Проектируем справочную службу с помощью протокола LDAP.

mkondrin
из hppi.troitsk.ru

28 ноября 2007 г.

Аннотация

Протокол LDAP хорошо известен как пользователям, например, в качестве протокола для раздачи корпоративных адресных книг, так и системным администраторам, которые используют его для хранения сведений об учётных записях, компьютерах и другой системной информации. Однако протокол LDAP достаточно прост и гибок, чтобы с его помощью можно было создать свою собственную нестандартную справочно-информационную службу.

LDAP - lightweight directory access protocol является упрощённым вариантом более старого протокола DAP. Название этого протокола обычно переводят на русский язык как протокол службы каталогов, хотя более точно было бы назвать его просто протоколом справочной или информационной службы [1]. Так что LDAP в сущности - это иерархическая база данных оптимизированная для применений, когда чтение из базы происходит существенно чаще, чем добавление в нее новых записей или изменение старых. Два наиболее типичных применения LDAP: адресная книга и учетные записи пользователей операционной системы, прекрасно иллюстрируют пример базы данных с не часто изменяющейся информацией. Однако, если подумать, то достаточно много мы сталкиваемся как раз с такой информацией, которой мы часто пользуемся, но редко переписываем. В принципе это может быть любой печатный справочник (какие нибудь ”Сборник типовых документов, рекомендованных к использованию...”) или же где-то купленная и неизменяемая база данных (например с данным о кристаллических структурах бинарных химических соединений). В принципе, даже прайс-лист какого-нибудь магазина может быть реализован в виде LDAP: хотя и меняется он довольно часто, пусть даже раз в сутки, но продавцам всегда хочется надеяться, что покупатели и клиенты заглядывают в него во много раз чаще! С другой стороны, хотя понятие “база данных” сейчас ассоциируется исключительно со словом ”реляционная но сравнению с ними иерархическая модель данных, принятая в LDAP, может оказаться гораздо удобной[2]. Тем более, что иерархическая модель более “интуитивно-понятна” большинству пользователей компьютеров, поскольку у всех перед глазами есть пример такой базы данных - файловая система. К примеру, в случае с прайс-листом иерархическая модель полностью совпадает с общепринятой практикой - группированием продуктов по назначению, торговым маркам и ценовому диапазону. Немаловажным

[1] Одно из значений directory - телефонный справочник

[2] Хотя иерархические структуры можно реализовать в реляционной модели и наоборот, но это преобразование требует известных усилий

моментом, также является и простота разработки. В то время как деятельность администраторов и разработчиков баз данных существенно отражается на бюджете любого предприятия, то разработка же справочной службы в виде LDAP не требует навыков программирования - от автора требуется только описать иерархию объектов, представляющих данные. Последним (в смысле очерёдности, но не важности!) же аргументом является стандартизованность протокола LDAP - в отличии от баз данных, где совместимость со стандартом SQL еще не гарантирует совместимость клиента и сервера, любой LDAP-клиент можно использовать для подключения к любому серверу LDAP, если только они используют одинаковый номер версии протокола.

В качестве примера в данной статье будет рассмотрен пример построения библиографической информационной системы. Схема и несколько утилит были разработаны итальянским программистом Пиранжело Мазарати (Pierangelo Masarati) [1] в 1999 году для распространённой тогда версии LDAPv2 и выложены в открытый доступ [2]. Я же, взяв за основу его пакет `ldap2bibtex`, немного видоизменил структуру данных, добавил совместимость с новой версией протокола LDAPv3 и возможности сильной аутентификации доступные в новом протоколе. Обновлённую версию пакета можно скачать здесь [3]. На мой взгляд поскольку эта справочная система достаточно проста и ее функции понятны любому читателю, то она является хорошим примером, с помощью которого можно разобрать основные принципы построения нестандартных информационных служб.

1 Что такое BibTeX и для чего нужна библиографическая база данных?

BibTeX - это база данных и программа для хранения библиографическими данными предназначенная для работы в составе LaTeX. Краткий обзор возможностей BibTeX и LaTeX можно найти здесь [4], хотя там несколько неправильно расставлены акценты. На самом деле ведение библиографии является достаточно важной частью любой исследовательской работы и представляет интерес само по себе - любопытно же узнать, что до вас сделали другие люди в интересующей вас области. Поэтому одним из функций BibTeX является ведение файловой базы с выходными данными интересующих вас статей и книг. Записи из этой базы в последствии легко можно вставить в свою собственную работу, причём в зависимости от выбранного стиля эта ссылка будет соответствующим образом оформлена (издательство может требовать не указывать название цитируемой работы), а ссылка на эту публикацию будет пронумерована в зависимости от места появления её в тексте. Хотя такая автоматизация при подготовке публикации и является весьма полезной чертой, но не менее важной функцией BibTeX следует считать возможность упорядоченного хранения библиографической базы данных. Пока библиография не разрослась, пока вы используете один клиентский компьютер и у вас нет необходимости делиться своими данными с коллегами, то BibTeX является идеальным решением. В самом деле - в своём домашнем каталоге в файле `~/texmf/bibtex/bib/Mypub.bib` вы собираете интересующие вас библиографические записи такого вида:

```
@Article{baldin06,
  author =   {Е.Балдин},
  title =   {Латех -- компьютерная типография. Введение},
  journal =   {LinuxFormat},
  year =   2006,
```

```
  volume =  83,
  number =  9
}
```

Поскольку подкаталоги `~/texmf` автоматически просматриваются при запуске bibtex и latex, то достаточно вставить в своей работе ссылку вида `\cite{baldin06}`, а в конце добавить две строки вида

```
\bibliography{Mypub}
\bibliographystyle{unsrtnat}
```

и тогда последующий запуск команд latex bibtex автоматически сгенерирует запись в ссылке литературы и автоматически пронумерованную ссылку на нее, точно так же как в данной статье пункт [4].

BibTeX очень популярный формат, и к нему не просто привыкли, а имеется большое количество утилит для работы с ним, например графические оболочки - tkbibtex[5], jabref[6], клиентские модули для emacs reftex и bibtex-mode[7], набор фильтров для преобразования из одного формата в другой... Но как только возникает необходимость каким-то образом “масштабировать” базу данных BibTeX - обеспечить выдачу её на несколько компьютеров или обеспечить её совместное ведение группой людей, как начинаются проблемы связанные с ее файловым характером.

Для преодоления этого ограничения BibTeX была сделана пара попыток создания реляционных библиографических баз данных - bibus и refdb [8, 9]. Хотя эти решения можно использовать, но мне они показались сложными в настройке, повседневном администрировании и (самое главное) довольно “негибкими” в том смысле, что реляционная модель с её табличным форматом слишком жёсткая, чтобы хранить в ней BibTeX - овскую информацию. Иными словами - при переносе в базу данных происходит конвертирование BiBTeX-овских записях в табличные колонки. Но практически все поля там в отличии от таблицы - необязательны. Например, в приведённой выше записи вполне допустимо добавить поле abstract с аннотацией к статье или поле url с ссылкой на опубликованный в сети вариант этой статьи. Для оформления ссылок в публикациях эти поля не нужны и как правило игнорируются, но очень полезны для упрощения работы с библиографической базой. Хотя для BibTeX нормально, но такие полупустые колонки в реляционной базе данных способны повергнуть в ужас любого разработчика и администратора баз данных, которые склонны видеть в этом святотатство по отношению к “пятой нормальной форме”.

