержкой или поддержкой сообщества. Также посмотрите раздел Устранение неполадок.

К содержанию

20. Мониторинг

slapd(8) поддерживает опциональный интерфейс мониторинга LDAP, который можно использовать для получения информации о текущем состоянии Вашего экземпляра *slapd*. Например, данный интерфейс позволит определить, сколько клиентов в настоящий момент подключено к серверу. Информация мониторинга предоставляется с помощью специального механизма манипуляции данными *monitor*, для которого существует man-страница *slapd-monitor*(5).

При включенном интерфейсе мониторинга для доступа к информации, предоставляемой механизмом манипуляции данными *monitor*, можно использовать клиенты LDAP. На данную информацию может налагаться контроль доступа и другие виды контроля.

Во включенном состоянии механизм *monitor* в ответ на поисковые запросы к поддереву *cn=Monitor* динамически создаёт и возвращает объекты. Каждый объект содержит информацию о конкретном аспекте сервера. Эта информация содержится в комбинации пользовательских и операционных атрибутов. Доступ к данной информации можно осуществить с помощью *ldapsearch(1)*, любого LDAP-браузера общего назначения или с помощью специализированных инструментов мониторинга. В подразделе Доступ к информации мониторинга дано краткое руководство по использованию *ldapsearch*(1) для доступа к информации мониторинга, а в подразделе Информация мониторинга детально описывается информационная база мониторинга и её организация.

Хотя поддержка механизма манипуляции данными monitor включена в сборку slapd(8) по умолчанию, для её активации требуется произвести некоторые настройки. Это может быть сделано как при использовании `cn=config`, так и при использовании *slapd.conf*(5). Первый вариант обсуждается в подразделе Настройка мониторинга при использовании cn=config(5) этого раздела. Второй вариант - в подразделе Настройка мониторинга при использовании slapd.conf(5) этого раздела. В этих подразделах подразумевается, что механизм monitor вкомпилирован в *slapd* (то есть при запуске `configure` была указана опция `--enable-monitor=yes`, что является значением по умолчанию). Если же механизм monitor был собран как модуль (то есть при запуске `configure` была указана опция `--enable-monitor=mod`), данный модуль должен быть загружен. Загрузка модулей обсуждается в разделах Настройка slapd и Конфигурационный файл slapd.

20.1. Настройка мониторинга при использовании cn=config(5)

Этот раздел еще не написан.

20.2. Настройка мониторинга при использовании slapd.conf(5)

Настройка поддержки мониторинга LDAP через slapd.conf(5) достаточно проста.

Во-первых, убедитесь, что файл набора схемы *core.schema* подключен в Вашем файле *slapd.conf*(5). Это требуется для механизма манипуляции данными *monitor*.

Во-вторых, инициализируйте *monitor backend* добавлением директивы *database monitor* ниже Ваших существующих секций баз данных. Например:

```
database monitor
```

Наконец, добавьте дополнительные глобальные директивы или директивы базы данных по мере необходимости.

Как и большинство других баз данных, базы данных механизма манипуляции данными monitor подлежат контролю доступа и другим видам административного контроля slapd(8). Поскольку некоторую часть информации мониторинга нежелательно выносить на всеобщее обозрение, в общем случае рекомендуется предоставить доступ к cn=monitor только администраторам каталога и их уполномоченным по мониторингу. Добавление директивы *access* непосредственно за директивой *database monitor* - простой и эффективный подход для управления доступом. Например, добавление следующей директивы *access* непосредственно за директивой *database monitor* ограничивает доступ всем, за исключением указанного менеджера каталога.

```
access to *
        by dn.exact="cn=Manager,dc=example,dc=com
        by * none
```

Дополнительную информацию по настройке контроля доступа в *slapd*(8) можно найти в подразделе *Директива access* раздела Конфигурационный файл slapd и в man-странице *slapd.access*(5).

После перезапуска *slapd*(8) Вы можете приступить к изучению информации мониторинга, представленной в `cn=Monitor`, как описано в подразделе Доступ к информации мониторинга данного раздела.

Можно убедиться в том, что slapd(8) правильно настроен на предоставление информации мониторинга, попытавшись прочитать объект `cn=monitor`. Например, если следующая команда *ldapsearch*(1) возвратит объект cn=monitor (без атрибутов, как и было запрошено), значит всё работает.

```
ldapsearch -x -D 'cn=Manager,dc=example,dc=com' -W \
        -b 'cn=Monitor' -s base 1.1
```

Обратите внимание, что в отличие от баз данных механизмов манипуляции данными общего назначения, в данном случае суффикс базы данных назначен жёстко. Это всегда `cn=Monitor`. Поэтому не нужно указывать директиву *suffix*. Кроме того, механизм monitor не может быть инициирован несколько раз. Поэтому в конфигурации сервера может быть только одно (или ноль) вхождений `database monitor`.

20.3. Доступ к информации мониторинга

Как уже было сказано, во включенном состоянии механизм *monitor* динамически создаёт и возвращает объекты в ответ на поисковый запрос к поддереву *cn=Monitor*. Каждый объект содержит информацию о конкретном аспекте сервера. Эта информация содержится в комбинации пользовательских и операционных атрибутов. Доступ к данной информации можно осуществить с помощью *ldapsearch(1)*, любого LDAP-браузера общего назначения или с помощью специализированных инструментов мониторинга.

Этот раздел содержит краткое руководство по использованию *ldapsearch*(1) для доступа к информации мониторинга.

Для просмотра какого-либо конкретного объекта мониторинга нужно выполнить операцию поиска данного объекта с диапазоном baseObject и фильтром `(objectClass=*)`. Поскольку информация мониторинга содержится в комбинации пользовательских и операционных атрибутов, требуется запросить все пользовательские (`'*'`) и все операционные (`'+'`) атрибуты. Например, чтобы прочитать сам объект `cn=Monitor`, выполните такую команду *ldapsearch*(1) (с изменениями в соответствии с Вашей конфигурацией):

```
ldapsearch -x -D 'cn=Manager,dc=example,dc=com' -W \
        -b 'cn=Monitor' -s base '(objectClass=*)' '*' '+'
```

При запуске на Вашем сервере такая команда выдаст что-то вроде этого:

```
dn: cn=Monitor
objectClass: monitorServer
structuralObjectClass: monitorServer
cn: Monitor
creatorsName:
modifiersName:
createTimestamp: 20061208223558Z
modifyTimestamp: 20061208223558Z
description: This subtree contains monitoring/managing objects.
description: This object contains information about this server.
description: Most of the information is held in operational attributes, which
 must be explicitly requested.
monitoredInfo: OpenLDAP: slapd 2.4 (Dec  7 2006 17:30:29)
entryDN: cn=Monitor
subschemaSubentry: cn=Subschema
hasSubordinates: TRUE
```

Если в данный момент Вас не интересуют сразу все атрибуты, можно уменьшить их число, указав при запросе, какие атрибуты должны быть возвращены. Например, вместо вывода всех пользовательских и операционных атрибутов, можно запросить все атрибуты, разрешённые объектным классом *monitorServer* (`@monitorServer`):

```
ldapsearch -x -D 'cn=Manager,dc=example,dc=com' -W \
        -b 'cn=Monitor' -s base '(objectClass=*)' '@monitorServer'
```

Это ограничит вывод следующими строками:

```
dn: cn=Monitor
objectClass: monitorServer
cn: Monitor
description: This subtree contains monitoring/managing objects.
description: This object contains information about this server.
description: Most of the information is held in operational attributes, which
 must be explicitly requested.
monitoredInfo: OpenLDAP: slapd 2.X (Dec  7 2006 17:30:29)
```

Чтобы вернуть имена всех объектов мониторинга, нужно выполнить поиск `cn=Monitor` с диапазоном sub и фильтром `(objectClass=*)` и запросить, чтобы атрибуты не возвращались (`1.1`):

```
ldapsearch -x -D 'cn=Manager,dc=example,dc=com' -W -b 'cn=Monitor' -s sub 1.1
```

При выполнении этой команды Вы обнаружите, что в поддереве *cn=Monitor* есть много объектов. В следующем подразделе описано несколько объектов мониторинга, к которым обычно осуществляется доступ.

20.4. Информация мониторинга

Механизм манипуляции данными *monitor* предоставляет огромное количество полезной для мониторинга slapd(8) информации, содержащейся в наборе объектов мониторинга. Каждый объект содержит информацию о конкретных аспектах сервера, таких как механизмы манипуляции данными, соединения или потоки. Некоторые объекты служат контейнерами для других объектов и используются для создания иерархии объектов.

Корневой объект в этой иерархии - cn=Monitor. Хотя этот объект в первую очередь служит контейнером для других объектов, большинство из которых также являются контейнерами, он содержит информацию о данном сервере. В частности, он предоставляет информацию о версии slapd(8). Например:

```
dn: cn=Monitor
monitoredInfo: OpenLDAP: slapd 2.X (Dec  7 2006 17:30:29)
```

Примечание: Примеры в данном подразделе (и его подразделах) были обрезаны, чтобы показать только ключевую информацию.

20.4.1. Backends

Сам объект `cn=Backends,cn=Monitor` предоставляет список доступных механизмов манипуляции данными. В этот список входят как вкомпилированные, так и загружаемые в виде модулей механизмы манипуляции данными. Например:

```
dn: cn=Backends,cn=Monitor
monitoredInfo: config
monitoredInfo: ldif
monitoredInfo: monitor
monitoredInfo: bdb
monitoredInfo: hdb
```

В данном примере доступны механизмы *config*, *ldif*, *monitor*, *bdb* и *hdb*.

Объект `cn=Backends,cn=Monitor` также является контейнером для объектов доступных механизмов манипуляции данными. Каждый объект доступного механизма содержит информацию о конкретном объекте механизма манипуляции данными. Например:

```
dn: cn=Backend 0,cn=Backends,cn=Monitor
monitoredInfo: config
monitorRuntimeConfig: TRUE
supportedControl: 2.16.840.1.113730.3.4.2
seeAlso: cn=Database 0,cn=Databases,cn=Monitor

dn: cn=Backend 1,cn=Backends,cn=Monitor
monitoredInfo: ldif
monitorRuntimeConfig: TRUE
```

```
supportedControl: 2.16.840.1.113730.3.4.2

dn: cn=Backend 2,cn=Backends,cn=Monitor
monitoredInfo: monitor
monitorRuntimeConfig: TRUE
supportedControl: 2.16.840.1.113730.3.4.2
seeAlso: cn=Database 2,cn=Databases,cn=Monitor

dn: cn=Backend 3,cn=Backends,cn=Monitor
monitoredInfo: bdb
monitorRuntimeConfig: TRUE
supportedControl: 1.3.6.1.1.12
supportedControl: 2.16.840.1.113730.3.4.2
supportedControl: 1.3.6.1.4.1.4203.666.5.2
supportedControl: 1.2.840.113556.1.4.319
supportedControl: 1.3.6.1.1.13.1
supportedControl: 1.3.6.1.1.13.2
supportedControl: 1.3.6.1.4.1.4203.1.10.1
supportedControl: 1.2.840.113556.1.4.1413
supportedControl: 1.3.6.1.4.1.4203.666.11.7.2
seeAlso: cn=Database 1,cn=Databases,cn=Monitor

dn: cn=Backend 4,cn=Backends,cn=Monitor
monitoredInfo: hdb
monitorRuntimeConfig: TRUE
supportedControl: 1.3.6.1.1.12
supportedControl: 2.16.840.1.113730.3.4.2
supportedControl: 1.3.6.1.4.1.4203.666.5.2
supportedControl: 1.2.840.113556.1.4.319
supportedControl: 1.3.6.1.1.13.1
supportedControl: 1.3.6.1.1.13.2
supportedControl: 1.3.6.1.4.1.4203.1.10.1
supportedControl: 1.2.840.113556.1.4.1413
supportedControl: 1.3.6.1.4.1.4203.666.11.7.2
```

Для каждого из этих объектов атрибут monitorInfo указывает, сведения о каком механизме манипуляции данными содержится в данном объекте. Например, объект `cn=Backend 3,cn=Backends,cn=Monitor` в данном примере содержит информацию о механизме *bdb*.

Атрибут	**Описание**
monitoredInfo	название механизма манипуляции данными
supportedControl	поддерживаемые расширения элементов управления LDAP
seeAlso	Объекты базы данных, являющиеся экземплярами этого механизма манипуляции данными

20.4.2. Connections

Основная запись пуста; она должна содержать некоторые статистические данные о количестве соединений.

Динамические дочерние записи создаются для каждого открытого соединения и содержат статистические данные по активности этого соединения (формат будет подробно описан позже). Существуют две специальные дочерние записи, показывающие соответственно общее количество соединений и количество текущих соединений.

Пример:

Всего соединений:

```
dn: cn=Total,cn=Connections,cn=Monitor
structuralObjectClass: monitorCounterObject
monitorCounter: 4
entryDN: cn=Total,cn=Connections,cn=Monitor
subschemaSubentry: cn=Subschema
hasSubordinates: FALSE
```

Текущие соединения:

```
dn: cn=Current,cn=Connections,cn=Monitor
structuralObjectClass: monitorCounterObject
monitorCounter: 2
entryDN: cn=Current,cn=Connections,cn=Monitor
subschemaSubentry: cn=Subschema
hasSubordinates: FALSE
```

20.4.3. Databases

Основная запись содержит контексты именования всех настроенных баз данных; дочерние записи содержат тип и контекст именования для каждой отдельной базы данных.

Пример:

```
dn: cn=Database 2,cn=Databases,cn=Monitor
structuralObjectClass: monitoredObject
monitoredInfo: monitor
monitorIsShadow: FALSE
monitorContext: cn=Monitor
readOnly: FALSE
entryDN: cn=Database 2,cn=Databases,cn=Monitor
subschemaSubentry: cn=Subschema
hasSubordinates: FALSE
```

20.4.4. Listener

Содержит описание интерфейсов, на которых сервер в данный момент ожидает соединения:

```
dn: cn=Listener 0,cn=Listeners,cn=Monitor
structuralObjectClass: monitoredObject
monitorConnectionLocalAddress: IP=0.0.0.0:389
entryDN: cn=Listener 0,cn=Listeners,cn=Monitor
subschemaSubentry: cn=Subschema
hasSubordinates: FALSE
```

20.4.5. Log

Содержит активные в настоящее время элементы журнала. Подсистема *Log* позволяет пользователям выполнять операции модификации атрибута *description*, значения которого *ДОЛЖНЫ* быть из списка допустимых уровней журналирования:

```
Trace
Packets
Args
Conns
BER
Filter
Config
ACL
Stats
Stats2
Shell
Parse
Sync
```

Эти значения могут быть добавлены, заменены или удалены; они влияют на то, какие сообщения будут посылаться системе syslog. Для разных модулей могут задаваться разные настройки.

20.4.6. Operations

Показывает некоторые статистические данные по операциям, выполняемым сервером:

```
Initiated
Completed
```

для каждого типа операций, а именно:

```
Bind
Unbind
Add
Delete
Modrdn
Modify
Compare
Search
Abandon
Extended
```

Для разных типов операций выходные данные отличаются друг от друга, поэтому примеры тут не

приводятся. Попробуйте сделать запросы самостоятельно по инструкциям из подраздела Доступ к информации мониторинга.

20.4.7. Overlays

Основная запись содержит типы наложений, доступных во время исполнения; дочерние записи для каждого наложения содержат тип наложения.

Также в основной записи могут содержаться загруженные модули, если включена поддержка динамических наложений:

```
# Overlays, Monitor
dn: cn=Overlays,cn=Monitor
structuralObjectClass: monitorContainer
monitoredInfo: syncprov
monitoredInfo: accesslog
monitoredInfo: glue
entryDN: cn=Overlays,cn=Monitor
subschemaSubentry: cn=Subschema
hasSubordinates: TRUE
```

20.4.8. SASL

В настоящий момент запись пуста.

20.4.9. Statistics

Показывает некоторую статистику по данным, отправляемым сервером:

```
Bytes
PDU
Entries
Referrals
```

Пример:

```
# Entries, Statistics, Monitor
dn: cn=Entries,cn=Statistics,cn=Monitor
structuralObjectClass: monitorCounterObject
monitorCounter: 612248
entryDN: cn=Entries,cn=Statistics,cn=Monitor
subschemaSubentry: cn=Subschema
hasSubordinates: FALSE
```

20.4.10. Threads

Содержит максимальное количество потоков, включенных во время запуска, и текущую загруженность.

Пример:

```
# Max, Threads, Monitor
dn: cn=Max,cn=Threads,cn=Monitor
structuralObjectClass: monitoredObject
monitoredInfo: 16
entryDN: cn=Max,cn=Threads,cn=Monitor
subschemaSubentry: cn=Subschema
hasSubordinates: FALSE
```

20.4.11. Time

Содержит две дочерние записи со временем запуска сервера и текущим временем сервера.

Пример:

Время запуска:

```
dn: cn=Start,cn=Time,cn=Monitor
structuralObjectClass: monitoredObject
```

```
monitorTimestamp: 20061205124040Z
entryDN: cn=Start,cn=Time,cn=Monitor
subschemaSubentry: cn=Subschema
hasSubordinates: FALSE
```

Текущее время:

```
dn: cn=Current,cn=Time,cn=Monitor
structuralObjectClass: monitoredObject
monitorTimestamp: 20061207120624Z
entryDN: cn=Current,cn=Time,cn=Monitor
subschemaSubentry: cn=Subschema
hasSubordinates: FALSE
```

20.4.12. TLS

В настоящий момент запись пуста.

20.4.13. Waiters

Содержит число текущих ожидающих обработки запросов.

Пример:

Ожидающие обработки запросы на чтение:

```
dn: cn=Read,cn=Waiters,cn=Monitor
structuralObjectClass: monitorCounterObject
monitorCounter: 7
entryDN: cn=Read,cn=Waiters,cn=Monitor
subschemaSubentry: cn=Subschema
hasSubordinates: FALSE
```

Ожидающие обработки запросы на запись:

```
dn: cn=Write,cn=Waiters,cn=Monitor
structuralObjectClass: monitorCounterObject
monitorCounter: 0
entryDN: cn=Write,cn=Waiters,cn=Monitor
subschemaSubentry: cn=Subschema
hasSubordinates: FALSE
```

Сюда следует добавить новые позиции мониторинга, их описание, ссылки на man-страницы и примеры выходных данных.

К содержанию

21. Настройка производительности

Этот раздел, возможно, один из самых важных в данном руководстве, поскольку, если Вы не настроили *slapd*(8) должным образом или не знаете, как спроектировать Ваш каталог и его окружение, производительность может оказаться очень низкой.

Прочтение и понимание инструкций и информации из следующих подразделов, а также эксперименты на их основе дадут Вам полное представление о том, как адаптировать Ваш сервер каталогов в соответствии с Вашими требованиями.

Необходимо отметить, что представленная ниже информация постепенно собиралась из основанного нашим сообществом списка FAQ. Очевидно, какую большую пользу читателю может принести данный опыт и рекомендации, почерпнутые из работы со службами каталогов в реальном мире.

21.1. Факторы, влияющие на производительность

Различные факторы могут влиять на то, какая производительность будет у Вашего каталога на выбранном Вами оборудовании и окружении. Мы постараемся обсудить это здесь.

21.1.1. Оперативная память

Увеличьте размер Вашего кэша, чтобы использовать всю доступную для работы каталога оперативную память. Если есть возможность, увеличьте количество оперативной памяти в системе.

Советы по настройке кэша BDB смотрите в подразделе "Кэширование". Обратите внимание, что LMDB не использует кэш и не имеет никаких параметров настройки, так что при использовании LMDB подразделом "Кэширование" можно пренебречь.

21.1.2. Диски

Используйте быстрые файловые системы и проведите собственное тестирование, чтобы выявить, какие типы фаловых систем лучше работают с Вашим уровнем нагрузки. Наши тесты на Linux показали, что EXT2 и JFS, как правило, обеспечивают лучшую производительность операций записи, нежели любые другие файловые системы, в том числе такие новые, как EXT4, BTRFS и т.п.

Используйте быстрые подсистемы. Разместите каждую базу данных и журналы на разные диски (для BDB это настраивается с помощью файла *DB_CONFIG*):

```
# Директория для размещения данных
set_data_dir /data/db

# Установки журнала транзакций
set_lg_dir /logs
```

21.1.3. Топология сети

http://www.openldap.org/faq/data/cache/363.html

Здесь будет иллюстрация.

21.1.4. Проектирование структуры каталога

Здесь будут ссылки на другие разделы и иллюстрация хорошего/плохого макета каталога.

21.1.5. Предполагаемое использование

Здесь будет обсуждение.

21.2. Индексирование

21.2.1. Разберёмся, как работает поиск

Если в поисковом фильтре используются проиндексированные атрибуты, то в процессе поиска читаются индексы и непосредственно выбираются те записи, на которые эти индексы ссылаются. Если атрибуты в фильтре не были проиндексированы, тогда процесс поиска должен прочитать каждую отдельную запись в целевом диапазоне и проверить, соответствует ли она условию фильтра. Очевидно, при правильном использовании индексации, она помогает избегать выполнения большого количества работы.

21.2.2. Что нужно индексировать

Вы должны создать индексы в соответствии с фактическими условиями фильтров, используемых в поисковых запросах.

```
index cn,sn,givenname,mail eq
```

Затем каждый индексируемый атрибут может быть дополнительно настроен путём подбора типов индексов, которые необходимо сгенерировать для этого атрибута. Например, поиски соответствия подстроке и примерного соответствия для атрибута organization (o) вряд ли будут иметь большой смысл (и осуществляться очень часто). А поиски по атрибуту *userPassword*, вероятно, вообще лишены всякого смысла.

Основное правило: не переусердствуйте с индексами. Неиспользуемые индексы также требуют времени на обслуживание, и, следовательно, могут только замедлять ход событий.

За дополнительной информацией обращайтесь к man-страницам *slapd.conf*(8) и *slapdindex*(8).

21.2.3. Индексы наличия

Если клиентское приложение использует фильтры наличия и целевой атрибут присутствует в большинстве записей в Вашем целевом диапазоне, то все эти записи будут прочитаны в любом случае, поскольку все они попадают в результирующее подмножество. В поддереве, где все 100% записей будут содержать одни и те же атрибуты, индексы наличия не делают абсолютно НИЧЕГО полезного для поиска, поскольку все 100% записей совпадут с фильтром наличия.

Поэтому работы по созданию таких индексов превращаются в бесполезную трату процессорного времени, дискового пространства и оперативной памяти. Если Вы не знаете точно, как они будут использоваться, и не уверены, что фигурирующие в фильтрах наличия запросов атрибуты редко встречаются в целевых данных, лучше не создавать их.

Практически нет приложений, использующих фильтры наличия в своих поисковых запросах. Индексы наличия бессмысленны, если целевой атрибут существует в большинстве записей базы данных. Чаще всего в LDAP-системах не требуется делать индексы наличия, это просто бесполезная трата ресурсов.

Смотрите в подразделе Журналирование ниже, на что нужно обратить внимание, если в Вашем каталоге часто осуществляется поиск по неиндексированным атрибутам.

21.3. Журналирование

21.3.1. Какой уровень журналирования использовать

Уровень журналирования по умолчанию *loglevel stats* (256) — это действительно лучший выбор, поскольку, если проблемы начинают проявляться, не следует пытаться отследить их с помощью syslog. Вместо этого используйте флаги отладки и изучайте содержимое вывода потока stderr slapd. syslog слишком медленен для выполнения отладки, и, кроме того, он потенциально может терять данные, отбрасывая пришедшее сообщение, если не успевает его обработать.

Вопреки распространенному мнению, *loglevel 0* — не идеальный вариант в рабочих системах, поскольку Вы лишаете себя возможности отследить, когда проблема стала проявлять себя впервые.

21.3.2. На что обращать внимание

Самое распространённое сообщение, на которое нужно обратить внимание:

```
"<= bdb_equality_candidates: (foo) index_param failed (18)"
```

Это означает, что какое-то приложение пытается использовать фильтр соответствия (*foo=<некоторое значение>*), а атрибут *foo* не имеет индекса соответствия. Если таких сообщений много, нужно добавлять индекс. Если же оно появляется всего пару раз за месяц, можно его проигнорировать.

Уровень журналирования по умолчанию — stats (256). Если установлен этот уровень, в журнал помещаются основные параметры каждого запроса, обычно получается 1-3 строки вывода. В Solaris и системах, которые предоставляют только синхронное журналирование, Вы можете отключить его полностью, но обычно его оставляют включённым, чтобы можно было отследить сообщения об индексировании, как только они появятся. В Linux Вы можете настроить syslogd на асинхронное журналирование, в этом случае и производительность

возрастёт, и оперативность просмотра трафика syslog практически не пострадает.

21.3.3. Увеличение производительности

На некоторых системах Вы можете увеличить производительность журналирования, настроив syslog так, чтобы он не синхронизировал журналы из буфера в оперативной памяти с файловой системой при каждой записи (смотрите man-страницы *syslogd/syslog.conf*). В Linux Вы можете поставить "-" перед именем файла журнала в *syslog.conf*. Например, если Вы используете канал журналирования по умолчанию LOCAL4, попробуйте сделать так:

```
# LDAP logs
LOCAL4.*          -/var/log/ldap
```

Для системы syslog-ng добавьте или измените следующую строку в *syslog-ng.conf*:

```
options { sync(n); };
```

где n — количество строк, которые будут помещаться в буфер перед записью в файл.

21.4. Кэширование

Все мы знаем, что такое кэширование, правда?

Если коротко, "кэш — это блок памяти для временного хранения данных, которые могут быть повторно использованы" — http://en.wikipedia.org/wiki/Cache

Существует 3 типа кэшей: собственный кэш BerkeleyDB, кэш записей *slapd*(8) и кэш IDL.

21.4.1. Кэш Berkeley DB

Есть два подхода к настройке размера кэша BDB:

(а) размер кэша BDB, который требуется для загрузки базы данных через slapadd за оптимальное время;

(б) размер кэша BDB, который требуется для получения высокой производительности работы slapd после того, как данные были загружены.

Для подхода (а) оптимальный размер кэша — это размер всей базы данных. Если Ваша база данных уже загружена, получить его можно просто выполнив команду

```
du -c -h *.bdb
```

в директории, содержащей данные OpenLDAP (*/usr/local/var/openldap-data*).

Для подхода (б) оптимальный размер кэша — это просто размер файла *id2entry.bdb* плюс около 10% на прирост.

Необходимо выполнить настройку *DB_CONFIG* для каждого экземпляра базы данных типа BDB (back-bdb, back-hdb).

Обратите внимание, что в отличие от кэша BDB, представляющего собой просто необработанные куски оперативной памяти и конфигурируемого как размер памяти, кэш записей *slapd*(8) содержит записи, прошедшие разбор, и размер каждой записи может быть различным.

Существует также кэш IDL, используемый для временного хранения результатов поиска данных по индексу (Index Data Lookups). Если Вы сможете вместить всю базу данных в кэш записей slapd и все поиски по индексам в кэш IDL, это обеспечит Вам максимальную производительность.

Если же так сделать не получается, но есть возможность разместить базу данных целиком в кэш BDB, тогда лучше сделать это и уменьшить кэш записей slapd до необходимого размера.

В противном случае придётся искать баланс между кэшем BDB и кэшем записей.

Стоит отметить, что совершенно не требуется конфигурировать размер кэша BerkeleyDB эквивалентно размеру всей базы данных. Всё, что Вам действительно необходимо, — это кэш, достаточно большой, чтобы вместить Ваш "рабочий набор".

Это означает, что кэш должен быть настолько велик, чтобы содержать все те данные, к которым обращаются чаще всего, а также еще некоторое количество элементов, доступ к которым осуществляется не так часто.

Для получения дополнительной информации обращайтесь по адресу: http://www.oracle.com/technology/documentation/berkeley-db/db/ref/am_conf/cachesize.html

21.4.1.1. Расчёт размера кэша

База данных back-bdb размещается в двух основных файлах: `dn2id.bdb` и `id2entry.bdb`. Оба они являются базами данных в формате B-деревьев (B-tree). До этого мы никогда не описывали в документации внутреннее представление back-bdb, поскольку данный вопрос не представлялся ни тем, о чём следует беспокоиться каждому, ни тем, что следует запечатлеть на века. Однако, вот как обстоят дела в данный момент, в OpenLDAP 2.4.

B-дерево — это сбалансированное дерево (balanced tree); данные в нём хранятся в листовых узлах, а описание организации данных — во внутренних узлах (если Вы не знаете, что собой представляет древовидная структура данных, поищите информацию в Google, поскольку подробное описание выходит за рамки данной документации).

Для получения достойной производительности Вам потребуется достаточно кэш-памяти, чтобы разместить там все узлы по пути от корня дерева до конкретного нужного Вам элемента данных. Это будет достаточный кэш для единичной операции поиска. В общем случае, желательно иметь кэш, позволяющий разместить все внутренние узлы в базе данных.

```
db_stat -d
```

покажет Вам, сколько внутренних страниц находится в базе данных. Вы должны проверить это число как для dn2id, так и для id2entry.

Также имейте в виду, что *id2entry* всегда используют по 16KB на каждую "страницу", а *dn2id* используют столько, сколько по умолчанию использует та файловая система, на которой они размещаются, обычно 4 или 8KB. Во избежание переполнений, Ваш кэш должен быть как минимум настолько велик, чтобы вмещать общее количество внутренних страниц обоих баз данных *dn2id* и *id2entry*, плюс некоторое количество дополнительного пространства для размещения фактических данных из листовых страниц.

Например, я загрузил в мою тестовую базу данных OpenLDAP 2.4 LDIF-файл размером около 360MB. При этом механизм манипуляции данными back-hdb создал *dn2id.bdb* размером около 68MB, и *id2entry* размером около 800MB. db_stat показал мне, что *dn2id* использует страницы размером 4KB и содержит 433 внутренних и 6378 листовых страниц. id2entry использует страницы размером 16KB и содержит 52 внутренних и 45912 листовых страниц. Для того, чтобы эффективно извлечь какую-нибудь единичную запись из этой базы данных, кэш должен быть как минимум размером

```
(433+1) * 4KB + (52+1) * 16KB = 1736KB + 848KB =~ 2.5MB.
```

В расчёт не были приняты другие накладные расходы библиотеки, так что это даже меньше необходимого минимума. Если же ничего не было сконфигурировано, размер кэша по умолчанию — лишь 256KB.

В это число 2.5MB не входят также расходы на индексирование. Для каждого индексированного атрибута создаётся отдельный файл базы данных. В ранних версиях OpenLDAP эти индексные базы данных были в хэш-формате, но, начиная с OpenLDAP 2.2, для представления индексных баз данных также используется формат B-деревьев, поэтому при вычислении размера кэша для каждой индексной базы данных может применяться аналогичная процедура.

Например, если Вы проиндексировали только атрибут objectClass и db_stat показал, что *objectClass.bdb* использует 339 внутренних станицы размером 4KB, то дополнительный кэш, который потребуется для индекса данного атрибута, будет:

```
(339+1) * 4KB =~ 1.3MB.
```

Даже если включен только этот индекс, то, по расчётам, для этого механизма манипуляции данными нужен кэш размером как минимум 4MB. (Конечно, используя единственный общий кэш для всех файлов базы данных нельзя быть уверенным в том, что страницы кэша не будут использованы для чего-то такого, на что Вы не рассчитывали, однако подобное предположение даёт Вам некоторый шанс на успех).

С таким кэшем размером 4MB я могу получить слепок всей этой базы данных с помощью slapcat на моём 1.3GHz PIII за 1 минуту 40 секунд. Если удвоить размер кэша до 8MB, получение слепка займёт те же самые 1 минуту 40 секунд. Как только Вы выделили достаточно кэша, чтобы вместить внутренние страницы B-дерева, дальнейшее увеличение не даст никакого эффекта, пока размер кэша не станет настолько большим, чтобы вместить 100% страниц данных. У меня недостаточно свободной оперативной памяти, чтобы содержать в ней все 800MB данных id2entry, поэтому 4MB вполне достаточно.

Для механизмов back-bdb и back-hdb Вы можете использовать "db_stat -m" для проверки эффективности производительности кэша базы данных.