На мой взгляд вариант предложенный Перанжело Мазарати с LDAP и иерархической структурой данных более естественно подходит для работы с библиографией, чем реляционные базы данных. Так что перейдём к запуску сервера LDAP и его заселению.

2 Собираем и запускаем сервер OpenLDAP.

Как следует из названия параграфа для хранения ссылок будет использован открытый и не требовательный к ресурсам сервер OpenLDAP. Это не единственный вариант, поскольку есть еще несколько серверов сертифицированных на совместимость со стандартом LDAPv3 - например, Apache Directory Sever или Sun Open Directory Server, оба написанные на Java [10, 11]. Хотя есть свои резоны предпочесть именно их (например, из-за возможности модифицировать схемы на лету без

перезагрузки сервера), но в данном случае нетребовательность к ресурсам и простота для нас важнее.

Если по каким-то причинам вы решите собирать сервер OpenLDAP из исходников [12](разумеется, собирается он стандартной последовательностью команд `./configure; make; make install`), то обратите внимания на следующие важные опции скрипта ./configure :

```
--enable-bdb=yes
--with-cyrus-sasl=yes
```

Первая из опций указывает на необходимость создания bdb движка для хранения баз данных OpenLDAP, а вторая требует линковки с оболочкой простой криптографической защиты Cyrus-SASL[13]. Первая из опций важна, поскольку создаваемый по умолчанию движок ldbm и быстрее и легче, чем bdb, но у него совершенно удручающие показатели работы с alias-ами, которые хотя не являются строго обязательными, но очень удобны при работе с библиографическими записями. Эта опция требует наличия в системе библиотек для работы с базами BerkleyDB последней 4-ой версии [14]. Использование же Cyrus-SASL позволяет вынести проблемы с аутентификацией пользователей за пределы OpenLDAP на отдельный уровень и решить их раз и навсегда для всей системы. Строго говоря, для нашей задачи это не столь важно, но умение решать проблемы до их появления ещё никому не принесло большого вреда.

Заодно перед запуском сервера можно скомпилировать пакет `ldap2bibtex` [3]. Этот пакет не имеет скрипта configure и перед сборкой возможно потребуется внести кое-какие изменения в Makefile (скорее всего потребуется поправить пути к include файлам). После запуска make в рабочей директории должны появиться два исполняемых файла ldap2bibtex и bibtex2ldif, для извлечения данных из ldap сервера и его заполнения. Нас же в данный момент интересует файл схемы ldap2bibtex.v3.schema, который находится в поддиректории `etc` данного каталога. Его нужно переместить в каталог `/etc/openldap/schema` и создать конфигурационный файл сервера OpenLDAP `/etc/openldap/slapd.conf`

```
include         /etc/openldap/schema/core.schema
include         /etc/openldap/schema/ldap2bibtex.v3.schema

pidfile         /var/run/slapd.pid
argsfile        /var/run/slapd.args

database        bdb
suffix          "ou=Bib,o=Library"
rootdn          "cn=root,ou=People,o=Library"
rootpw  testpass

directory       /var/lib/ldap

index   objectClass     eq
```

Это минимальный файл, который на протяжении этой статьи будет ещё пару раз усовершенствован. Во-первых, в него включены (директивами `include`) базовая схема и собственная схема библиографической базы данных - если сервер будет использован только под библиографическую базу данных, то большего и не требуется. В качестве движка для базы данных выбран BerkleyDB (`database bdb`),

причём в каталоге LDAP все записи будут располагаться в узле `suffix`, а сами файлы базы данных - в каталоге `/var/lib/ldap`. Ключевое слово `database` кроме того отделяет область глобальных настроек, применимых к серверу LDAP целиком, от настроек специфичных для конкретного каталога. Замечу, что один сервер LDAP может иметь произвольное число отдельных каталогов, каждый со своими настройками и движками БД. Добавлять записи в каталог `ou=Bib,o=Library` может только оператор с именем `rootdn` идентифицируемый паролем `rootpw`, чтение же не требует пароля и разрешено всем (поскольку в конфигурационном файле ничего об этом не сказано, то используется именно такая политика по-умолчанию). Этого достаточно чтобы запустить сервер OpenLDAP (командой `slapd`) и начать заполнение библиографической базы данных. Однако перед этим стоит напомнить некоторые сведения о внутреннем устройстве LDAP

3 Внутреннее устройство LDAP

Данные в LDAP организованы в пары “имя атрибута:его значение”, которые сгруппированы в контейнеры, помеченные уникальным именем и собранные в иерархический каталог или Directory Information Tree (DIT). Это очень напоминает файловую структуру, но в LDAP нет разделения на файлы и директории, т.е. любой контейнер может одновременно содержать в себе информацию и одновременно являться группирующим элементом. Какие именно атрибуты содержит в себе тот или иной элемент, зависит от типа элемента, а единственным необходимым требованием остаётся наличие у каждого элемента уникального имени (оно же distingushed name - DN), по которому определяется положение этого элемента в иерархическом каталоге.

Каждый сервер OpenLDAP поддерживает ограниченное число типов элементов, которые обязаны быть описаны в конфигурационном файле до запуска сервера, и не могут быть изменены в процессе работы сервера [3]. Обычно, описания типов выносятся в отдельные файлы - файлы схем, подгружаемые в конфигурационный файл с помощью директив `include`. В стандартном дистрибутиве OpenLDAP имеются несколько готовых схем - помимо базовых (core, cosine), ещё схема для хранения учётных записей POSIX-типа (файл inetorgperson.schema), учётных записей пользователей Samba (файл samba.schema), структуры для хранения паролей Kerberos (Heimdal-Kerberos можно настроить таким образом, что в качестве своей базы данных он будет использовать базу данных LDAP). Каждая схема в свою очередь может содержать описание нескольких типов классов и атрибутов LDAP.

Собственно говоря, основная часть работы при проектировании нестандартной информационной службы связана с проектированием файла-схемы. Описание класса включает в себя название класса, ссылку на родительский класс, перечисление составляющих его атрибутов и их модальность, т.е. являются ли их присутствие в объекте обязательными или опциональным. Классы LDAP также представляют собой иерархическую структуру, так что атрибуты могут наследоваться потомком от своего родителя, причем если один и тот же атрибут определен как в родителе так и в потомке, то объявления в потомке имеют приоритет. Объекты в каталоге могут быть составными, т.е. при соблюдении некоторых условий, о которых речь пойдет ниже, объект может обладать свойствами нескольких классов. В этом случае его свойства представляют из себя объединение свойств каждого из классов.

[3] Напомню, что это является особенностью именно OpenLDAP, другие сервера LDAP могут быть свободными от этого ограничения и допускать добавление новых типов без перезапуска сервера.