Подробнее о *db_stat*: http://www.oracle.com/technology/documentation/berkeley-db/db/utility/db_stat.html

21.4.2. Кэш записей *slapd*(8) (cachesize)

Кэш записей *slapd*(8) работает с декодированными записями. Смысл в том, что записи в кэше записей могут использоваться непосредственно, при этом достигается наиболее быстрый отклик. Если какой-либо записи нет в кэше записей, но её можно получить из кэша BDB, это позволит избежать операций ввода/вывода на диск, но всё равно потребует время на разбор, поэтому запрос такой записи будет выполняться медленнее.

Если этой записи нет и в кэше BDB, тогда BDB потребуется освободить часть своих текущих закэшированных страниц и получить необходимые страницы, в результате чего произойдёт несколько дорогостоящих операций ввода/вывода, а также разбора.

Конечно, самый оптимальный размер кэша таков, чтобы вместить полное количество записей в базе данных. Однако, большинство серверов службы каталогов не выдают ответов сразу по всей базе данных, так что достаточно установить размер кэша в меньшую величину, более точно соответствующую предполагаемому рабочему набору данных. Это второй из важнейших параметров для базы данных.

Что касается нахождения баланса между кэшем записей и кэшем BDB, то нужно учесть, что разобранные записи обычно занимают в памяти в два раза больше места, чем неразобранные на диске.

Как уже упоминалось, неверный выбор размера кэша базы данных приводит к снижению производительности. Этот признак не является показателем возникновения проблем с базой данных. Это просто факт того, что кэш не справляется с нагрузкой, что приводит к снижению производительности и увеличению времени отклика.

21.4.3. Кэш IDL (idlcachesize)

Каждый IDL хранит результаты поиска заданных запросов, поэтому кэш IDL в конечном итоге будет содержать результаты наиболее часто запрашиваемых поисков. Как правило, рекомендуемый размер этого кэша для back-bdb должен совпадать с "cachesize", а для back-hdb — 3x"cachesize".

Примечание: Настройка idlcachesize непосредственно влияет на эффективность поиска.

21.5. Потоки *slapd*(8)

slapd(8) может обрабатывать запросы с помощью настраиваемого числа потоков, которые, в свою очередь, влияют на скорость ввода/вывода в процессе соединения.

Это число зависит от количества "реальных" ядер в системе. Например, на сервере с двумя одноядерными процессорами следует установить 8 потоков, по 4 на каждое реальное ядро. Это максимальное значение для

операций "чтения". Установка большего количества потоков на одно ядро приведёт к тому, что *slapd*(8) будет медленнее отвечать на операции "чтения". С другой стороны, это позволит ускорить операции записи в системах, где операции записи преобладают над операциями чтения.

Расчётная верхняя граница количества потоков для хорошей производительности операций чтения — 16 (что и является значением по умолчанию).

К содержанию

22. Устранение неполадок

Если у Вас возникли проблемы при использовании OpenLDAP, подпишитесь на список рассылки программного обеспечения OpenLDAP, либо:

- Изучите архивы списков рассылки по адресу http://www.openldap.org/lists/#archives.
- Поищите ответ в FAQ по адресу http://www.openldap.org/faq/.
- Поищите ответ в Системе отслеживания проблемных вопросов по адресу http://www.openldap.org/its/.

Скорее всего, проблема уже была решена и подробно описана множество раз задолго до того, как Вы с ней столкнулись.

22.1. Чьи ошибки, пользователя или программного обеспечения?

Часто ошибки возникают из-за проблем с настройкой или непонимания того, что Вы пытаетесь внедрить и/или достичь.

Попытаемся обсудить часто возникающие ошибки пользователей.

22.2. Список тестов

Следующий список тестов может помочь отследить Вашу проблему. Воспользуйтесь им **перед** тем, как размещать своё сообщение в список рассылок или публиковать отчёт об ошибке.

1. **Воспользуйтесь инструментом** *slaptest* **для проверки конфигурации перед запуском** *slapd*.
2. **Убедитесь, что** *slapd* **ожидает подключения на указанном порту (портах) (как правило, 389 и 636), перед тем как пытаться выполнить запрос с помощью** *ldapsearch*.
3. **Вернул ли** *ldapsearch* **что-нибудь в ответ на Ваш запрос?**
4. **Если нет, - может быть Вы настроили комплексные ACL, не понимая до конца, как они работают?**
5. **Есть ли у Вас общесистемные настройки LDAP, указывающие на неверный каталог LDAP?**
6. **Используете ли Вы TLS?**
7. **Не истёк ли срок Вашего сертификата?**

22.3. Ошибки OpenLDAP

Иногда Вы можете столкнуться с реальной ошибкой OpenLDAP, в этом случае сообщите об этом в нашей Системе отслеживания проблемных вопросов по адресу http://www.openldap.org/its/. Однако, убедитесь, что Ваша ошибка не входит в число известных ошибок или часто возникающих проблем пользователей.

- ошибки в устаревших версиях OpenLDAP рассматриваться не будут;
- ошибки в релизах, которые больше не присутствуют в главной ветке Git (либо потому, что они уже были исправлены, либо этот функционал был исключён), рассматриваться не будут;
- ошибки в дистрибутивах программного обеспечения OpenLDAP, которые не относятся к программному обеспечению, предоставляемому OpenLDAP (скорее всего имеются ввиду патчи, дополнения и т.п.), рассматриваться не будут; в этих случаях обращайтесь к распространителям.

Примечание: Наша Система отслеживания проблемных вопросов **НЕ ПРЕДНАЗНАЧЕНА для осуществления поддержки** OpenLDAP, для этого, пожалуйста, подписывайтесь на наши списки рассылок: http://www.openldap.org/lists/.

Данные, которые необходимо предоставить в Вашем отчёте об ошибке, обсуждаются на нашем FAQ-O-MATIC по адресу http://www.openldap.org/faq/data/cache/59.html.

22.4. Ошибки программного обеспечения третьих сторон

Проект OpenLDAP осуществляет поддержку только программного обеспечения OpenLDAP.

Однако Вы можете воспользоваться коммерческой поддержкой (http://www.openldap.org/support/) или присоединиться к общему форуму LDAP для некоммерческих обсуждений и информации, относящейся к LDAP, по адресу: http://www.umich.edu/~dirsvcs/ldap/mailinglist.html.

22.5. Как связаться с проектом OpenLDAP

- Списки рассылки: http://www.openldap.org/lists/.
- Проект: http://www.openldap.org/project/.
- Отслеживание проблемных вопросов: http://www.openldap.org/its/.

22.6. Как представить Вашу проблему

22.7. Отладка *slapd*(8)

После прочтения всех предыдущих подразделов и перед тем, как отправить письмо в списки рассылки OpenLDAP, Вы можете попробовать некоторые из следующих действий, чтобы выяснить причину Вашей проблемы:

- Как правило, loglevel stats (256) - это хороший первичный уровень журналирования, чтобы попытаться получить полезную информацию для включения её в свой вопрос;
- Запуск *slapd -d -1* часто помогает выяснить простые вопросы, вроде пропущенного набора схемы или недостаточных прав на файл для пользователя, от которого запускается *slapd* (например, в случае использования сертификатов);
- Проверьте ошибки в Ваших журналах, как обсуждается в http://www.openldap.org/faq/data/cache/358.html.

22.8. Коммерческая поддержка

Перечисленные в http://www.openldap.org/support/ фирмы предлагают услуги технической поддержки сообществу OpenLDAP.

Нахождение той или иной фирмы в этом списке не должно рассматриваться как одобрение или рекомендация любого рода, и никоим образом не указывает на существование деловых отношений или принадлежности между любой из перечисленных фирм и OpenLDAP Foundation, проектом OpenLDAP или его участниками.

К содержанию

A. Изменения по сравнению с предыдущей версией

В следующих подразделах предпринята попытка обобщить новые возможности и изменения в программном обеспечении OpenLDAP по сравнению с версией 2.3.x и в Руководстве администратора OpenLDAP.

A.1. Новые разделы руководства

Чтобы Руководство администратора стало более глубоким и охватывало большинство вопросов, задаваемых в списках рассылки OpenLDAP и обсуждаемых там способов их решения, мы добавили следующие разделы:

- Для чего можно использовать LDAP?
- Для чего LDAP лучше не использовать?
- Непростые взаимоотношения LDAP и реляционных СУБД
- Контроль доступа
- Механизмы манипуляции данными
- Наложения
- Репликация
- Обслуживание
- Мониторинг
- Настройка производительности
- Устранение неполадок
- Изменения по сравнению с предыдущей версией
- Переход с версии 2.3.x
- Распространённые ошибки при использовании программного обеспечения OpenLDAP
- Рекомендуемые версии зависимостей программного обеспечения OpenLDAP
- Примеры развёртывания OpenLDAP в реальном мире
- Состав программного обеспечения OpenLDAP
- Примеры конфигурационных файлов
- Коды возврата LDAP
- Глоссарий

Кроме того, для облегчения навигации в содержании теперь представлены разделы 3-го уровня вложенности.

A.2. Новые функции и возможности в версии 2.4

A.2.1. Лучшая функциональность cn=config

Появилась новая man-страница slapd-config(5) для механизма манипуляции данными **cn=config**. В версии 2.3 была декларирована, но не реализована оригинальная конструкция, позволяющая, при добавлении или удалении конфигурационных записей с именами, которые могут быть упорядочены, автоматически переименовывать такие записи. Теперь эта возможность доступна в 2.4. Это означает, что если у Вас, к примеру, есть база данных

```
olcDatabase={1}bdb,cn=config
olcSuffix: dc=example,dc=com
```

и Вы хотите добавить новую подчинённую базу данных, теперь Вы можете добавить такую запись с помощью ldapadd:

```
olcDatabase={1}bdb,cn=config
olcSuffix: dc=foo,dc=example,dc=com
```

Новая база данных BDB будет добавлена в слот 1, а все последующие базы данных будут смещены на единицу; таким образом, оригинальная база данных BDB теперь будет называться:

```
olcDatabase={2}bdb,cn=config
olcSuffix: dc=example,dc=com
```

A.2.2. Лучшая функциональность cn=schema

В версии 2.3 Вы могли лишь добавлять новые элементы схемы, а удалять или изменять существующие элементы - нет. В 2.4 Вы можете изменять схему по своему желанию (конечно, за исключением жёстко закодированной системной схемы).

A.2.3. Более тонкая настройка Syncrepl

Оригинальная реализация Syncrepl в OpenLDAP 2.2 была предназначена для поддержки нескольких потребителей для одной базы данных, но эта возможность никогда не работала и была исключена в OpenLDAP 2.3; в этой версии Вы могли настроить только одного потребителя для каждой базы данных.

В 2.4 Вы можете настроить несколько потребителей для одной базы данных. Возможности конфигурации здесь достаточно комплексны и многочисленны. Вы можете настроить потребителей над произвольными поддеревьями базы данных (не пересекающимися, либо дублирующими друг друга). Любая часть базы данных, в свою очередь, может быть предоставлена другим потребителям с помощью наложения Syncprov. Наложение Syncprov работает с любым количеством потребителей для одной базы данных или для сколь угодно большого числа склеенных баз данных.

A.2.4. Разнонаправленная репликация с несколькими главными серверами

В результате работы по поддержке контекста нескольких потребителей система syncrepl теперь поддерживает разнонаправленную репликацию с несколькими главными серверами с разрешением конфликтов на уровне записи. Конечно, есть ряд важных ограничений: для того чтобы поддерживать целостность данных между всеми серверами, Вы должны обеспечить полную синхронизацию времени на всех серверах-участниках (например, использовать NTP на всех серверах).

Используемые для репликации entryCSN теперь сохраняют метки времени с разрешением в микросекунды, а не просто в секундах. Код delta-syncrepl ещё не был обновлён для использования в варианте с несколькими главными серверами, планируется осуществить это позже в ходе дальнейшей разработки версии 2.4.

A.2.5. Репликация конфигурации *slapd* (syncrepl и cn=config)

Syncrepl над cn=config был явно отключен в 2.3. В 2.4 он полностью поддерживается; Вы можете использовать syncrepl для репликации всей конфигурации сервера с одного сервера на сколь угодно большое количество других серверов. Можно клонировать всю конфигурацию работающего slapd, используя небольшое количество настроек (менее 10 строк), можно реплицировать только поддерево схемы данных, можно придумать и другие варианты. В тестах 049 и 050 из набора тестов есть рабочие примеры этих возможностей.

A.2.6. Репликация в режиме посылок

В версии 2.3 можно было настроить syncrepl для полноценной репликации в режиме посылок, используя его в сочетании с back-ldap, указывающим на целевой сервер. Но, поскольку база данных back-ldap должна иметь суффикс, соответствующий целевому суффиксу, можно было настроить только один экземпляр такой репликации на одном slapd.

В версии 2.4 Вы можете определить базу данных как "hidden", это означает, что при проверке конфликта имён её суффикс будет проигнорирован, и что база данных не будет использоваться для ответов на

поступающие извне запросы. Использование данной возможности скрытых ("hidden") баз данных позволяет настроить несколько баз данных с одним и тем же суффиксом, что в свою очередь позволяет настроить несколько экземпляров back-ldap для отправки репликации одной базы данных на несколько целевых серверов. Есть и другие способы применения скрытых баз данных (например, использование потребителя syncrepl для поддержки *локального* зеркала базы данных на отдельной файловой системе).

A.2.7. Более гибкое управление конфигурацией TLS

В версии 2.3 настройки TLS в slapd использовались только для входящих подключений к демону slapd. Для исходящих подключений, используемых, например, back-ldap или syncrepl, параметры TLS этих подключений брались из системного файла ldap.conf.

В версии 2.4 все подобные сессии наследуют свои параметры из основной конфигурации slapd, но есть возможность переопределить настройки индивидуально для каждого типа исходящего соединения. Это особенно полезно, если Вы используете аутентификацию на основе сертификатов и необходимо использовать разные клиентские сертификаты для разных направлений.

A.2.8. Улучшения производительности

Слишком много, чтобы всё перечислить. Некоторые заметные изменения - ldapadd когда-то был на два порядка медленнее, чем "slapadd -q". Теперь он в худшем случае в два раза медленнее slapadd -q. Некоторые сравнения всех релизов OpenLDAP 2.x доступны по адресу http://www.openldap.org/pub/hyc/scale2007.pdf.

В этом исследовании сравниваются версии 2.0.27, 2.1.30, 2.2.30, 2.3.33 и текущую из репозитория CVS. По диаграмме "Производительность кэшированных запросов" трудно увидеть разницу, поскольку время отклика очень мало, но новый код примерно на 25% быстрее чем 2.3, который примерно на 20% быстрее чем 2.2, который примерно на 100% быстрее чем 2.1, который примерно на 100% быстрее чем 2.0 в данном конкретном сценарии поиска. В этом тесте производился поиск по базе данных размером 1.3GB, содержащей 380836 записей (все в кэше записей slapd) менее чем за 1 секунду, таким образом, на CPU 2.4GHz с RAM DDR400 ECC/Registered мы можем производить поиск со скоростью более чем 500 тысяч записей в секунду. Поиск осуществлялся по неиндексированным атрибутам с использованием фильтра, не соответствующего ни одной записи, чтобы заставить slapd исследовать каждую запись в базе данных на соответствие фильтру.

По существу, кэш записей slapd в back-bdb/back-hdb настолько эффективен, что время на осуществление самого поиска практически незаметно; время отклика ограничивается только пропускной способностью памяти машины (скорость поиска данных соответствует примерно 3.5GB/сек; пропускная способность памяти машины составляет примерно 4GB/сек, что связано с задержками ECC и регистров).

A.2.9. Новые наложения

- slapo-constraint (Ограничения на значения атрибутов)
- slapo-dds (Динамические службы каталогов, RFC 2589)
- slapo-memberof (Обратное обслуживание членства в группах)

A.2.10. Новые возможности в существующих наложениях

- slapo-pcache
  - Инспектирование/Обслуживание
    - Доступ к базе данных кэша может быть получен напрямую через LDAP путём добавления специального элемента управления к каждому запросу LDAP; специальная расширенная операция позволяет последовательно удалять кэшированные записи и кэшированные запросы целиком
  - Горячая перезагрузка
    - Закэшированные запросы сохраняются на диск при остановке сервера, и вновь загружаются при следующем запуске, если срок их действия ещё не истёк
- slapo-rwm может безопасно взаимодействовать с другими наложениями
- Наложения Dyngroup/Dynlist объединененены, а также улучшена безопасность
  - добавлена поддержка dgIdentity (draft-haripriya-dynamicgroup)

A.2.11. Новые возможности в slapd

- мониторинг back-bdb и back-hdb: заполнение кэша, поиск по неиндексированным атрибутам;
- управление отслеживанием сессий (draft-wahl-ldap-session);
- удаление поддерева в back-sql (draft-armijo-ldap-treedelete);
- отсортированные значения в в атрибутах с несколькими значениями для более быстрого поиска соответствий;
- облегчённый диспетчер для повышения пропускной способности при высокой нагрузке и на многопроцессорных машинах (на 33% быстрее чем 2.3 на AMD quad-socket двухядерном сервере).

A.2.12. Новые возможности в libldap

- API клиента ldap_sync (LDAP Content Sync Operation, RFC 4533)

A.2.13. Новые клиенты, инструменты и усовершенствования существующих инструментов

- ldapexop для любых расширенных операций;
- Полная поддержка элементов управления в запросах/ответах для всех клиентов;
- Клиентские инструменты LDAP теперь соблюдают SRV-записи.

A.2.14. Новые опции сборки

- Поддерживается сборка с GnuTLS

A.3. Устаревшие возможности, удалённые из версии 2.4

Следующие возможности серьёзно устарели в 2.3 и были удалены в версии 2.4.

A.3.1. Slurpd

Чтобы ознакомиться с тем, почему этот демон больше не присутствует в OpenLDAP, прочитайте раздел Репликация.

A.3.2. back-ldbm

back-ldbm был и медленным, и ненадёжным одновременно. Его устаревший код индексирования был склонен к спонтанному разрушению, это было связано с основополагающими библиотеками баз данных (GDBM или NDBM), которые обычно использовались с данным механизмом манипуляции данными. back-bdb и back-hdb превосходят данный механизм во всех аспектах: упрощённое индексирование, позволяющее избежать разрушения индексов, высокоточная блокировка для улучшения параллельной работы, иерархическое кэширование для повышения производительности, модернизированный формат хранения данных на диске для повышения эффективности и переносимости, и полная поддержка транзакций для повышения надёжности.

К содержанию

B. Переход с версии 2.3.x

В следующих подразделах предпринята попытка документировать шаги, которые необходимо выполнить, чтобы осуществить переход с последних версий OpenLDAP 2.3.x.

Конечно же, перед тем, как производить эти шаги, следует выполнить нормальную процедуру обновления, обсуждаемую в разделе Обслуживание.

B.1. Атрибуты olc* cn=config

Некоторые атрибуты *olc** стали теперь устаревшими. Если в Ваших журналах встречаются записи, подобные той, что приведена ниже, просто удалите эти атрибуты из соответствующего файла ldif.

```
olcReplicationInterval: value #0: <olcReplicationInterval> keyword is obsolete (ignored)
```

B.2. ACL: чтобы выполнить поиск, требуются привилегии для базы поиска

Операции поиска теперь требуют привилегий "search" для псевдо-атрибута "entry" базы поиска. После перехода с 2.3.x убедитесь, что Ваши ACL предоставляют такой доступ для всех желаемых баз поиска.

Для примера предположим, что у Вас есть следующий ACL:

```
access to dn.sub="ou=people,dc=example,dc=com" by * search
```

Поисковые запросы, использующие в качестве базы "dc=example,dc=com", будут разрешены лишь в том случае, если Вы добавите следующий ACL:

```
access to dn.base="dc=example,dc=com" attrs=entry by * search
```

Примечание: man-страница *slapd.access*(5) утверждает, что данное требование выдвигалось уже в OpenLDAP 2.3. Однако, поведением по умолчанию это стало, начиная с версии 2.4.

Сюда будут добавлены другие подразделы.

К содержанию

C. Распространённые ошибки при использовании программного обеспечения OpenLDAP

В следующих подразделах предпринята попытка обобщить наиболее распространённые причины ошибок LDAP при использовании OpenLDAP.

C.1. Распространённые причины ошибок LDAP

C.1.1. ldap_*: Can't contact LDAP server (невозможно соединиться с сервером LDAP)

Ошибка **Can't contact LDAP server** обычно возвращается, когда к серверу LDAP невозможно подключиться. Такое может произойти по многим причинам:

- Сервер LDAP не запущен; это можно проверить, запустив, например,

  ```
  telnet <хост> <порт>
  ```

  заменяя *<хост>* и *<порт>* именем хоста и портом, на котором сервер должен принимать запросы.

- клиенту не сообщили, как соединиться с работающим сервером; при использовании инструментов командной строки OpenLDAP это достигается путём предоставления параметра -H, аргумент которого - корректный LDAP url, соответствующий интерфейсу, на котором сервер должен принимать запросы.

C.1.2. ldap_*: No such object (нет такого объекта)

Ошибка **no such object** обычно возвращается, когда невозможно найти целевой DN операции. В этом подразделе описаны причины, общие для всех типов операций. Необходимо также искать частные причины для конкретных типов операций (указанных в сообщении об ошибке).

Наиболее распространённой причиной этой ошибки является то, что объекта с таким именем не существует. Прежде всего, проверьте наличие опечаток.

Также имейте в виду, что по умолчанию вновь созданный сервер каталогов не содержит объектов (за исключением нескольких системных записей). Таким образом, если Вы настроили новый сервер каталогов и получили такое сообщение, то это может попросту означать, что Вы еще не добавили объект, который пытаетесь найти.

Обычно такая ошибка возникает, когда при запросе не указан базовый DN поиска, а значение по умолчанию не было корректно настроено.

Если, например, в slapd.conf Вы определили суффикс

```
suffix "dc=example,dc=com"
```

то Вам нужно использовать параметр `-b`

```
ldapsearch -b 'dc=example,dc=com' '(cn=jane*)'
```

чтобы указать утилите, где начинать поиск.

Параметр `-b` должен быть указан для всех команд LDAP, если Вы не настроили значение по умолчанию в *ldap.conf*(5).

Подробнее об этом смотрите в *ldapsearch*(1), *ldapmodify*(1).

Кроме того, *slapadd*(8) и её вспомогательные команды очень строго относятся к синтаксису файла LDIF.

Если в файле LDIF допущены некоторые вольности, то в результате его обработки может создаться видимость успешного создания базы данных, однако доступ к некотором частям этой базы может быть затруднён.

Одна из известнейших распространённых ошибок при создании базы данных - помещение пустой строки перед первой записью в файле LDIF. **В начале файла LDIF не должно быть пустых строк.**

При добавлении новых записей в каталог в общем случае рекомендуется использовать *ldapadd*(1) вместо *slapadd*(8). *slapadd*(8) нужно использовать для загрузки большого количества записей достоверно правильного формата.

Другой причиной появления этого сообщения может быть запись referral (смотрите Построение распределённой службы каталогов), ссылающаяся на незаполненный каталог.

В таком случае либо удалите эту запись referral, либо добавьте одну запись с базовым DN, на который ссылается referral, в пустой каталог.

Также данная ошибка может возникнуть, если slapd не может получить доступ к своим базам данных из-за проблем с правами на файлы. Например, в системе Red Hat Linux slapd запускается от имени пользователя 'ldap'. Если при создании базы данных с нуля slapadd был запущен от root, файлы в директории `/var/lib/ldap` будет созданы с владельцем и группой root и правами 600, делая тем самым содержимое каталога недоступным для сервера slapd.

C.1.3. ldap_*: Can't chase referral (невозможно последовать по ссылке)

Причиной этого является строка

```
referral        ldap://root.openldap.org
```

в `slapd.conf`. В оригинальном файле данная строка помещена как пример использования объектов referrals. Однако, если Ваша машина не подключена к Internet постоянно, она не сможет найти этот сервер, и, как следствие, выводит данное сообщение об ошибке.

Чтобы решить эту проблему, просто поместите # в начало данной строки и перезапустите slapd, либо направьте его на доступный сервер LDAP.

Смотрите также *ldapadd*(1), *ldapmodify*(1) и *slapd.conf*(5)

C.1.4. ldap_*: server is unwilling to perform (сервер не желает выполнять)

slapd вернёт ошибку unwilling to perform, если механизм манипуляции данными, содержащий целевую запись, не поддерживает запрошенную операцию.

Механизм манипуляции данными password может выполнять только поиски. Он будет возвращать ошибку unwilling to perform для всех остальных операций.

Механизм манипуляции данными shell является настраиваемым и может поддерживать ограниченное подмножество операций. Проверьте также другие ошибки, свидетельствующие о нехватке ресурсов, необходимых для сервера каталогов, например, у Вас может быть переполнен диск и т.п.

C.1.5. ldap_*: Insufficient access (недостаточные полномочия доступа)

Эта ошибка возникает, когда сервер запрещает операцию из-за недостаточных прав доступа. Обычно это происходит из-за подключения от имени DN, не обладающего достаточными привилегиями для выполнения операции (либо анонимного подключения).

Для получения полного доступа Вы можете подсоединиться от имени rootdn/rootpw, указанных в *slapd.conf*(5). В противном случае, Вы должны выполнять подключение от имени записи, которой предоставлены соответствующие права при настройке контроля доступа.

C.1.6. ldap_*: Invalid DN syntax (неверный синтаксис DN)

Целевой (или другой) DN операции является неверным. Это означает, что либо строковое представление DN не соответствует требуемой форме, один из типов в утверждениях значений атрибута не определён, либо одно из значений в утверждениях значений атрибута не соответствует требуемому синтаксису.

C.1.7. ldap_*: Referral hop limit exceeded (превышен лимит перехода по ссылкам)

Обычно эта ошибка возникает, когда клиент переходит по ссылке на сервер, который в свою очередь возвращает ему ссылку обратно на тот сервер, с которым он уже связывался. Этот сервер вновь отвечает также, как и в прошлый раз, и клиент зацикливается. Зацикливание обнаруживается, когда превышается лимит переходов.

Чаще всего причиной этого является неправильная настройка ссылки по умолчанию сервера. Ссылка по умолчанию не должна указывать на сам сервер, то есть на сервере ldap://myldap/ ссылка по умолчанию не должна указывать на ldap://myldap/ (или любое имя хоста/ip-адрес, эквивалентный myldap).

C.1.8. ldap_*: operations error (операционная ошибка)

В некоторых версиях *slapd*(8), ошибка *operationsError* выдавалась вместо других.

C.1.9. ldap_*: other error (другая ошибка)

В результате выполнения операции был возвращен нестандартный код неудачного завершения, свидетельствующий о внутренней ошибке. Возможно, проблему прояснит дополнительная информация, предоставляемая с кодом возврата, однако, скорее всего, придётся просматривать файлы журнала сервера.

C.1.10. ldap_add/modify: Invalid syntax (неверный синтаксис)

Сообщение об этой ошибке поступает, когда значение атрибута не соответствует ограничениям синтаксиса. Обычно в сопутствующей дополнительной информации даётся указание, какое значение и какого именно атрибута определено как неверное. Дважды проверьте это значение и другие значения, поскольку сервер выдаёт сообщение только о первой найденной им ошибке.

Типичные причины возникновения:

- посторонние пробелы (особенно конечные пробелы);
- символы в неправильной кодировке (LDAPv3 использует кодировку Unicode UTF-8);
- пустые значения (лишь немногие синтаксисы позволяют оставлять значения пустыми).

Для некоторых синтаксисов, таких как OBJECT IDENTIFIER (OID), данная ошибка свидетельствует о том, что предоставленный дескриптор ("короткое название") OID не распознан. Например, данная ошибка возвращается, если предоставленное значение *objectClass* не распознано.

C.1.11. ldap_add/modify: Object class violation (нарушение объектного класса)

Данная ошибка возвращается, когда добавляемая или изменяемая запись нарушает правила схемы объектного класса. Обычно в сопутствующей дополнительной информации данное нарушение детализируется. Ниже приведены некоторые информационные сообщения и описание причин их появления.

Нарушения, связанные с атрибутами записей:

```
Attribute not allowed (атрибут не разрешён)
```

Предоставленный атрибут не разрешён объектным классом (классами) записи.

```
Missing required attribute (отсутствует необходимый атрибут)
```

Атрибут, обязательный для объектного класса (классов) записи, не был предоставлен.

Нарушения, связанные с классом (классами) записи:

```
Entry has no objectClass attribute (у записи нет атрибута objectClass)
```

В записи не указано, к какому объектному классу она принадлежит.

```
Unrecognized objectClass (objectClass не распознан)
```

Одно (или несколько) из перечисленных значений атрибута objectClass не распознано.

```
No structural object class provided (не предоставлен структурный объектный класс)
```

Ни одно из перечисленных значений атрибута objectClass не является структурным объектным классом.

```
Invalid structural object class chain (неверная цепочка структурного объектного класса)
```

Два или более структурных объектных класса в значениях атрибута objectClass не принадлежат одной и той же цепочке структурного объектного класса.

```
Structural object class modification (изменение структурного объектного класса)
```

Попытка изменить структурный объектный класс записи в ходе операции модификации.

```
Instanstantiation of abstract objectClass (экземпляр абстрактного объектного класса).
```

Абстрактный класс не подчиняется ни одному из перечисленных структурных или вспомогательных классов.

```
Invalid structural object class (неверный структурный объектный класс)
```

Другая проблема со структурным объектным классом.

```
No structuralObjectClass operational attribute (нет операционного атрибута structuralObjectClass)
```

Данное сообщение обычно возвращается, когда теневой сервер предоставляет запись, не содержащую операционного атрибута structuralObjectClass.

Имейте в виду, что приведённые выше сообщения об ошибке и их описания предполагают наличие базовых знаний о схемах LDAP/X.500.

C.1.12. ldap_add: No such object (нет такого объекта)

Ошибка "ldap_add: No such object" обычно возвращается, когда не существует родительской записи для той записи, которую хотят добавить. Сначала добавьте родительскую запись...

Например, если при добавлении "cn=bob,dc=domain,dc=com" Вы получаете:

```
ldap_add: No such object
```

похоже, что запись "dc=domain,dc=com" не существует. Проверьте, существует ли она, с помощью ldapsearch:

```
ldapsearch -b 'dc=domain,dc=com' -s base '(objectclass=*)'
```

Если нет, добавьте её. Если нужна помощь, обратитесь к Руководству по быстрому развёртыванию и началу работы.

Примечание: если добавляемая запись совпадает с суффиксом базы данных, наличие родительской записи не требуется. Например, если Ваш суффикс - "dc=domain,dc=com", для добавления "dc=domain,dc=com" не требуется наличия "dc=com".

Также данная ошибка возникает, когда Вы пытаетесь добавить какую-либо запись, на хранение которой Ваш сервер не сконфигурирован.

Например, если суффикс Вашей базы данных - "dc=domain,dc=com" и Вы попытаетесь добавить "dc=domain2,dc=com", "dc=com", "dc=domain,dc=org", "o=domain,c=us", или какой либо другой DN в поддереве "dc=domain,dc=com" (не добавив предварительно саму эту запись), сервер вернёт ошибку "No such object" (либо отсылку).

slapd(8) обычно возвращает "no global superior knowledge" в качестве дополнительной информации, указывая, почему он вернул noSuchObject вместо отсылки. То есть в настройках сервера не было дано информации о глобальном вышестоящем сервере.

C.1.13. ldap add: invalid structural object class chain (неверная цепочка структурного объектного класса)

Данная ошибка относится к правилу о СТРУКТУРНЫХ объектных классах, которое утверждает, что объект может иметь только один СТРУКТУРНЫЙ класс, структурный класс данного объекта. Говорят, что объект принадлежит этому классу, нулю или более вспомогательным классам и их суперклассам.

Хотя все эти классы обычно перечислены в атрибуте objectClass записи, один из этих классов является структурным объектным классом записи. Таким образом, считается нормальным для атрибута objectClass иметь значения inetOrgPerson, organizationalPerson и person, поскольку они наследуются один от другого в форме одной цепочки суперкласса. То есть, inetOrgPerson наследуется от organizationPerson, который в свою очередь наследуется от person. С другой стороны, неверным будет перечислить в атрибуте objectClass сразу inetOrgPerson и account, поскольку inetOrgPerson и account не являются частью одной и той же цепочки суперкласса (если только вместе с ними не перечислен какой-то другой класс, являющийся подклассом от обоих).