В файле схемы также определяются типы атрибутов. В описание атрибута включается его имя, указание его “синтаксиса”, т.е. типа, который может принимать значение атрибута (к примеру, ASCII-строка, строка в кодировке UTF8, целочисленный тип - это все разные “синтаксисы”), а также и сведения о способе индексации и процедуре поиска. Для оптимизации процедуры поиска также стоит указывать множественность атрибута, т.е. может ли он встречаться несколько раз в классе или нет. Скажем, название книги не имеет смысла объявлять как множественный атрибут. Также как и классы, атрибут также может наследовать свойства другого атрибута.

Предком всех объектов является объект `top` с единственным обязательным множественным атрибутом objectClass с значением top. Каждый из потомков, помимо своих собственных атрибутов, добавляет к нему ещё одно дополнительное поле objectClass с именем своего класса.

Обмен данными между сервером LDAP и внешним миром (клиентами) происходит посредством стандартизованного формата `ldif`. Он собственно и представляет из себя пары “имя атрибута - его значение”, разделённые двоеточием и упорядоченные по классам. Разумеется, этот формат является лишь промежуточным носителем и не имеет ничего общего с реальным форматом, в котором данные хранятся на LDAP сервере.

Чтобы более подробно разобраться с форматом ldif и классами ldap, создадим корневую ветку на нашем сервере, в соответствии с описанием в приведённом выше конфигурационном файле. Она состоит из двух объектов типа `organization/o` и `organizational unit/ou` [4] - непосредственных потомков класса `top`.

Для этого нужно создать текстовый файл `suffixdn.ldif` такого вида:

```
dn: o=Library
o: Library
objectClass: top
objectClass: organization

dn: ou=Bib,o=Library
ou: Bib
objectClass: top
objectClass: organizationalUnit
```

Небольшие комментарии. Атрибут dn (distinguished name/уникальное имя) - как уже говорилось выше, с одной стороны однозначным образом идентифицирует объект, а с другой стороны определяет его расположение в каталоге. Каким образом dn идентифицирует объект не столь важно, т.е. для этой цели может быть использованы любые из атрибутов объекта и его предков, хотя как правило существует какая-то традиция выбора определённых атрибутов для помещения в dc. В данном случае объект `ou=Bib,o=Library` является потомком (в смысле расположения в каталоге, а не в смысле наследования классов) объекта `o=Library`, причём использованию полей o и ou в качестве элементов уникального имени dn практически нет вариантов, так как это единственные обязательные атрибуты для классов

[4] Здесь стоит упомянуть, что есть два альтернативных способа именования корневых объектов сервера LDAP. Используемый в данной статье предполагает, что верхний узел - это какая-то реальная или фиктивная организация со своими подразделениями. Второй метод основан на более глобальном взгляде на вещи, и корень LDAP в этом случае представляет из себя аналог доменного имени, с корневыми объектами класса `Domain Controller/dc`, так что уникальное имя корня имеет вид `...,dc=mysubdomain,dc=mydomain,dc=ru`.

organization и organizational unit соответственно. Также при описании объектов в ldif файле можно опустить полное перечисление всех родительских классов (вроде objectClass: top в данном примере), достаточно только указать класс самого объекта.

Этот файл можно загрузить на сервер командой:

```
ldapadd -h ldap.servername.ru -x \\
        -D "cn=root,ou=People,o=Library" \\
        -W -c -f suffixdn.ldif
```

и проконтролировать результат с помощью универсального инструмента поиска в каталоге LDAP:

```
ldapsearch -h ldap.servername.ru -x \\
    -D "cn=root,ou=People,o=Library" -W\\
    -b "o=Library" -s sub "(ou=Bib)" ou
```

В обеих случаях флажок -h указывает сервер LDAP, -D - пользователя информационной службы, причём в первом случае у него должны быть права на запись в каталог, т.е. это должен быть пользователь указанный как rootdn в конфигурационном файле, с помощью флагов -W и -x включаются запрос пароля и его проверка через простой механизм аутентификации вместо принятой по умолчанию проверки средствами SASL (о чём речь пойдёт ниже). В данном случае паролем будет значение параметра rootpw из конфигурационного файла. Ключ -c означает режим continue-on-error, т.е. работа программы не прерывается при ошибке, в частности (самый распространённый случай), когда объект с определённым значением dn уже существует. Для программы ldapsearch специфическими являются параметры -b указывающий верхний узел, с которого начинается поиск, -s со значением base/one/sub обозначающий является ли поиск рекурсивным или нет (отсутствие рекурсии, рекурсия на глубину одного подкаталога и полная рекурсия), а также два последних параметра - фильтр поиска и список выводимых атрибутов (последний является необязательным). Поскольку фильтры являются основным инструментом обеспечивающим быстрый поиск информации в каталоге LDAP, то сделаем небольшое отступление и рассмотрим их несколько подробнее.

Базовым блоком, составляющим фильтр, является выражение вида (круглые скобки обязательны):

```
(<имя атрибута><операция  сравнения><значение>)
```

Операций сравнения бывают трёх типов: = точное равенство, ~= примерное равенство, в смысле “звучит как”, а также соотношения “больше-меньше”, что для строк означает алфавитного упорядочение, <= и >=. В правой части для соотношений типа точного равенства можно использовать шаблоны с символом “*”, под который подпадает любая последовательность знаков. Базовые фильтры можно объединять с помощью логических операций “&” - “И”, “|” - “ИЛИ”, “!” - “НЕ” в более сложные фильтры вида (круглые скобки опять обязательны):

```
(<логическая операция><базовый фильтр1><базовый фильтр2> ...)
```

Например, для нашего дерева можно придумать такой фильтр:

```
(&(ou~=Beeb)(objectClass=organizationalUnit)(!(ou=Bob)))
```

При виде сравнения строк естественным образом напрашивается вопрос “Чувствительно ли сравнение к регистру букв? Как сравниваются строки отличные от ACSII?” и др. Ответ в данном случае зависит от типа сравниваемых атрибутов и будет рассмотрен несколько ниже.

Фильтры дорускают и так называемый расширенный синтаксис вида:

```
(o:dn:=Library)
```

что позволяет отыскивать точное соответствие одновременно в атрибутах объектов и отдельных частях их уникальных имен. Таким образом, это позволяет находить в каталоге не отдельные объекты, а сразу целые “ветви” каталога, из которых достаточно умный клиент может составить исчерпывающее описание объекта. К примеру, приведенный выше фильтр выберет оба созданных нами объекта, поскольку у `ou=Bib,o=Library` под искомое равенство подпадает хвост уникального имени. В итоге нами получено описание организации со всеми её подразделениями (пусть пока они и в единственном числе).

Теперь у нас всё готово для создания библиографической базы данных. Перед тем как заполнять её стоит более подробнее посмотреть на файл-схему этой информационной системы. Возможно, в будущем вам потребуется его редактировать самостоятельно.

4 bibtex2ldap.v3.schema

Первый вопрос, который возникает у неподготовленного человека при взгляде на schema-файл - “Что это за цифры с точками, напоминающие интернет-адреса, и для чего они нужны?”. Если на первую часть вопроса ответить довольно просто (”Это так называемые цифроиды/numericoids, которые в данном случае служат идентификаторами классов - OID”), то со второй придётся довольно долго выкручиваться.

Для чего вообще понадобились OID? Дело в том,что изначально LDAP как и его предок DAP/X.500 задумывался в виде мировой информационной системы, связанной в единую сеть, наподобие доменной системы имён. Так же как и DNS LDAP поддерживает отсылки (referrals), т.е. если запрашиваемая информация не может быть найдена на одном из серверов LDAP, то он перенаправляет клиента на вышестоящий сервер, который или знает эту информацию или знает сервер, на каком её можно найти. Хотя в виде разрозненных “ островков” кластеров, предназначенных для балансировки нагрузки или для хранения большого объема информации, эта особенность LDAP используется и в настоящее время, но тем не менее до общемировой информационной службы LDAP так и не дорос. Однако, такая конструктивная возможность (возможно, что с прицелом на будущее) в нем заложена[5], причём OID в этом играет одну из ключевых ролей, препятствуя “конфликту имён” (namespace pollution). С этой проблемой, когда разные объекты случайно получают одинаковые имена, сталкиваются многие разработчики больших распределённых систем. В LDAP эта проблема решается введением глобально-уникального идентификатора, причём в этом случае от имени объекта не требуется быть уникальным, оно представляет из себя лишь псевдоним OID. Уникальность OID обеспечивается также как и уникальность интернет-адресов, т.е. административными средствами, для чего разработчику информационной системы нужно поместить заявку на

[5]Можно в этой связи вспомнить и про упоминавшуюся выше традицию наименования корневых узлов LDAP посредством объектов типа dc.

Рис. 1: Часть структуры ldap2bibtex.

определённый диапазон значений OID в специальную организацию. Причём, в отличии от тех же интернет адресов, в LDAP нет приватных диапазонов, т.е. конечно же внутри своей сети в отладочных целях разработчик может использовать те OID, которые ему нравятся, но при опубликовании своих разработок ему так или иначе придётся получать “официальный” диапазон. Проще всего это сделать обратившись в IANA [15], которая бесплатно резервирует OID начинающиеся с последовательности 1.3.6.1.4.1. Следующее число в последовательности, является уникальным, которое приписывается запросившему его разработчику, а к нему уже сам разработчик может присоединить любую последовательность чисел. Просмотреть уже занятые OID можно на сайте [16], в частности можно убедиться, что OID 1.3.6.1.4.1.7036 закреплён за Pierangelo Masarati и его программой MBDyn (свободная программа для расчёта динамики летательных аппаратов [17]). Цифроид 1.3.6.1.4.1.7036.1 используется в самой программе MBDyn, а 1.3.6.1.4.1.7036.2 - в программе ldap2bibtex. Какую нумерацию использовать в OID - это, конечно же, выбор разработчика, но обычно используют последовательности, воспроизводящие иерархию атрибутов и классов в schema файле, так что для ldap2bibtex цифроиды, начинающиеся с 1.3.6.1.4.1.7036.2.1, обозначают атрибуты, а с 1.3.6.1.4.1.7036.2.2 - классы.

Если же разработчик принадлежит тому типу людей, которые считают, что номер типа 77777 на бампере автомобиля - “это круто!”, то он может обратиться в хорошо известную организацию ANSI, где за определённую плату можно подобрать себе OID по своему вкусу [18].

Структура bibtex2ldap схемы (точнее небольшая её часть) приведена на Рис. 1. Для того чтобы подробнее разобраться в синтаксисе schema-файлов можно заглянуть в руководства на английском языке - [19, 20, 21]. Здесь же я дам только необходимые пояснения.

Каждый тип записей bibtex представлен в schema-файле в виде отдельного класса LDAP с очевидным соответствием, т.е. класс типа `bibtexArticle` соответствует bibtex записи `@Article`. Всего таких классов 13. Каждый из них является потомком класса `bibtexEntry` с единственным обязательным полем `bibtexEntryTag`, который с одной стороны соответствует ключу bibtex-записи (т.е. полю используемому в качестве аргумента команды `\cite`), а с другой является элементом `dn` для объекта LDAP. Также класс содержит список из 33 необязательных атрибутов, очевидным образом соответствующих полям в bibtex записях, например, атрибут `bibtexAuthor` соответствует полю `author` в bibtex-записи. Каждый из потомков класса `bibtexEntry` может объявлять некоторые из этих атрибутов обязательными, в частности, атрибут `bibtexAuthor` обязателен для класса `bibtexArticle`, но по-прежнему опционален для класса `bibtexManual`. Жёстких указаний какой атрибут считается обязательным для какого типа bibtex-записей не существует, поэтому мной были использованы рекомендации пакета `bibtex-mode` [7] из состава `Emacs`.

Вот как выглядит запись для `bibtexEntry` в файле `ldap2bibtex.v3.schema`:

```
objectClass ( 1.3.6.1.4.1.7036.2.2.1
        NAME            'bibtexEntry'
        DESC            'Entry for BibTeX application'
        SUP             top
        STRUCTURAL
```

```
MUST (
        bibtexEntryTag
)
MAY (
         cn $
        bibtexEntryType $
        bibtexAddress $
        bibtexAuthor $
        bibtexBookTitle $
        bibtexChapter $
        bibtexEdition $
        bibtexEditor $
        bibtexHowPublished $
        bibtexInstitution $
        bibtexJournal $
        bibtexMonth $
        bibtexNote $
        bibtexNumber $
        bibtexOrganization $
        bibtexPages $
        bibtexPublisher $
        bibtexSchool $
        bibtexSeries $
        bibtexString $
        bibtexTitle $
        bibtexType $
        bibtexVolume $
        bibtexYear $
        bibtexAbstract $
        bibtexUrl $
        bibtexKeywords $
        bibtexFulltext $
        bibtexKey $
        bibtexAnnote
 ) )
```

Первый элемент - это OID, затем идёт имя класса NAME, причём поскольку имя это на самом деле псевдоним, то можно определить несколько имен для одного и того же OID. DESC-это краткое описание объекта, а SUP - указатель на родительский класс (соответственно для bibtexArticle значение этого поля будет bibtexEntry). Поле ABSTRACT / STRUCTURAL / AUXILIARY указывает каким образом этот класс может быть использован в иерархии объектов. Объектов типа ABSTRACT существовать не может, класс такого типа используется только для получения от него потомков типа STRUCTURAL / AUXILIARY. Примером такого класса является общий предок - абстрактный класс top. В принципе, bibtexEntry тоже можно было бы объявить абстрактным, если бы не требовалась совместимость со старой схемой, где все объекты были этого типа. При наследовании классов необходимо соблюдать "чистоту линии т.е. STRUCTURAL классы создаются из STRUCTURAL/ABSTRACT классов, AUXILIARY классы - из AUXILIARY/ABSTRACT, а ABSTRACT только из ABSTRACT. Разница между структурными и вспомогательными классами проявляется при создании составных объектов, на что накладывается ограничение, что ровно один

из составляющих их классов должен быть STRUCTURAL, а все остальные должны быть AUXILIARY. Т.е. класс типа AUXILIARY это просто вспомогательный контейнер с атрибутами, который может быть "прицеплен"к объекту другого класса типа STRUCTURAL. Поля MUST и MAY перечисляют обязательные и допустимые атрибуты. Для составных объектов, если один и тот же атрибут определен в нескольких родительских классах, но его модальность разная, то в итоге он всегда будет MUST. Точно так же при наследовании классов можно переводить атрибуты их категории MAY в категорию MUST, что, как было показано выше, широко используется в ldap2bibtex схеме. Следует также отметить, что в качестве разделителей полей в схемах LDAP используется знак $, что можно видеть в данном случае на примере MAY. Никаких ограничений на добавление своих собственных полей в bibtex-записях нет, поэтому можно добавлять любые атрибуты. Используемый в пакете ldap2bibtex список полей был составлен на основании длительного опыта использования различных онлайновых библиографических баз данных. В большинстве случаев его достаточно, чтобы предотвратить ошибки, связанные с передачей на сервер LDAP атрибута, который отсутствует в загруженных на сервер схемах.