Для решения этой проблемы необходимо определить, какой из классов будет лучше использовать в качестве структурного объектного класса для данной записи, добавить этот класс в атрибут objectClass (если он ещё не добавлен), и убрать из атрибута objectClass записи все остальные структурные классы, не являющиеся суперклассом для данного структурного объектного класса.

Какой объектный класс лучше использовать, зависит от конкретной ситуации. В общем случае, нужно проконсультироваться с документацией по тому приложению, которое использует данный объект из Вашего каталога. Скорее всего, это поможет сделать выбор.

C.1.14. ldap_add: no structuralObjectClass operational attribute (нет операционного атрибута structuralObjectClass)

ldapadd(1) может выдать ошибку:

```
adding new entry "uid=XXX,ou=People,o=campus,c=ru"
  ldap_add: Internal (implementation specific) error (80)
     additional info: no structuralObjectClass operational attribute
```

если slapd(8) не может определить на основании содержимого атрибута objectClass, какой должен быть структурный объектный класс.

C.1.15. ldap_add/modify/rename: Naming violation (нарушение именования)

OpenLDAP slapd проверяет целостность атрибутов именования и отличительных значений в соответствии с RFC 4512.

Атрибуты именования - это атрибуты тех типов, которые присутствуют в RDN записи; отличительные значения - это значения атрибутов именования, которые присутствуют в RDN записи, например, в записи

```
cn=Someone+mail=someone@example.com,dc=example,dc=com
```

атрибуты именования - cn и mail, а отличительные значения - Someone и someone@example.com.

OpenLDAP slapd проверяет целостность когда:

- добавляется запись;
- изменяется запись, если при этом изменяются значения атрибутов именования;
- переименовывается запись, если при этом изменяется RDN записи.

Возможные причины появления ошибки:

- атрибуты именования не присутствуют в записи, например:

```
dn: dc=example,dc=com
objectClass: organization
o: Example
# обратите внимание: "dc: example" пропущено
```

- атрибуты именования присутствуют в записи, но те типы атрибутов, которыми они определены, помечены как:
  - коллективные;
  - операционные;
  - устаревшие.
- атрибуты именования присутствуют в записи, а отличительные значения - нет, например:

```
dn: dc=example,dc=com
objectClass: domain
dc: foobar
# обратите внимание: "dc" присутствует, но его значение не "example"
```

- атрибуты именования присутствуют в записи вместе с отличительными значениями, но атрибуты именования:
  - не имеют поля равенства (equality field), вследствие чего невозможно утверждать равенство;
  - не поддерживают правило соответствия (возможно, пока);
  - не удовлетворяют правилу соответствия.
- предоставленные отличительные значения не соответствуют установленному для них синтаксису.
- другие ошибки, возникающие в процессе проверки/нормализации/сравнения; даже если Вам не подошел ни один из описанных выше вариантов, ситуация может проясниться при просмотре файлов журнала.

В любом случае, убедитесь, что определение типа атрибута для атрибута именования содержит соответствующее поле равенства (EQUALITY); либо же поле равенства содержит вышестоящий тип атрибута, если определение данного типа атрибута основано на attributeType (смотрите поле SUP). Подробности можно найти в RFC 4512.

C.1.16. ldap_add/delete/modify/rename: no global superior knowledge (нет сведений о глобальной вышестоящей части дерева)

Если имя целевой записи указывает, что она должна быть размещена в поддереве, на обслуживание которого не настроена ни одна из баз данных сервера, и, при этом, у сервера нет сведений о глобальной вышестоящей части дерева, сервер покажет, что он не желает выполнять операцию и выдаст "no global superior knowledge" в качестве дополнительного текста.

Возможно, допущена ошибка в имени записи, либо сервер должен хранить записи с таким именем, но был неверно настроен, либо, если каталог распределённый, не была настроена отсылка по умолчанию.

C.1.17. ldap_bind: Insufficient access (недостаточные полномочия доступа)

В текущих версиях slapd(8) перед тем, как допустить клиентов к выполнению операции подключения, требуется, чтобы у них были права на аутентификацию к тем типам атрибутов, которые используются в целях аутентификации. Поскольку все операции подключения выполняются анонимно (независимо от предыдущих успешных подключений), пользователю anonymous должны быть предоставлены права доступа auth.

В приведённом ниже примере ACL предоставляются следующие права доступа:

- анонимным пользователям:
  - права на прохождение аутентификации с использованием значений атрибута userPassword
- аутентифицированным пользователям:
  - права на обновление (но не чтение) своих собственных значений атрибута userPassword
  - права на чтение любого объекта, за исключением значений атрибута userPassword

Весь остальной доступ запрещён.

```
access to attr=userPassword
  by self =w
  by anonymous auth
access *
  by self write
  by users read
```

C.1.18. ldap_bind: Invalid credentials (неверные учётные данные)

Данная ошибка обычно возникает, когда предоставленные учётные данные (пароль) не совпадают с хранимыми в атрибуте userPassword той записи, от имени которой Вы пытаетесь подключиться.

Также данная ошибка может возникнуть, когда указанный при подключении DN неизвестен тому серверу, к которому Вы пытаетесь подключиться.

Проверьте и то, и другое! Кроме двух вышеуказанных причин, проверьте также, не запрещён ли доступ к атрибутам userPassword в выбранной части каталога. Фактически, slapd всегда возвращает "Invalid credentials" в случае неудачной попытки подключения, независимо от причины неудачи, поскольку другие коды возврата могут помочь злоумышленнику выяснить, правильное ли имя пользователя он предоставляет.

Для отладки правил доступа, определённых в slapd.conf, добавьте уровень журналирования "ACL".

C.1.19. ldap_bind: Protocol error (ошибка протокола)

Данная ошибка обычно возникает, когда запрашиваемая клиентом версия LDAP не поддерживается сервером.

Сервер OpenLDAP 2.x по умолчанию принимает запросы на подсоединение только LDAP версии 3, но может быть настроен для приёма запросов на подсоединение LDAP версии 2.

Примечание: если клиент запрашивает 3-ю версию протокола, сервер OpenLDAP 2.x ожидает, что будет использоваться LDAPv3 [RFC4510], если же запрашивается 2-я версия протокола, сервер ожидает, что будет использоваться ограниченный вариант LDAPv3 (в основном, синтаксис и семантика LDAPv3 в PDU LDAPv2).

Данный вариант также иногда называется LDAPv2+. Он отличается от U-Mich варианта LDAP в ряде направлений.

C.1.20. ldap_modify: cannot modify object class (не могу изменить объектный класс)

Данное сообщение обычно возвращается, когда предпринята попытка изменения атрибута objectClass способом, несовместимым с информационной моделью LDAP/X.500. На практике, такая ошибка обычно возникает, когда кто-то пытается изменить структуру объекта с одного класса на другой, например, пытается поменять 'яблоко' на 'грушу' или 'фрукт' на 'грушу'.

slapd(8) не допускает такие изменения в соответствии с ограничениями LDAP и X.500.

C.1.21. ldap_sasl_interactive_bind_s: ...

Если Вы хотели подсоединиться, используя DN и пароль, и получили ошибку из семейства ldap_sasl_interactive_bind_s, возможно, Вы просто забыли указать команде аргумент '-x'. По умолчанию используется SASL-аутентификация. Для выбора простой ("simple") аутентификации требуется аргумент '-x'.

C.1.22. ldap_sasl_interactive_bind_s: No such Object (нет такого объекта)

Данное сообщение указывает на то, что функция SASL-аутентификации LDAP не может прочитать Root DSE. Эта ошибка возникает, когда сервер не предоставляет root DSE. Такое может происходить из-за ограничений контроля доступа.

C.1.23. ldap_sasl_interactive_bind_s: No such attribute (нет такого атрибута)

Данное сообщение указывает на то, что функция SASL-аутентификации LDAP может прочитать Root DSE, но эта запись не содержит атрибута supportedSASLMechanism.

В атрибуте supportedSASLmechanism перечисляются механизмы, доступные в настоящее время. Данный список может быть пустым, если ни один из поддерживаемых механизмов в данный момент не доступен. Например, EXTERNAL перечисляется, только если клиент установил свою идентичность с помощью аутентификации на более низком уровне (например, TLS).

Примечание: данный атрибут может быть невидим из-за ограничений контроля доступа

Примечание: подсоединение с использованием SASL применяется по умолчанию для всех инструментов OpenLDAP, таких как ldapsearch(1), ldapmodify(1). Чтобы принудительно использовать подсоединение "simple", воспользуйтесь аргументом "-x". Использование подсоединения "simple" не рекомендовано, если не предприняты адекватные меры по защите конфиденциальности (такие, как TLS/SSL, IPSEC).

C.1.24. ldap_sasl_interactive_bind_s: Unknown authentication method (неизвестный метод аутентификации)

Данное сообщение указывает, что ни один из поддерживаемых сервером механизмов аутентификации SASL не поддерживается клиентом, либо они слишком слабы, либо по иной причине не подходят для использования клиентом. Имейте в виду, что параметры безопасности по умолчанию запрещают использование определённых механизмов, таких как ANONYMOUS и PLAIN (без TLS).

Примечание: подсоединение с использованием SASL применяется по умолчанию для всех инструментов

OpenLDAP, таких как ldapsearch(1), ldapmodify(1). Чтобы принудительно использовать подсоединение "simple", воспользуйтесь аргументом "-x". Использование подсоединения "simple" не рекомендовано, если не предприняты адекватные меры по защите конфиденциальности (такие, как TLS/SSL, IPSEC).

C.1.25. ldap_sasl_interactive_bind_s: Local error (82) (локальная ошибка)

Возможная причина возникновения этой ошибки - отсутствие прямой и обратной записи DNS для сервера LDAP.

C.1.26. ldap_search: Partial results and referral received (получены частичные результаты и отсылка)

Данная ошибка возвращается, если при ответе на поисковый запрос LDAPv2 сервер возвращает сразу и результаты (ноль или более совпавших записей), и ссылки (отсылки на другие серверы). Смотрите также ldapsearch(1).

Если при операции обновления реплики на ней не существует записи, которую требуется обновить (updatedn), будет возвращена отсылка. Также это может произойти, если ACL требует корректировки.

C.1.27. ldap_start_tls: Operations error (ошибка операций)

ldapsearch(1) и другие инструменты возвращают

```
ldap_start_tls: Operations error (1)
      additional info: TLS already started
```

когда пользователь (с помощью аргументов командной строки и/или файла ldap.conf(5)) запрашивает второй раз запустить TLS (SSL). Например, когда указываются сразу и "-H ldaps://server.do.main" и "-ZZ".

C.2. Другие ошибки

C.2.1. ber_get_next on fd X failed errno=34 (Numerical result out of range) (числовой результат за границей диапазона)

Данная ошибка slapd обычно указывает на то, что клиент отправил сообщение, превысившее административное ограничение. Смотрите директивы конфигурации sockbuf_max_incoming и sockbuf_max_incoming_auth в slapd.conf(5).

C.2.2. ber_get_next on fd X failed errno=11 (Resource temporarily unavailable) (ресурс временно недоступен)

Это сообщение не указывает на ненормальное поведение или ошибку. Оно просто означает, что ожидаемые данные еще не доступны из запрошенного ресурса, в запрошенном контексте, либо из запрошенного сетевого сокета. slapd(8) обработает данные, как только они станут доступны.

C.2.3. daemon: socket() failed errno=97 (Address family not supported) (семейство адресов не поддерживается)

Это сообщение указывает на то, что операционная система не поддерживает одно из семейств адресов (протоколов), на поддержку которого настроен slapd(8). Чаще всего это происходит, когда slapd(8) настраивался на поддержку IPv6, а ядро операционной системы - нет. В таких случаях данное сообщение можно проигнорировать.

C.2.4. GSSAPI: gss_acquire_cred: Miscellaneous failure; Permission denied; (различные сбои; доступ запрещён)

Это сообщение означает, что slapd был запущен не с правами root, и поэтому не может получить свой ключ Kerberos 5 из keytab, обычно файл /etc/krb5.keytab.

Файл keytab используется для хранения ключей, которые требуются сервисам и демонам, запускающимся во время загрузки. Очень важно хранить эти секретные сведения вне досягаемости злоумышленников.

Поэтому по умолчанию владельцем файла keytab является root, а для всех остальных чтение этого файла не разрешено. Не следует ослаблять эти ограничения на права. Вместо этого создайте другой файл keytab для slapd и убедитесь, что владельцем его является тот пользователь, с правами которого запускается slapd.

Чтобы это сделать, запустите kadmin и введите следующие команды:

```
addprinc -randkey ldap/ldap.example.com@EXAMPLE.COM
ktadd -k /etc/openldap/ldap.keytab ldap/ldap.example.com@EXAMPLE.COM
```

Затем, перейдя в оболочку, сделайте:

```
chown ldap:ldap /etc/openldap/ldap.keytab
chmod 600 /etc/openldap/ldap.keytab
```

Теперь нужно сообщить slapd (ну, на самом деле сообщить библиотеке gssapi Kerberos 5, вызываемой Cyrus SASL), где найти новый keytab. Сделайте это путём задания переменной окружения KRB5_KTNAME примерно так:

```
export KRB5_KTNAME="FILE:/etc/openldap/ldap.keytab"
```

Задайте эту переменную окружения в скрипте запуска slapd (на Red Hat лучше задать её в более подходящем месте /etc/sysconfig/ldap).

Всё это работает, только если Вы используете MIT kerberos. Это не будет работать, к примеру, с Heimdal.

В Heimdal есть функция gsskrb5_register_acceptor_identity(), устанавливающая путь к файлу keytab, который Вы хотите использовать. В Cyrus SASL 2 для использования данной функции Вы можете добавить

```
keytab: /path/to/file
```

в конфигурационный файл SASL Вашего приложения. Это работает только с Heimdal.

C.2.5. access from unknown denied (запрещён доступ от неизвестного)

Появление сообщения "access from unknown denied" в файле журнала связано с TCP wrappers. Дополнительную информацию смотрите в hosts_access(5). Чтобы избавиться от этой ошибки, Вы можете, например, добавить строку "slapd: .hosts.you.want.to.allow" в /etc/hosts.allow.

C.2.6. ldap_read: want=# error=Resource temporarily unavailable (ресурс временно недоступен)

Появление данного сообщения является нормальным. Это означает, что необходимые данные еще не доступны из запрошенного ресурса, либо из запрошенного сетевого сокета. slapd(8) обработает данные, как только они станут доступны.

C.2.7. `make test' fails (`make test' завершился неудачей)

Иногда `make test' завершается неудачей на самых ранних тестах с невразумительным сообщением вроде такого:

```
make test
make[1]: Entering directory `/ldap_files/openldap-2.4.6/tests'
make[2]: Entering directory `/ldap_files/openldap-2.4.6/tests'
Initiating LDAP tests for BDB...
Cleaning up test run directory leftover from previous run.
 Running ./scripts/all...
>>>>> Executing all LDAP tests for bdb
```

```
    >>>>> Starting test000-rootdse ...
    running defines.sh
    Starting slapd on TCP/IP port 9011...
    Using ldapsearch to retrieve the root DSE...
    Waiting 5 seconds for slapd to start...
    ./scripts/test000-rootdse: line 40: 10607 Segmentation fault $SLAPD -f $CONF1 -h $URI1 -d $LVL $TIMING
>$LOG1 2>&1
    Waiting 5 seconds for slapd to start...
    Waiting 5 seconds for slapd to start...
    Waiting 5 seconds for slapd to start...
    Waiting 5 seconds for slapd to start...
    Waiting 5 seconds for slapd to start...
    ./scripts/test000-rootdse: kill: (10607) - No such pid
    ldap_sasl_bind_s: Can't contact LDAP server (-1)
    >>>>> Test failed
    >>>>> ./scripts/test000-rootdse failed (exit 1)
    make[2]: *** [bdb-yes] Error 1
    make[2]: Leaving directory `/ldap_files/openldap-2.4.6/tests'
    make[1]: *** [test] Error 2
    make[1]: Leaving directory `/ldap_files/openldap-2.4.6/tests'
    make: *** [test] Error 2
```

Обычно, пять строк:

Waiting 5 seconds for slapd to start...