В схеме также описан специальный класс bibtexString не являющийся потомком bibtexEntry, который служит для хранения bibtex записей @String или просто строковых констант. Эти константы могут вставляться в произвольные bibtex записи в виде макроподстановок. При получении данных из LDAP каталога программой ldap2bibtex, она в начале всегда извлекает из каталога записи этого типа, а только потом уже переходит к собственно библиографическим записям, указанных в её входных параметрах.

Теперь перейдём к атрибутам (по порядку последовательности цифроидов). Первый из них bibtexEntryType оставлен только для совместимости со старой версией ldap2bibtex (также как и необязательное поле cn)/common name в классе bibtexEntry, которое применялось в качестве элемента уникального имени). Сейчас же функции bibtexEntryType выполняет имя класса, а в качестве элемента dn, о чём уже шла речь, используется атрибут bibtexEntryTag. Его описание таково:

```
attributetype ( 1.3.6.1.4.1.7036.2.1.2
        NAME            'bibtexEntryTag'
        DESC            'Tag of BibTeX entry'
        EQUALITY        caseIgnoreMatch
        SUBSTR          caseIgnoreSubstringsMatch
        ORDERING        caseIgnoreOrderingMatch
        SYNTAX          1.3.6.1.4.1.1466.115.121.1.15
        SINGLE-VALUE
        )
```

Элементы OID, NAME и DESC (а также неиспользуемый здесь SUP) полностью аналогичны элементам из определения objectClass. SYNTAX указывает тип элемента, в данном случае в виде очередного OID-а. Этот тип (и ещё несколько других) определены в RFC2252 "LDAP(v3): Attribute Syntax Definitions", где он описан как Directory String или строка в кодировке UTF8. SINGLE-VALUE означает, что у объекта не может быть более одного атрибута такого типа. Поскольку основным предназначением LDAP, помимо хранения данных, является ещё и организация их быстрого поиска, то в определение атрибута включены ещё три поля, определяющие алгоритмы поиска по шаблону и поиска точного соответствия SUBSTR и EQUALITY, а также упорядочивания элементов ORDERING для фильтров "больше-меньше". Заметьте, что первые два правила поиска относятся к точному

равенству (а не к примерному, алгоритм поиска для которого фиксирован[6]), причём какое из этих двух правил будет применено, зависит от того, используется ли знак * в правой части выражений сравнения или нет. Для рассматриваемого сейчас класса все типы поиска нечувствительны к регистру, хотя можно было бы указать и альтернативный вариант - caseExact.... Разумеется, для разного типа атрибутов и алгоритмы поиска будут разные, например для строк, представляющих числа (numericString с OID), в качестве процедур поиска используются numericStringMatch-numericStringSubstringMatch. Для некоторых типов атрибутов процедуры поиска вообще не определены (например для бинарных данных OID=1.3.6.1.4.1.1466.115.121.1.5), а для других - определена только процедура точного поиска. К последнему типу относится атрибут objectClass (со специальным синтаксисом 1.3.6.1.4.1.1466.115.121.1.38), для которого имеется только одно правило поиска objectIdentifierMatch. При этом соответствие типа поиска объявленного для атрибута типу самого атрибута проверяется при загрузке файлов схем (т.е. при запуске сервера), и в случае их несовместимости приводит к остановке сервера. Но для нас это имеет в основном теоретический интерес, поскольку все атрибуты, объявленные в схеме ldap2bibtex, являются потомками bibtexAttribute:

```
attributetype ( 1.3.6.1.4.1.7036.2.1.3
        NAME            'bibtexAttribute'
        DESC            'BibTeX attribute'
        EQUALITY        caseIgnoreMatch
        SUBSTR          caseIgnoreSubstringsMatch
        ORDERING        caseIgnoreOrderingMatch
        SYNTAX          1.3.6.1.4.1.1466.115.121.1.15
        SINGLE-VALUE
        )
```

с одним и тем же типом и алгоритмами поиска. При создании одного атрибута из другого, для чего как и в случае классов используется директива SUP, наследуются все свойства родителя, кроме множественности, в том числе и имя родителя (хотя и в неявном виде). Т.е. в случае ldap2bibtex это означает, что фильтр вида:

```
(&(bibtexAttr=*OpenLDAP*)(objectClass=bibtexEntry))
```

будет искать по всем полям в bibtex-записях, кроме полей bibtexEntryTag/bibtexEntryType/cn не являющихся потомками bibtexAttr.

Следует также заметить, что альтернативой типу Directory String является только синтаксис 1.3.6.1.4.1.1466.115.121.1.26 - строка в формате ASCII. Из этого следует вывод, что строки с национальными символами можно хранить только в кодировке UTF8, любая другая однобайтная кодировка просто “не влезет” в 7-битную ASCII строку, а в строку типа Directory String она не пройдёт из-за несоответствия типа. Поэтому с одной стороны все атрибуты в схеме ldap2bibtex объявлены как DirectoryString, а с другой, русским пользователям однобайтных кодировок, типа славной KOI8-R, необходимо пропускать ldif файлы через утилиту iconv с опциями -f UTF8 или -t UTF8 (в зависимости от направления передачи данных).

Приведенных сведений достаточно, чтобы в случае необходимости можно было расширить схему - добавить в нее новый атрибут или класс. Как уже говорилось,

[6]Хотя сервер OpenDS допускает в файлах схем перегрузку правил поиска для нечёткого равенства, хотя это и не закреплено стандартом.

никаких ограничений на содержимое bibtex-записей не накладывается, поэтому можно добавить в нее поле, например, lastseen - дату, когда эту книгу или статью последний раз просматривали. Для этой цели нужно добавить дополнительный атрибут `bibtexLastseen`, создав его по образцу имеющихся атрибутов и выбрав для него следующий свободный OID, а затем добавить `bibtexLastSeen` в список `MAY` класса `bibtexEntry`. Тем самым этот атрибут будет доступен во всех классах, потомках `bibtexEntry`.

5 Заполняем библиографический каталог.