указывают на то, что slapd вообще не запустился.

В tests/testrun/slapd.1.log есть полный отчёт о том, что выдавал slapd во время попыток запуска. Уровень журналирования может быть повышен путём задания переменной окружения SLAPD_DEBUG в соответствующее значение; о том, что означают различные уровни журналирования, смотрите loglevel в slapd.conf(5).

Типичная причина такого поведения - проблема компоновки времени исполнения, то есть slapd не может найти некоторые динамические библиотеки, для работы с которыми он скомпонован. Попробуйте запустить ldd(1) на slapd (для тех архитектур, которые поддерживают компоновку времени исполнения).

Также могут быть и другие причины; содержимое файла журнала должно прояснить их.

Тесты, запускающие сразу несколько экземпляров slapd обычно помещают отчёты в tests/testrun/slapd.<n>.log, с различным <n> для каждого экземпляра slapd; такие тесты просматривают tests/testrun/ для назначения возможного значения <n>.

C.2.8. ldap_*: Internal (implementation specific) error (80) - additional info: entry index delete failed (внутренняя ошибка, зависящая от конкретной реализации, дополнительная информация: не удалось удалить индекс записи)

Возможно, это связано с неправильной принадлежностью директории BDB (/var/lib/ldap) и файлов в ней. Эти файлы должны принадлежать пользователю, с правами которого запускается slapd.

```
chown -R ldap:ldap /var/lib/ldap
```

устраняет данную ошибку в Debian.

C.2.9. ldap_sasl_interactive_bind_s: Can't contact LDAP server (-1) (не могу соединиться с сервером LDAP)

При использовании SASL, когда клиент соединяется с сервером LDAP, сервис slapd немедленно аварийно завершает работу и клиент получает сообщение об ошибке:

```
SASL/GSSAPI authentication started ldap_sasl_interactive_bind_s: Can't contact LDAP server (-1)
```

Если проверить сервис slapd, он окажется остановленным.

Это может произойти из-за использования различных несовместимых между собой версий BerkeleyDB при установке SASL и установке OpenLDAP. Проблема возникает при использовании нескольких версий BerkeleyDB. Решение: проверьте, какая версия BerkeleyDB использовалась при установке Cyrus SASL.

Переустановите OpenLDAP с нужной версией BerkeleyDB.

К содержанию

D. Рекомендуемые версии зависимостей программного обеспечения OpenLDAP

В данном приложении перечислены рекомендуемые версии программного обеспечения, от которого зависит OpenLDAP.

Пожалуйста, ознакомьтесь с подразделом Программное обеспечение, от которого зависит OpenLDAP, чтобы узнать больше о зависимостях от перечисленного ниже программного обеспечения.

D.1. Версии зависимостей

Таблица D.1: Версии зависимостей программного обеспечения OpenLDAP

Функция	**ПО**	**Версия**
Transport Layer Security:		
	OpenSSL	0.9.7+
	GnuTLS	2.0.1
	MozNSS	3.12.9
Simple Authentication and Security Layer	Cyrus SASL	2.1.21+
Сервис аутентификации Kerberos:		
	Heimdal	Version
	MIT Kerberos	Version
Программное обеспечение баз данных	Berkeley DB:	
		4.4
		4.5
		4.6
		4.7
		4.8
		5.0
		5.1
		Примечание: Настоятельно рекомендуется применить патчи от Oracle к указанным релизам.
Потоки:		
	POSIX *pthreads*	Version
	Mach *CThreads*	Version
TCP Wrappers	Name	Version

К содержанию

E. Примеры развёртывания OpenLDAP в реальном мире

Здесь будут примеры и их обсуждение

К содержанию

F. Состав программного обеспечения OpenLDAP

В следующих подразделах предпринята попытка обобщить различные составные части программного обеспечения OpenLDAP, расположенные в `openldap_src/contrib`

F.1. Клиентские API

Вступление и обсуждение будут позже.

F.1.1. ldapc++

Вступление и обсуждение будут позже.

F.1.2. ldaptcl

Вступление и обсуждение будут позже.

F.2. Наложения

F.2.1. acl

Плагины, реализующие правила доступа. В настоящее время только posixGroup, который реализует управление доступом, основанное на членстве в posixGroup.

F.2.2. addpartial

Преобразует запросы на добавление (Add) в запросы на изменение (Modify), если запись существует.

F.2.3. allop

Возвращает операционные атрибуты для корневого DSE, даже если они не были запрошены, поскольку некоторые клиенты ожидают этого.

F.2.4. autogroup

Автоматически обновляет членство в группах.

F.2.5. comp_match

Поддерживает правила соответствия компонентов (RFC 3687).

F.2.6. denyop

Запрещает указанные операции, возвращая *unwillingToPerform*.

F.2.7. dsaschema

Разрешает загрузку специфичной для DSA схемы, включая операционные атрибуты.

F.2.8. lastmod

Отслеживает время последней операции записи в базу данных.

F.2.9. nops

Удаляет нулевые операции, например, изменение значение на на такое же, что и было раньше.

F.2.10. nssov

Обрабатывает поисковые запросы NSS через локальный сокет домена Unix.

F.2.11. passwd

Осуществляет поддержку дополнительных парольных механизмов.

F.2.12. proxyOld

Поддерживает совместимость прокси-авторизации с устаревшим internet-draft.

F.2.13. smbk5pwd

Выполняет расширенную операцию PasswordModify, обновляя вместе с *userPassword* ключи Kerberos и хэши паролей Samba.

F.2.14. trace

Отслеживает вызовы наложений.

F.2.15. usn

Обслуживает атрибуты *usnCreated* и *usnChanged* по аналогии с Microsoft AD.

F.3. Инструменты

Вступление и обсуждение будут позже.

F.3.1. Журналирование статистики

statslog

F.4. Плагины SLAPI

Вступление и обсуждение будут позже.

F.4.1. addrdnvalues

Будет позже.

К содержанию

G. Примеры конфигурационных файлов

G.1. slapd.conf

G.2. ldap.conf

G.3. a-n-other.conf

К содержанию

H. Коды возврата LDAP

В рамках данного руководства мы включили стандартные коды возврата LDAP из *Приложения A. Коды возврата LDAP* RFC4511, копия которого есть в директории `doc/rfc` исходных кодов OpenLDAP.

Мы приводим развёрнутое описание каждой ошибки, применительно к работе набора инструментов OpenLDAP. Расширения LDAP могут возвращать специфичные для них коды возврата, не являющиеся частью RFC4511. Для расширений, реализованных в OpenLDAP, возвращаются соответствующие коды возврата. Их значения описаны в документации к соответствующему расширению.

H.1. Неошибочные коды возврата

Эти коды возврата (называемые "неошибочными") не указывают на возникновение ошибки:

```
success (0),
compareFalse (5),
compareTrue (6),
referral (10) и
saslBindInProgress (14).
```

Коды возврата *success*, *compareTrue* и *compareFalse* указывают на успешное завершение (и, соответственно, называются "успешными").

Коды возврата *referral* и *saslBindInProgress* указывают клиенту, что для завершения операции нужно произвести дополнительное действие.

H.2. Коды возврата

Далее приведены описания существующих кодов возврата LDAP.

H.3. success (0) - успех

Указывает на успешное завершение операции.

Примечание: этот код не используется при операциях сравнения (Compare). Смотрите compareFalse (5) и compareTrue (6).

H.4. operationsError (1) - ошибка операций

Указывает, что данная операция нарушает последовательность выполнения по отношению к другим операциям (того же или отличного типа).

Например, этот код возвращается, если клиент пытается выполнить StartTLS (раздел 4.14 RFC4511) в тот момент, когда выполнение других операций не закончено, либо когда слой TLS уже был установлен.

H.5. protocolError (2) - ошибка протокола

Указывает на получение сервером некорректно сформированных данных.

Применительно к операциям подсоединения (Bind), этот код также используется, чтобы указать, что сервер не поддерживает запрашиваемую версию протокола.

Применительно к расширенным (Extended) операциям, этот код также используется, чтобы указать, что сервер не поддерживает (по конструкции или конфигурации) расширенную операцию, ассоциированную с *requestName*.

Для операций запросов с указанием нескольких элементов управления данный код может использоваться, чтобы показать, что сервер не может проигнорировать порядок, в котором указаны элементы управления, либо была задана неверная или неопределённая комбинация элементов управления.

H.6. timeLimitExceeded (3) - превышение ограничения времени

Указывает, что заданное клиентом ограничение времени превышено до окончания операции.

H.7. sizeLimitExceeded (4) - превышение ограничения размера

Указывает, что заданное клиентом ограничение размера превышено до окончания операции.

H.8. compareFalse (5) - сравнение выявило ложь

Указывает, что операция сравнения (Compare) завершилась удачно и утверждение оказалось ложным (FALSE) или неопределённым (Undefined).

H.9. compareTrue (6) - сравнение выявило истину

Указывает, что операция сравнения (Compare) завершилась удачно и утверждение оказалось истинным (TRUE).

H.10. authMethodNotSupported (7) - метод аутентификации не поддерживается

Указывает на то, что данный метод или механизм аутентификации не поддерживается.

H.11. strongerAuthRequired (8) - требуется более строгая аутентификация

Указывает, что для завершения операции сервер требует строгой (более строгой) аутентификации.

При использовании операции с уведомлением о рассоединении (Notice of Disconnection), этот код указывает, что сервер обнаружил неожиданный обрыв или компрометацию установленного защищенного соединения между клиентом и сервером.

H.12. referral (10) - отсылка

Указывает, что для завершения операции требуется переход по отсылке (смотрите раздел 4.1.10 RFC4511).

H.13. adminLimitExceeded (11) - превышение административного ограничения

Указывает, что было превышено административное ограничение.

H.14. unavailableCriticalExtension (12) - недоступное критическое расширение

Указывает на то, что критический элемент управления не распознан (смотрите раздел 4.1.11 RFC4511).

H.15. confidentialityRequired (13) - требуется конфиденциальность

Указывает на то, что требуется защита конфиденциальности данных.

H.16. saslBindInProgress (14) - выполняется подсоединение SASL

Указывает, что для продолжения процесса аутентификации сервер требует посылки клиентом нового запроса на подсоединение, с тем же самым механизмом SASL (смотрите раздел 4.2 RFC4511).

H.17. noSuchAttribute (16) - нет такого атрибута

Указывает, что поименованная запись не содержит указанный атрибут или значение атрибута.

H.18. undefinedAttributeType (17) - неопределённый тип атрибута

Указывает, что запрашиваемое поле содержит описание нераспознанного атрибута.

H.19. inappropriateMatching (18) - неподходящее соответствие

Указывает, что была предпринята попытка (например, в утверждении) использовать правило соответствия, не определённое для представленного в операции типа атрибута.

H.20. constraintViolation (19) - нарушение ограничений

Указывает, что клиент предоставил значение атрибута, не удовлетворяющее ограничениям, наложенным на этот атрибут моделью данных.

Например, этот код возвращается, когда атрибуту, имеющему ограничение на присвоение одного значения (SINGLE-VALUE), присваивается несколько значений.

H.21. attributeOrValueExists (20) - атрибут или значение существует

Указывает, что клиент предоставил атрибут или значение для добавления к записи, но такой атрибут или значение уже существует.

H.22. invalidAttributeSyntax (21) - неверный синтаксис атрибута

Указывает, что предполагаемые значения атрибута не соответствуют синтаксису атрибута.

H.23. noSuchObject (32) - нет такого объекта

Указывает, что объект не существует в DIT.

H.24. aliasProblem (33) - проблема с псевдонимом

Указывает, что возникла проблема с псевдонимом. Например, код, который использовался для указания псевдонима, был разыменован, и полученное в результате имя не оказалось именем объекта.

H.25. invalidDNSyntax (34) - неверный синтаксис DN

Указывает, что запрошенное поле LDAPDN или RelativeLDAPDN (например, в базе поиска, целевой записи, операции ModifyDN newrdn, и т.д.) не соответствует требуемому синтаксису или содержит значения атрибутов, не соответствующие синтаксису данных типов атрибутов.

H.26. aliasDereferencingProblem (36) - проблема с разыменованием псевдонима

Указывает, что возникла проблема при разыменовании псевдонима. Обычно такое случается, когда псевдоним появляется там, где он не разрешен, или доступ к нему запрещён.

H.27. inappropriateAuthentication (48) - несоответствующая аутентификация

Указывает на то, что сервер требует от клиента предоставить учётные данные в какой-либо форме, когда тот пытается подключиться анонимно или без указания учётных данных.

H.28. invalidCredentials (49) - неверные учётные данные

Указывает, что предоставленные учётные данные (например, имя пользователя и пароль) являются неверными.

H.29. insufficientAccessRights (50) - недостаточные права доступа

Указывает, что клиент не имеет достаточных прав доступа для выполнения операции.

H.30. busy (51) - занят

Указывает, что сервер слишком занят чтобы обслужить операцию.

H.31. unavailable (52) - недоступен

Указывает, что сервер остановлен, либо подсистема, требуемая для завершения операции, недоступна.

H.32. unwillingToPerform (53) - не желают выполнять

Указывает, что сервер не желает выполнять операцию.

H.33. loopDetect (54) - обнаружено зацикливание

Указывает, что сервер обнаружил внутреннее зацикливание (например, при разыменовании псевдонима или при выполнении цепочки операций).

H.34. namingViolation (64) - нарушение именования

Указывает, что имя записи нарушает ограничения именования.

H.35. objectClassViolation (65) - нарушение объектного класса

Указывает, что запись нарушает ограничения объектного класса.

H.36. notAllowedOnNonLeaf (66) - не разрешено на нелистовой

записи

Указывает, что выполняемая операция является недопустимым действием на нелистовой записи.

H.37. notAllowedOnRDN (67) - не разрешено на RDN

Указывает, что выполняемая операция является недопустимой попыткой удаления значения, которое формирует относительное уникальное имя записи.

H.38. entryAlreadyExists (68) - запись уже существует

Указывает, что запрос (на добавление, перемещение, переименование) не может быть выполнен, поскольку целевая запись уже существует.

H.39. objectClassModsProhibited (69) - изменение объектного класса запрещено

Указывает, что попытка изменить объектный класс (классы) в атрибуте 'objectClass' записи запрещена.

Например, данный код возвращается, когда клиент пытается изменить структурный объектный класс записи.

H.40. affectsMultipleDSAs (71) - оказывается влияние на несколько DSA

Указывает, что выполняемая операция не может быть разрешена, поскольку это повлияет на несколько серверов (DSA).

H.41. other (80) - другое

Указывает, что на сервере произошла внутренняя ошибка.