Поскольку мы уже разобрались со схемой, используемой в каталоге ldap2bibtex, можно начать его использовать. Для начала неплохо было бы взять какой нибудь уже имеющийся bibtex файл или создать новый. В качестве примера можно использовать запись типа приведённой выше [4], которую поместим в файл temp.bib. Этот файл мы конвертируем в ldif формат с помощью утилиты bibtex2ldif из пакета ldap2bibtex, перекодируем с помощью системной утилиты `iconv` из системной локали в UTF8[7] и с помощью клиентской программы ldapadd передадим на сервер. Все это можно сделать одной строкой:

```
./bibtex2ldif -b "ou=Bib,o=Library" \\
    -i temp.bib | iconv -c -t UTF8 | ldapadd -c -h server.name
```

Существенным параметром для команды bibtex2ldif является параметр `-b` - базовый узел, в который будут помещены записи из .bib файла. По умолчанию входным и выходным файлом являются stdin и stdout соответственно, но значения по-умолчанию можно изменить с помощью ключей `-i` и `-o`. Полезно также посмотреть на результат команды bibtex2ldif:

```
dn: bibtexEntryTag=baldin06,ou=Bib,o=Library
objectClass: bibtexEntry
objectClass: bibtexarticle
bibtexEntryTag: baldin06
bibtexAuthor: {Е.Балдин}
bibtexTitle: {Латех -- компьютерная типография. Введение}
bibtexJournal: {LinuxFormat}
bibtexYear: 2006
bibtexVolume: 83
bibtexNumber: 9
```

Как видно, все поля перенесены один к одному из bib файла в ldif файл, единственно, что к уникальному имени `dn` добавилось значение параметра `-b` из командной строки. Параметры ldapadd аналогичны рассмотренным выше при создании корневого узла библиографической директории.

Данные переданные на сервер можно посмотреть с помощью команды ldapsearch, наподобие того как это было сделано выше, или воспользоваться второй (одноимённой) утилитой из пакета ldap2bibtex.

[7] Напомню, что конвертирование в UTF8 необходимо если вы собираетесь хранить в библиографической базе данных строки с национальными символами. Если же библиографическая запись содержит только ASCII символы, или используемая системная локаль является UTF8, то вызов iconv можно опустить. В дальнейшем на этом останавливаться не будем.

```
./ldap2bibtex -b"ou=Bib,o=hppi" -h server.name \\
              -F "(bibtexAuthor=*Балдин*)" | iconv -c -f UTF8
```

Эта программа представляет из себя клиентское приложение, которое соединяется с сервером LDAP и извлекает из директории данные, конвертируя их в bibtex формат. Поиск идёт по узлу задаваемом параметром `-b` причём `ldap2bibtex` всегда использует рекурсивный алгоритм поиска, а в качестве фильтра используется значение параметра `-F`, логически перемноженное с выражением `(objectClass=bibtexEntry)`. Таким образом программа `ldap2bibtex` на выходе всегда будет выдавать только библиографические записи. С помощью ключа `-W` запрашивается ввод пароля для пользователя с уникальным именем заданным параметром `-D`, причем в качестве его значения допустимо использовать как полную `dn` запись, так и одно имя пользователя.

Однако, более важной особенностью программы ldap2bibtex является умение извлекать ключи цитируемой литературы из tex-файлов, точнее из вспомогательных файлов с расширением aux, получающихся из исходного файла после первого запуска команды latex. Имя этого файла задаётся в качестве параметра `-a`. Таким образом, общая последовательность действий при создании текста в latex с использованием каталога LDAP выглядит достаточно просто. В текст вставляются ссылки на литературу с использованием команд `\cite` и ключей `bibtexEntryTag`, под которыми эти источники хранятся в каталоге, а потом просто запускается последовательность команд:

```
latex temp.tex
ldap2bibtex -b"ou=Bib,o=Library" -h server.name \\
            -a temp | iconv -c -f UTF8 > temp.bib
bibtex temp
latex
```

При этом предполагается, что исходный файл имеет название temp.tex и список литературы в нем генерируется командой `\bibliography{temp}`.

Для просмотра содержимого LDAP каталога (или добавления отдельных записей) можно использовать клиентское приложение с графическим интерфейсом GQ-client [22]. Настройка его достаточно очевидна и на ней я останавливаться не буду. На Рис. 2 показано как выглядит кусок библиографической базы данных в GQ-client и в популярном редакторе latex-файлов emacs (помимо многочисленных других его применений), после того как этот каталог был импортирован в bibtex-файл.

При использовании графических менеджеров типа gq обычно бывает удобнее не искать нужную запись с помощью фильтров, а пролистать каталог, типа того как с помощью файлового менеджера ищется нужный файл. Но для этой цели хотелось бы разбить все содержимое каталога на подкаталоги, т.е. рассортировать его содержимое. При этом как в таких случаях обычно оказывается, некоторые из записей подходят под несколько категорий. К примеру, можно представить, что рассмотренную нами запись [4] можно классифицировать как публикацию в журнале “LinuxFormat” так и отнести к категории справочной литературы по TeX. При этом так же как и в случае с файловой системой на помощь приходят “символьные ссылки”, роль которых играют объекты класса `alias`.

Рассмотрим эту ситуацию на примере. В каталоге `ou=Bib,o=Library` создадим два подкаталога класса `organizationalUnit`, а в каждый из этих подкаталогов положим по ссылке на уже созданный узел с уникальным именем `bibtexEntryTag=baldin06,ou=Bib,o=Library`. Для этого напишем ldif файл и загрузим его на сервер уже рассмотренным выше способом.