К содержанию

I. Глоссарий

I.1. Термины

Термин	Определение	Перевод и пояснения
3DES	Triple DES	Симметричный блочный шифр Triple DES
ABNF	Augmented Backus-Naur Form	Расширенная спецификация синтаксиса Бэкуса-Наура
ACDF	Access Control Decision Function	Функция принятия решения в системе контроля доступа
	ASCII Compatible	

ACE	Encoding	ASCII-совместимая кодировка
ASCII	American Standard Code for Information Interchange	Американская стандартная кодировочная таблица для печатных символов и некоторых специальных кодов
ACID	Atomicity, Consistency, Isolation, and Durability	Атомарность, Согласованность, Изолированность, Долговечность — требования к транзакционной системе, в т.ч. СУБД
ACI	Access Control Information	Информация контроля доступа — вся совокупность информации, на основании которой принимается решение о предоставлении доступа
ACL	Access Control List	Список контроля доступа
AES	Advance Encryption Standard	Симметричный алгоритм блочного шифрования AES, принятый в качестве стандарта шифрования правительством США в 2002 году
ABI	Application Binary Interface	Двоичный интерфейс приложений
API	Application Program Interface	Интерфейс программирования приложений (интерфейс прикладного программирования)
ASN.1	Abstract Syntax Notation — One	Язык для описания абстрактного синтаксиса данных (ASN.1), используемый OSI
AVA	Attribute Value Assertion	Утверждение значения атрибута — комбинация описания атрибута и значения атрибута
AuthcDN	Authentication DN	Аутентификационный DN
AuthcId	Authentication Identity	Аутентификационная идентификационная сущность
AuthzDN	Authorization DN	Авторизационный DN
AuthzId	Authorization Identity	Авторизационная идентификационная сущность
BCP	Best Current Practice	Лучший современный опыт (статус RFC)
BDB	Berkeley DB (Backend)	Механизм манипуляции данными BDB (Berkley DB)
BER	Basic Encoding Rules	Основные правила кодирования — набор правил стандарта ASN.1 для кодирования абстрактной информации о структурах данных в конкретный поток данных
BNF	Backus-Naur Form	Спецификация синтаксиса Бэкуса-Наура
C	The C Programming Language	Язык программирования C
CA	Certificate Authority	Центр сертификации или Удостоверяющий центр (термин TLS)
CER	Canonical Encoding Rules	Канонические правила кодирования — подмножество BER
CLDAP	Connection-less LDAP	LDAP без установки соединения (RFC 1798), часть стандарта X.500, доступ к службам каталогов по UDP и другим протоколам без установки соединения
CN	Common Name	Общепринятое имя (термин служб каталогов)
CRAM-MD5	SASL MD5 Challenge/Response Authentication Mechanism	Механизм аутентификации SASL MD5 Challenge/Response
CRL	Certificate Revocation List	Список отозванных сертификатов
DAP	Directory Access Protocol	Протокол доступа к каталогу, часть стандарта X.500

DC	Domain Component	Доменный компонент (термин служб каталогов)
DER	Distinguished Encoding Rules	Уникальные правила кодирования — подмножество BER
DES	Data Encryption Standard	Симметричный алгоритм шифрования DES, разработанный фирмой IBM и утвержденный правительством США в 1977 году как официальный стандарт
DIB	Directory Information Base	Информационная база каталога
DIGEST-MD5	SASL Digest MD5 Authentication Mechanism	Механизм аутентификации SASL Digest MD5
DISP	Directory Information Shadowing Protocol	Протокол резервирования информации каталога, часть стандарта X.500
DIT	Directory Information Tree	Информационное дерево каталога (термин служб каталогов)
DNS	Domain Name System	Система доменных имён
DN	Distinguished Name	Уникальное имя (термин служб каталогов)
DOP	Directory Operational Binding Management Protocol	Протокол управления операционным соединением каталога, часть стандарта X.500
DSAIT	DSA Information Tree	Информационное дерево DSA
DSA	Directory System Agent	Системный агент каталога (термин X.500)
DSE	DSA-specific Entry	Запись, специфичная для DSA
DSP	Directory System Protocol	Системный протокол каталога, часть стандарта X.500
DS	Draft Standard	Статус, получаемый проектным стандартом Internet-Draft на одном из этапов стандартизации
DUA	Directory User Agent	Пользовательский агент каталога (термин X.500)
EXTERNAL	SASL External Authentication Mechanism	Механизм аутентификации SASL External
FAQ	Frequently Asked Questions	Часто задаваемые вопросы
FTP	File Transfer Protocol	Протокол передачи файлов
FYI	For Your Information	К Вашему сведению (статус RFC)
GSER	Generic String Encoding Rules	Общие правила кодирования строк — набор правил ASN.1, разработанный для LDAP
GSS-API	Generic Security Service Application Program Interface	Общий интерфейс программирования приложений службы безопасности
GSSAPI	SASL Kerberos V GSS-API Authentication Mechanism	Механизм аутентификации SASL Kerberos V GSS-API
HDB	Hierarchical Database (Backend)	Механизм манипуляции данными HDB (Иерархическая база данных)

I-D	Internet-Draft	Проектный стандарт, не получивший (возможно, пока) официального статуса RFC
IA5	International Alphabet 5	Интернациональный алфавит 5, то же самое, что и ASCII
IDNA	Internationalized Domain Names in Applications	Интернационализованные доменные имена в приложениях — механизм преобразования доменных имён содержащих символы, не входящие в ASCII
IDN	Internationalized Domain Name	Интернационализованные доменные имена — доменные имена, которые содержат символы национальных алфавитов
ID	Identifier	Идентификатор
IDL	Index Data Lookups	Поиск данных по индексу
IP	Internet Protocol	Протокол Интернет
IPC	Inter-process communication	Межпроцессное взаимодействие — набор способов обмена данными между множеством потоков в одном или более процессах
IPsec	Internet Protocol Security	IPSec — набор протоколов защиты сетевого трафика на IP-уровне
IPv4	Internet Protocol, version 4	Протокол интернет версии 4
IPv6	Internet Protocol, version 6	Протокол интернет версии 6
ITS	Issue Tracking System	Система отслеживания проблемных вопросов
JPEG	Joint Photographic Experts Group	Объединённая группа экспертов по фотографии — совместная рабочая группа (комитет), образованная несколькими международными организациями, разработала стандарт JPEG и другие
Kerberos	Kerberos Authentication Service	Служба аутентификации Kerberos
LBER	Lightweight BER	Облегчённый BER
LDAP	Lightweight Directory Access Protocol	Облегчённый протокол доступа к службам каталогов
LDAP Sync	LDAP Content Synchronization	Синхронизация содержимого LDAP
LDAPv3	LDAP, version 3	Протокол LDAP версии 3
LDIF	LDAP Data Interchange Format	Формат обмена данными LDAP
LMDB	Lightning Memory-Mapped Database	Механизм манипуляции данными LMDB (высокоскоростная отображаемая в памяти база данных)
MD5	Message Digest 5	128-битный алгоритм хеширования, разработанный профессором Рональдом Л. Ривестом в 1991 году
MIB	Management Information Base	База управляющей информации, используемая в процессе управления сетью в качестве модели управляемого объекта в архитектуре агент-менеджер. В частности используется протоколом SNMP
MODDN	Modify DN	Операция изменения DN
MODRDN	Modify RDN	Операция изменения RDN
NSSR	Non-specific Subordinate Reference	Неконкретная ссылка подчинённости — термин X.500
OID	Object Identifier	Идентификатор объекта
OSI	Open Systems	Взаимодействие открытых систем — проект по стандартизации протоколов сетевого взаимодействия, в результате которого была разработана эталонная

	Interconnect	сетевая модель OSI
OTP	One Time Password	Одноразовый пароль
PDU	Protocol Data Unit	Блок данных протокола — блок специфичных для протокола определённого уровня данных, содержащий управляющую информацию и, возможно, пользовательские данные протокола этого уровня
PEM	Privacy Enhanced eMail	Криптографический алгоритм, изначально разрабатывавшийся для безопасности электронной почты, но широкого распространения в этом направлении не получил. Используется в системах PKI как формат хранения сертификатов CA
PEN	Private Enterprise Number	Номер частного предприятия — глобально-уникальный OID, выдаваемый по запросу организации из пространства Private Enterprise, поддерживаемого IANA
PKCS	Public Key Cryptosystem	Криптографическая система с открытым ключом
PKI	Public Key Infrastructure	Инфраструктура открытых ключей — технология аутентификации с помощью открытых ключей
PKIX	Public Key Infrastructure (X.509)	X.509 — стандарт, определяющий форматы данных и процедуры распределения открытых ключей с помощью сертификатов с цифровыми подписями, которые предоставляются сертификационными органами (CA)
PLAIN	SASL Plaintext Password Authentication Mechanism	Механизм аутентификации SASL Plaintext Password (Пароли в открытом виде)
POSIX	Portable Operating System Interface	Переносимый интерфейс операционных систем Unix — набор стандартов, описывающих интерфейсы между операционной системой и прикладной программой
PS	Proposed Standard	Предложенный стандарт (статус RFC)
RDN	Relative Distinguished Name	Относительное уникальное имя (термин служб каталогов)
RFC	Request for Comments	Запрос комментариев — документ из серии пронумерованных информационных документов Интернета, содержащих технические спецификации и стандарты, широко применяемые во всемирной сети
RPC	Remote Procedure Call	Вызов удалённой процедуры
RXER	Robust XML Encoding Rules	Однозначные правила кодирования XML (RFC4910)
SASL	Simple Authentication and Security Layer	Простой уровень аутентификации и безопасности — это фреймворк для предоставления аутентификации и защиты данных в протоколах на основе соединений
SDF	Simple Document Format	Простой формат документов — один из языков разметки текста для его дальнейшей переконвертации в тот или иной формат документа
SDSE	Shadowed DSE	Теневой DSE
SHA1	Secure Hash Algorithm 1	Алгоритм криптографического хэширования SHA1
SLAPD	Standalone LDAP Daemon	Автономный демон LDAP
SLURPD	Standalone LDAP Update Replication Daemon	Автономный демон обновления и репликации LDAP
SMTP	Simple Mail Transfer Protocol	Простой протокол передачи почты — это сетевой протокол, предназначенный для передачи электронной почты в сетях TCP/IP
SNMP	Simple Network Management Protocol	Простой протокол управления сетями — это протокол управления сетями связи на основе архитектуры UDP

SQL	Structured Query Language	Структурированный язык запросов — унифицированный язык для работы с базами данных
SRP	Secure Remote Password	Безопасные удалённые пароли. На этой технологии основан одноимённый протокол SRPP — протокол парольной аутентификации, устойчивый к прослушиванию и MITM-атаке и не требующий третей доверенной стороны
SSF	Security Strength Factor	Фактор силы безопасности
SSL	Secure Socket Layer	Уровень защищённых сокетов — криптографический протокол, который обеспечивает установление безопасного соединения между клиентом и сервером
STD	Internet Standard	Стандарт Интернета (статус RFC)
TCP	Transmission Control Protocol	Протокол управления передачей
TLS	Transport Layer Security	Безопасность транспортного уровня
UCS	Universal Multiple-Octet Coded Character Set	Универсальный набор символов — стандарт кодировки символов, на котором базируются многие кодировки символов
UDP	User Datagram Protocol	Протокол пользовательских дейтаграмм
UID	User Identifier	Идентификатор пользователя (термин служб каталогов)
Unicode	The Unicode Standard	Стандарт Unicode
UNIX	Unix	Unix (операционная система)
URI	Uniform Resource Identifier	Унифицированный (единообразный) идентификатор ресурса
URL	Uniform Resource Locator	Единообразный локатор (определитель местонахождения) ресурса
URN	Uniform Resource Name	Единообразное название (имя) ресурса
UTF-8	8-bit UCS/Unicode Transformation Format	Формат преобразования Юникода, совместимый с 8-битным кодированием текста
UTR	Unicode Technical Report	Технический отчёт Unicode — документация по различным аспектам Unicode
UUID	Universally Unique Identifier	Универсальный уникальный идентификатор — это стандарт идентификации, используемый в создании программного обеспечения, стандартизированный Open Software Foundation как часть Распредёленного компьютерного окружения
WWW	World Wide Web	Всемирная паутина — распределенная система, предоставляющая доступ к связанным между собой документам, расположенным на различных компьютерах, подключенных к Интернету
X.500	X.500 Directory Services	Службы каталогов X.500
X.509	X.509 Public Key and Attribute Certificate Frameworks	X.509 — стандарт, определяющий форматы данных и процедуры распределения открытых ключей с помощью сертификатов с цифровыми подписями, которые предоставляются сертификационными органами (CA)
XED	XML Enabled Directory	Каталог с включенным XML — фреймворк для управления объектами, представленными с использованием XML. XED строится на технологиях служб каталогов X.500 и LDAP.
XER	XML Encoding Rules	Правила кодирования XML — подмножество правил кодирования ASN.1 для представления структур данных XML
XML	Extensible Markup	Расширяемый язык разметки

	Language	
syncrepl	LDAP Sync-based Replication	Репликация, основанная на LDAP Sync

I.2. Связанные организации

Сокращение	Полное наименование	Перевод наименования	Ссылка
ANSI	American National Standards Institute	Американский национальный институт стандартов	http://www.ansi.org/
BSI	British Standards Institute	Британский институт стандартов	http://www.bsi-global.com/
COSINE	Co-operation and Open Systems Interconnection in Europe	Проект Сотрудничество и взаимодействие открытых систем в Европе	
CPAN	Comprehensive Perl Archive Network	Всеобъемлющая сеть архивов Perl	http://cpan.org/
Cyrus	Project Cyrus	Проект Cyrus	http://cyrusimap.web.cmu.edu/
FSF	Free Software Foundation	Фонд свободного программного обеспечения	http://www.fsf.org/
GNU	GNU Not Unix Project	Проект GNU Not Unix	http://www.gnu.org/
IAB	Internet Architecture Board	Совет по архитектуре Интернета	http://www.iab.org/
IANA	Internet Assigned Numbers Authority	Администрация адресного пространства Интернет	http://www.iana.org/
IEEE	Institute of Electrical and Electronics Engineers	Институт инженеров электротехники и электроники	http://www.ieee.org
IESG	Internet Engineering Steering Group	Группа по выработке инженерного регламента Интернета	http://www.ietf.org/iesg/
IETF	Internet Engineering Task Force	Инженерный совет Интернет	http://www.ietf.org/
IRTF	Internet Research Task Force		http://www.irtf.org/
ISO	International Standards Organisation	Исследовательская группа Интернет-технологий	http://www.iso.org/
ISOC	Internet Society	Общество Интернета	http://www.isoc.org/
ITU	International Telephone Union	Международный союз электросвязи	http://www.itu.int/
OLF	OpenLDAP Foundation	Фонд OpenLDAP	http://www.openldap.org/foundation/
OLP	OpenLDAP Project	Проект OpenLDAP	http://www.openldap.org/project/
OpenSSL	OpenSSL Project	Проект OpenSSL	http://www.openssl.org/
RFC Editor	RFC Editor	База данных RFC	http://www.rfc-editor.org/
Oracle	Oracle Corporation	Корпорация Oracle	http://www.oracle.com/
UM	University of Michigan	Мичиганский Университет	http://www.umich.edu/
UMLDAP	University of Michigan LDAP Team	Команда LDAP Мичиганского Университета	http://www.umich.edu/~dirsvcs/ldap/ldap.html

I.3. Связанные продукты

Название	Ссылка
SDF	http://search.cpan.org/src/IANC/sdf-2.001/doc/catalog.html
Berkeley DB	http://www.oracle.com/database/berkeley-db/db/index.html
Cyrus	http://cyrusimap.web.cmu.edu/generalinfo.html
Cyrus SASL	http://asg.web.cmu.edu/sasl/sasl-library.html
Git	http://git-scm.com/
GNU	http://www.gnu.org/software/
GnuTLS	http://www.gnu.org/software/gnutls/
Heimdal	http://www.pdc.kth.se/heimdal/
JLDAP	http://www.openldap.org/jldap/
MIT Kerberos	http://web.mit.edu/kerberos/www/
MozNSS	http://developer.mozilla.org/en/NSS
OpenLDAP	http://www.openldap.org/
OpenLDAP FAQ	http://www.openldap.org/faq/
OpenLDAP ITS	http://www.openldap.org/its/
OpenLDAP Software	http://www.openldap.org/software/
OpenSSL	http://www.openssl.org/
Perl	http://www.perl.org/
UMLDAP	http://www.umich.edu/~dirsvcs/ldap/ldap.html

I.4. Документация

Документ	Название документа	Статус	Ссылка
UM-GUIDE	Руководство администратора SLAPD и SLURPD	O	http://www.umich.edu/~dirsvcs/ldap/doc/guides/slapd/guide.pdf
RFC2079	Определение типов атрибутов и объектных классов X.500 для хранения унифицированных идентификаторов ресурсов	PS	http://tools.ietf.org/html/rfc2079
RFC2296	Использование языковых кодов в LDAP	PS	http://tools.ietf.org/html/rfc2296
RFC2307	Подход к использованию LDAP в качестве Сетевой Информационной Службы	X	http://tools.ietf.org/html/rfc2307
RFC2589	Lightweight Directory Access Protocol (v3): Расширения динамических служб каталогов	PS	http://tools.ietf.org/html/rfc2589
RFC2798	Определение объектного класса LDAP inetOrgPerson	I	http://tools.ietf.org/html/rfc2798
RFC2831	Использование аутентификации Digest в качестве механизма SASL	PS	http://tools.ietf.org/html/rfc2831

RFC2849(рус.) RFC2849(ориг.)	Формат обмена данными LDAP (LDIF)	PS	http://tools.ietf.org/html/rfc2849
RFC3088	Корневой сервис OpenLDAP	X	http://tools.ietf.org/html/rfc3088
RFC3296	Именованные ссылки подчинённости в LDAP	PS	http://tools.ietf.org/html/rfc3296
RFC3384	Требования репликации Lightweight Directory Access Protocol (version 3)	I	http://tools.ietf.org/html/rfc3384
RFC3494	Присвоение Lightweight Directory Access Protocol version 2 (LDAPv2) статуса исторического протокола	I	http://tools.ietf.org/html/rfc3494
RFC4013	SASLprep: Профиль строкового представления для имён пользователей и паролей	PS	http://tools.ietf.org/html/rfc4013
RFC4346	The Transport Layer Security (TLS) Protocol, Version 1.1	PS	http://tools.ietf.org/html/rfc4346
RFC4422	Simple Authentication and Security Layer (SASL)	PS	http://tools.ietf.org/html/rfc4422
RFC4510	Lightweight Directory Access Protocol (LDAP): Техническая спецификация	PS	http://tools.ietf.org/html/rfc4510
RFC4511(рус.) RFC4511(ориг.)	Lightweight Directory Access Protocol (LDAP): Определение протокола	PS	http://tools.ietf.org/html/rfc4511
RFC4512	Lightweight Directory Access Protocol (LDAP): Информационные модели каталога	PS	http://tools.ietf.org/html/rfc4512
RFC4513	Lightweight Directory Access Protocol (LDAP): Методы аутентификации и механизмы безопасности	PS	http://tools.ietf.org/html/rfc4513
RFC4514	Lightweight Directory Access Protocol (LDAP): Строковое представление уникальных имён	PS	http://tools.ietf.org/html/rfc4514
RFC4515	Lightweight Directory Access Protocol (LDAP): Строковое представление поисковых фильтров	PS	http://tools.ietf.org/html/rfc4515
RFC4516	Lightweight Directory Access Protocol (LDAP): Uniform Resource Locator	PS	http://tools.ietf.org/html/rfc4516
RFC4517	Lightweight Directory Access Protocol (LDAP): Синтаксисы и правила соответствия	PS	http://tools.ietf.org/html/rfc4517
RFC4518	Lightweight Directory Access Protocol (LDAP): Подготовка интернационализированных строк	PS	http://tools.ietf.org/html/rfc4518
	Lightweight Directory Access Protocol (LDAP): Набор		

RFC4519	схемы для пользовательских приложений	PS	http://tools.ietf.org/html/rfc4519
RFC4520	Соглашения IANA относительно LDAP	BCP	http://tools.ietf.org/html/rfc4520
RFC4533(рус.) RFC4533(ориг.)	The Lightweight Directory Access Protocol (LDAP): Операция синхронизации содержимого	X	http://tools.ietf.org/html/rfc4533
Chu-LDAPI	Использование LDAP поверх механизмов IPC	ID	http://tools.ietf.org/html/draft-chu-ldap-ldapi-00

К содержанию

J. Общие инструкции configure

```
Базовая установка
==================

 Общие инструкции по установке.

   Скрипт `configure' пытается угадать правильные значения различных системно-зависимых  переменных,
используемых в ходе компиляции.  Он использует эти значения для создания файлов `Makefile' в каждой
директории пакета. Он также создаёт один или несколько файлов  `.h',  содержащих системно-зависимые
определения. Наконец, он создаёт скрипт  `config.status', который можно запустить при необходимости
для пересоздания  текущей  конфигурации,  файл  `config.cache',  в  котором  сохраняются результаты
проведённых  тестов  для  ускорения  переконфигурации,  и  файл  `config.log',  содержащий  вывод
компилятора (полезен, в основном, для отладки `configure').

   Если  для компиляции пакета  Вам  требуется  сделать  какие-то  нестандартные  вещи, попытайтесь
выяснить,  каким  образом  `configure'  может проверить возможность их выполнения, и отправьте diff-
файлы  или инструкции на адрес, данный в  `README'  для  того,  чтобы  они могли быть рассмотрены в
следующих релизах.  Если в файле  `config.cache'  содержатся результаты тестирования, которые Вы не
хотели бы сохранить, можно удалить или отредактировать его.

   Файл  `configure.in' используется  для  создания  скрипта  `configure'  программой,  называемой
`autoconf'.  `configure.in'  может  Вам понадобиться только в случае, если Вы захотите изменить его
или пересоздать скрипт `configure', используя более новую версию `autoconf'.

Простейший путь компиляции данного пакета:

  1. Выполните  `cd'  в  директорию, содержащую исходный код пакета, и  введите  `./configure'  для
     настройки пакета под Вашу систему. Если Вы используете `csh' на старой версии System V, вместо
     этого,  возможно,  потребуется ввести  `sh ./configure',  чтобы не дать  `csh'  самой выполнить
     `configure'.

     Выполнение  `configure'  займёт некоторое время.  Во время выполнения скрипт выводит некоторые
     сообщения о том, какие функции он проверяет.

  2. Введите `make' для компиляции пакета.

  3. При необходимости введите `make check' для запуска тестов самопроверки, поставляемых с пакетом.

  4. Введите `make install' для установки программ, других файлов с данными и документацией.

  5. Чтобы удалить бинарники  программ  и  объектные файлы из директории  с  исходным кодом,  можно
     ввести  `make clean'.  Чтобы удалить также файлы, созданные `configure' (для того, чтобы можно
     было собрать пакет для  различных типов  компьютеров),  введите  `make distclean'.  Есть также
     возможность запуска  `make maintainer-clean',  но  данная  возможность  предназначена  главным
     образом для разработчиков пакета.  Если Вы используете её, Вам могут понадобиться средства для
     восстановления файлов, полученных с дистрибутивом.

Компиляторы и опции
====================

   Некоторые системы требуют нестандартных опций компиляции  или  компоновки,  о  которых  не знает
скрипт  `configure'.  Вы можете  дать  `configure'  начальные значения переменных, установив  их  в
окружении.  При использовании  Bourne-совместимых оболочек Вы можете сделать это в командной строке
таким образом:
     CC=c89 CFLAGS=-O2 LIBS=-lposix ./configure

На системах, в которых есть программа `env', можно сделать это таким образом:
     env CPPFLAGS=-I/usr/local/include LDFLAGS=-s ./configure

Компиляция для нескольких архитектур
```

==========

Вы можете компилировать пакет для более чем одного типа компьютеров одновременно путём размещения объектных файлов для каждой архитектуры в их собственную директорию. Для этого Вам нужно использовать версию `make', поддерживающую переменную `VPATH', такую как GNU `make'. Выполните `cd' в директорию, объектные и запускаемые файлы из которой Вы хотите применить, и запустите скрипт `configure'. `configure' автоматически проверит исходный код в директории, в которой он находится, и в директории `..'.

Если Вы используете `make', не поддерживающий переменную `VPATH', Вам придётся компилировать пакет в директории с исходным кодом для каждой архитектуры отдельно. После установки пакета для одной архитектуры выполните `make distclean' перед переконфигурацией для другой архитектуры.

Пути и имена при установке

По умолчанию `make install' будет устанавливать файлы пакета в `/usr/local/bin', `/usr/local/man', и т.д. Можно задать префикс установки, отличный от `/usr/local' с помощью опции `--prefix=PATH' скрипта `configure'.

Можно задать разные префиксы установки для архитектурно-зависимых и архитектурно-независимых файлов. Если Вы передаёте `configure' опцию `--exec-prefix=PATH', PATH будет использован в пакете в качестве префикса для установки программ и библиотек. Документация и другие файлы с данными будут по прежнему использовать обычный префикс.

Кроме того, если Вы используете нестандартное расположение директорий, можно задать опции типа `--bindir=PATH' для указания различных значений для конкретных типов файлов. Запустите `configure --help' для получения списка директорий, которые Вы можете задать, и типов файлов, которые будут помещены в них.

Если пакет поддерживает это, можно указать, чтобы программы при установке получали дополнительный префикс или суффикс к их именам, передавая `configure' опции `--program-prefix=PREFIX' или `--program-suffix=SUFFIX'.

Дополнительные функции

Некоторые пакеты обращают внимание на опции `configure' `--enable-FEATURE', где FEATURE указывает на необязательную часть пакета. Они также могут принимать во внимание опции `--with-PACKAGE', где PACKAGE - что-то вроде `gnu-as' или `x' (для X Window System). В файле `README' перечислены все опции `--enable-' и `--with-', принимаемые пакетом.

В пакетах, использующих X Window System, `configure' обычно может автоматически найти подключаемые и библиотечные файлы X, в противном случае Вы можете использовать опции `configure' `--x-includes=DIR' и `--x-libraries=DIR', чтобы указать их местоположение.

Указание типа системы

Могут встречаться некоторые особенности, с которыми `configure' не может разобраться автоматически, но которые требуется определить для того типа хоста, на котором пакет будет работать. Обычно `configure' сам способен определить это, но если он выводит сообщение о том, что не может угадать тип хоста, задайте его с помощью опции `--host=TYPE'. TYPE может быть либо коротким именем типа системы, таким как `sun4', либо каноническим именем из трёх полей:

     CPU-COMPANY-SYSTEM

В файле `config.sub' перечислены возможные значения для каждого поля. Если файла `config.sub' нет в составе пакета, значит данному пакету не требуется знать тип хоста.

Если Вы создаёте средства для кросс-платформенной компиляции, Вы можете также воспользоваться опцией `--target=TYPE', чтобы выбрать тип системы, для которой будет компилироваться код, и опцию `--build=TYPE', чтобы выбрать тип системы, на которой Вы компилируете пакет.

Значения по умолчанию для использования в нескольких сборках

Если Вы хотите задать значения по умолчанию скрипта `configure' для дальнейшей сборки на других компьютерах, Вы можете создать сценарий оболочки, специфичный для Вашей системы, называемый `config.site', где задаются значения по умолчанию для переменных типа `CC', `cache_file' и `prefix'. `configure' изучает `PREFIX/share/config.site', если таковой существует, затем `PREFIX/etc/config.site', если таковой с уществует. Либо Вы можете задать переменную окружения `CONFIG_SITE' для указания местонахождения специфичного для Вашей системы сценария.
Предупреждение: не все скрипты `configure' осуществляют поиск специфичного для Вашей системы сценария.

Контроль над ходом работы

`configure' распознаёт следующие опции контроля над ходом своей работы:

`--cache-file=FILE'
     Использовать и сохранять результаты тестов в FILE вместо `./config.cache'. Для отключения кэширования при отладке `configure', установите FILE в `/dev/null'.

`--help'
     Вывести список всех опций `configure' и завершить работу.

```
`--quiet'
`--silent'
`-q'
     Не выводить сообщения  о выполняемых проверках.  Чтобы  пресечь все сообщения,  поступающие на
     стандартный вывод,  перенаправьте  его  в  `/dev/null'  (это  не пресечёт  вывода сообщений об
     ошибках).

`--srcdir=DIR'
     Искать исходный  код  пакета  в  директории DIR.  Обычно  `configure'  способен определить эту
     директорию автоматически.

`--version'
     Вывести версию Autoconf,  которая использовалась для создания скрипта `configure', и завершить
     работу.

`configure' также принимает некоторые другие, не используемые широко опции.
```

К содержанию

K. Уведомления об авторских правах на программное обеспечение OpenLDAP

K.1. Уведомление об авторских правах OpenLDAP

K.2. Дополнительные уведомления об авторских правах

Распространение и использование в форме исходных кодов и бинарных файлов, с модификацией или без неё, разрешено при условии, что данное уведомление будет сохранено. Имена владельцев авторских прав не могут быть использованы для одобрения или продвижения продуктов, основанных на этом программном обеспечении, без их предварительного письменного разрешения. Данное программное обеспечение поставляется ``как есть", без явных или подразумеваемых гарантий.

K.3. Уведомление об авторских правах Мичиганского Университета

Распространение и использование в форме исходных кодов и бинарных файлов разрешено при условии, что данное уведомление будет сохранено, и при оказании должного уважения Мичиганскому Университету в Энн-Арбор. Название Университета не может быть использовано для одобрения или продвижения продуктов, основанных на этом программном обеспечении, без предварительного письменного разрешения. Данное программное обеспечение поставляется ``как есть", без явных или подразумеваемых гарантий.

К содержанию

L. Открытая лицензия OpenLDAP

Примечание переводчика: Данный перевод лицензии не заверен нотариально и не является "точной копией" ("verbatim copy") оригинальной лицензии OpenLDAP. Приводится с целью облегчения понимания условий лицензии.

The OpenLDAP Public License
Версия 2.8, 17 августа 2003 года

Распространение и использование данного программного обеспечения и соответствующей документации ("Программное обеспечение"), с изменениями или без, разрешено при соблюдении следующих условий:

1. При распространении в виде исходного кода должны сохраняться заявления и уведомления об авторских правах,

2. При распространении в двоичной форме в документации и/или других материалах, поставляемых при распространении, должны воспроизводиться соответствующие заявления и уведомления об авторских правах, этот список условий и приведённый ниже отказ от ответственности, и

3. При распространении дистрибутив должен содержать точную копию этого документа.

Фонд OpenLDAP время от времени может пересматривать эту лицензию. Каждая версия лицензии отличается уникальным номером. Вы можете использовать данное программное обеспечение на условиях этой версии лицензии или на условиях любой последующей версии лицензии.

ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДОСТАВЛЕНО ФОНДОМ OPENLDAP И ЕГО СОТРУДНИКАМИ ``КАК ЕСТЬ'', БЕЗ КАКИХ-ЛИБО ЯВНЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ ГАРАНТИЙ. ТАКИМ ОБРАЗОМ, ЛЮБЫЕ ГАРАНТИИ, ВКЛЮЧАЯ, ПОМИМО ПРОЧЕГО, КОСВЕННЫЕ ГАРАНТИИ ТОВАРНОГО КАЧЕСТВА И ПРИГОДНОСТЬ ДЛЯ ЭКСПЛУАТАЦИИ ДЛЯ ОСОБЫХ ЦЕЛЕЙ, НЕ ПРИЗНАЮТСЯ. НИ ПРИ КАКИХ УСЛОВИЯХ ФОНД OPENLDAP, ЕГО СОТРУДНИКИ, АВТОРЫ ИЛИ ВЛАДЕЛЬЦЫ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ НЕ НЕСУТ НИКАКОЙ ОТВЕТСТВЕННОСТИ ЗА ЛЮБЫЕ ПРЯМЫЕ, КОСВЕННЫЕ, СЛУЧАЙНЫЕ, СПЕЦИАЛЬНЫЕ, ШТРАФНЫЕ ИЛИ ПОНЕСЁННЫЕ ВПОСЛЕДСТВИИ УБЫТКИ (ВКЛЮЧАЯ, ПОМИМО ПРОЧЕГО, ПРИОБРЕТЕНИЕ ЗАМЕЩАЮЩИХ ТОВАРОВ ИЛИ УСЛУГ, УТРАТУ ВОЗМОЖНОСТИ ЭКСПЛУАТАЦИИ, ПОТЕРЮ ДАННЫХ ИЛИ ПРИБЫЛИ, ПЕРЕРЫВ В ПРОИЗВОДСТВЕ) ЧЕМ БЫ ОНИ НЕ БЫЛИ ВЫЗВАНЫ И НЕСМОТРЯ НА КАКУЮ-ЛИБО ТЕОРИЮ ОТВЕТСТВЕННОСТИ, ПО КОНТРАКТУ, БЕЗУСЛОВНОМУ ОБЯЗАТЕЛЬСТВУ ИЛИ В СЛЕДСТВИЕ ПРАВОНАРУШЕНИЯ (ВКЛЮЧАЯ ХАЛАТНОСТЬ ИЛИ ЧТО-ЛИБО ДРУГОЕ), ЯВЛЯЮЩИЕСЯ РЕЗУЛЬТАТОМ ЭКСПЛУАТАЦИИ ДАННОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ, ДАЖЕ В СЛУЧАЕ УВЕДОМЛЕНИЯ О ВОЗМОЖНОСТИ ДАННЫХ УБЫТКОВ.

Имена авторов и владельцев авторских прав без предварительного письменного разрешения не должны быть использованы в рекламе или ином способе стимулирования продаж, использования или других операций с данным Программным обеспечением. Декларирование авторских прав в данном Программном обеспечении должно всегда приводиться вместе с владельцами авторских прав.

OpenLDAP является зарегистрированной торговой маркой Фонда OpenLDAP.

К содержанию