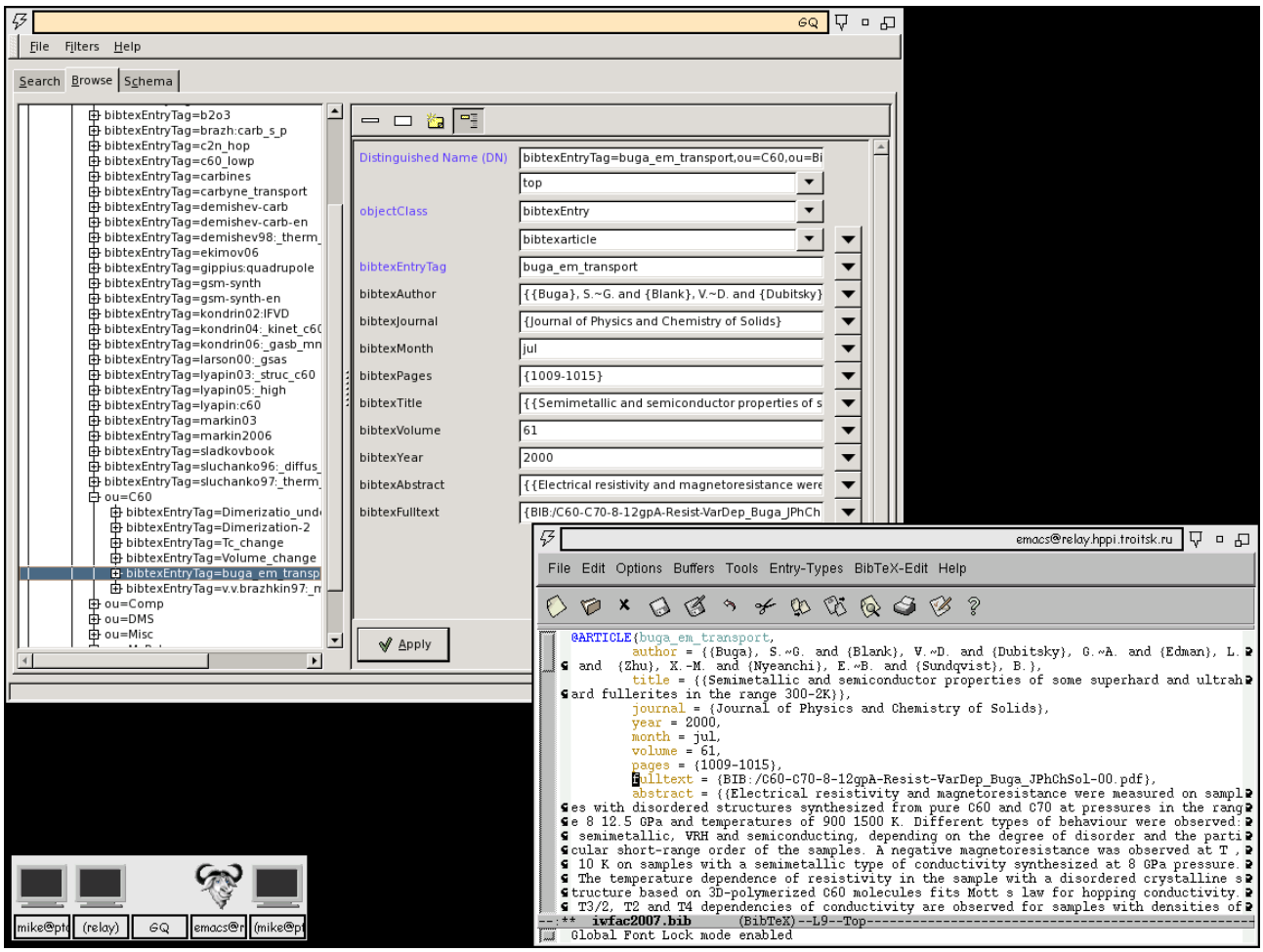

Рис. 2: Вид библиографического каталога в Emacs и GQ-client.

```
dn: ou=LF,ou=Bib,o=Library
ou: LF
objectClass: organizationalUnit

dn: ou=Tex,ou=Bib,o=Library
ou: Tex
objectClass: organizationalUnit

dn: ou="bibtexEntryTag=baldin06,ou=Bib,o=Library",\\
                                ou=LF,ou=Bib,o=Library
aliasedObjectName: bibtexEntryTag=baldin06,ou=Bib,o=Library
objectClass: alias

dn: uid=baldin06,ou=Tex,ou=Bib,o=Library
uid: baldin06
aliasedObjectName: bibtexEntryTag=baldin06,ou=Bib,o=Library
objectClass: uid
objectClass: alias
```

Специальный класс `alias` - это потомок класса `top` с единственным обязательным атрибутом `aliasedObjectName` - уникальным именем объекта, на который ссылается указатель. Если вас смущают длинные названия в `dn`, то лучше использовать вариант, наподобие того, что приведён во второй записи, с составным объектом и двойным наследованием - от `alias` и `uid`. Последний из них имеет тип `AUXILIARY` с единственным атрибутом – идентификатором `uid`. Печатать при этом получается ничуть не меньше, но зато это представляет хороший пример множественного наследования, которое было рассмотрено в предыдущем разделе.

Исключительность класса `alias` проявляется в том, что объекты такого типа обрабатываются клиентскими программами специальным образом. Результат поиска с помощью ldapsearch по каталогу, содержащему ссылки, зависит от параметра командной строки `-a` - способа разыменовывания указателей. По умолчанию, кстати, он указан как “never” - оставлять `alias` как есть, т.е. его необходимо поменять на “always” или “search”, чтобы вместо указателя подставлялся объект, на который он ссылается. У программы `ldap2bibtex` аналогичный параметр отсутствует, но её можно настроить с помощью редактирования конфигурационных файлов ldap.conf (с системными параметрами по умолчанию) и файла пользовательских настроек /.ldaprc[8]. Практически все параметры командной строки имеют свой аналог в этих конфигурационных файлах. Достаточно добавить в один из этих файлов строку `DEREF always`, чтобы практически любая клиентская программа, использующая библиотеку ldap, в том числе и `ldap2bibtex`, автоматически разыменовывала ссылки. Исключением является gq-client, но его авторы специально позаботились, чтобы “поправить” поведение по-умолчанию, и добавили в Preferences специальный пункт - использовать ли настройки ldap.conf/.ldaprc или настройки из XML-файла `.gq`. Первый вариант лучше, поскольку более гибкий. Тем не менее использование ссылок позволяет удобно рассортировать данные по подкаталогам без необходимости дублирования информации.

В bibtex есть интересная возможность “склеивания” данных из разных библиографических записей с помощью механизма crossref. Его идею проще пояснить на схеме (см Рис. 3). Предположим, мы хотим поместить в каталог серию книг, как,

[8] См. `man ldap.conf`

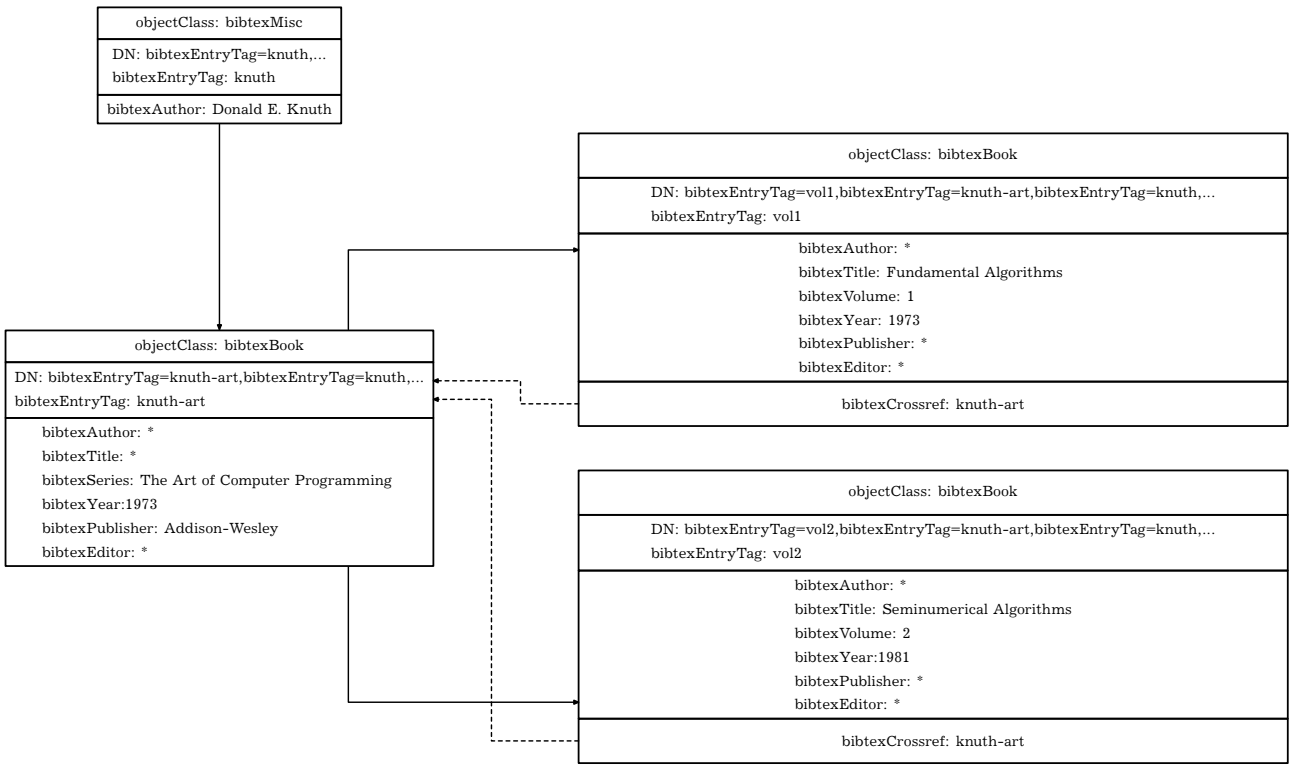

Рис. 3: Пример ”склеивания”bibtex-записей с помощью ldap2bibtex.

например, приведенное на схеме 4-х томное“Искусство программирования” Кнута. Чтобы не дублировать одни и те же выходные данные, имя автора, издательство и др. - то эту информацию можно сгруппировать в одной записи, а названия томов поместить в отдельные записи, в которые пимимо этого добавить специальное поле `crossref` с своего рода указателем или просто ключом первой bibtex-записи (на схеме эти зависимости показаны пунктирными стрелками). Хотя bibtex поддерживает только одно-уровневые crossref, но этого в большинстве случаев достаточно.

К сожалению, этот механизм плохо ложится на модель, применямую в ldap2bibtex. Вместо этого можно прийти к схожему результату, причем без ограничений на степень вложенности ссылок, используя иерархическое группирование в каталоге. Поскольку в LDAP любой объект может одновременно являться и контейнерем для других объектов, то ничего не мешает нам создать дерево из объектов, потомков bibtexEntry, наподобии того, что показано на Рис. 3 (иерархия в каталоге показана сплошными стрелками). Чтобы сослаться в тексте на подобные подкаталоги используется специальный синтаксис ссылок в latex-файле, согласно которому символ ’*’ в цитатах вида `\cite{vol1*knuth-art*knuth}` будет интерпретирован программой ldap2bibtex как разделитель bibtexEntryTag, принадлежащим трем разным объектам. Для их поиска последовательно используются три фильтра с расширенным синтаксисом (о фильтрах такого типа шла речь в конце раздела 3):

```
(&(bibtexEntryTag=vol1)(bibtexEntryTag:dn:=knuth-art)\\
                              (bibtexEntryTag:dn:=knuth))
(&(bibtexEntryTag=knuth-art)(bibtexEntryTag:dn:=knuth))
(bibtexEntryTag=knuth)
```

Заметьте, что первый фильтр выбирает объект с dn вида bibtexEntryTag=vol1,bibtexEntryTag=knuth-art,bibtexEntryTag=knuth, так что чередование ключей в команде \cite{...} повторяет их последовательность в уникальном имени.

Результат поиска по каждому из фильтров группируются в одну bibtex-запись. Понятно, что некоторые поля в получившейся записи могут быть многократно продублированы, но это не приводит к ошибкам, поскольку bibtex выдает предупреждение и использует первое из одноименных полей. К примеру, для ветки изображенной на Рис. 3 эта запись выглядит так (поле “year” встречается дважды):

```
@BOOK{vol1*knuth-art*knuth,
        title = {{Fundamental Algorithms}},
        volume = {1},
        year = {1973},
        publisher = {{Addison-Wesley}},
        series = {{The Art of Computer Programming}},
        year = {1973},
        author = {Donald E. Knuth}
}
```

Так что последовательность применения фильтров гарантирует, что атрибуты объекта, расположенного глубже в дереве каталога имеют приоритет над атрибутами родителя. Собственно, именно это нам и нужно. Так же ldap2bibtex игнорирует атрибуты, значение которых состоит из одного символа ’*’. Смысл этого правила, заключается в том, чтобы “обмануть” ldap-схему и фактически оставить незаполненным обязательный атрибут, например, в данном случае, пропускать имя автора (обязательное для объектов типа bibtexBook) в записях, соответствующих отдельному тому серии.

Как видно из приведенного примера, склеивание или merging записей, которое в определенном смысле обеспечивает ссылочную целостность информационной службы, в данном случае реализовано исключительно средствами клиента - программы ldap2bibtex. Можно сказать, что отсутствие “merging” на стороне сервера является одним из недостатков протокола LDAP. В тоже время интересно отметить, что в его прародителе (протоколе DAP) эта возможность была реализована автоматически средствами стека сетевых протоколов OSI, используемого в DAP. При миграции на более “лёгкий” стек TCP (откуда у LDAP появилась первая буква “L”) эта возможность была потеряна, но тем не менее в настояшее время предпринимаются попытки вернуть эту функциональность. Например, она оговаривается в RFC3671 (“Collective Attributes in the Lightweight Directory Access Protocol (LDAP)”). Хотя этот неформальный стандарт, насколько я знаю, ни в одном из серверов LDAP полностью не реализован, но в OpenLDAP есть демонстрационный модуль (“overlay” в терминологии OpenLDAP) - `slapo-collect`, который позволяет познакомиться с этими возможностями.

6 Административные вопросы и контроль доступа.

По мере заполнения и расширения библиографического каталога возникают проблемы с оптимизацией поиска по каталогу и предоставления пользователям доступа к нему. Пока мы использовали средства контроля доступа, встроенные по умолчанию в сервер OpenLDAP, так что любой, даже незарегистрированный, пользователь

может читать из каталога, а добавлять записи может только администратор сервера LDAP, который указан в качестве `rootdn` в конфигурационном файле. С одной стороны хотелось бы контроль доступа ужесточить, и дать возможность только зарегистрированным пользователям просматривать каталог (с тем чтобы анонимные злоумышленники не грузили сервер частыми и бессмысленными запросами), а с другой дать нескольким ответственным пользователям возможность пополнять каталог. Для этого на мой взгляд проще всего использовать возможности внешней аутентификации через SASL, появившиеся в последней версии протокола LDAPv3. Вообще-то говоря, использование SASL в последней версии протокола является рекомендуемым способом доступа к каталогам, а простая аутентификация сохранена исключительно из соображений совместимости.

Название SASL (Simple Authentication Security Layer) можно перевести на русский как прослойка простой аутентификации и сетевой безопасности. SASL представляет из себя как набор спецификаций, так и реализацию в виде библиотеки и встраиваемых модулей (например [13]), которые позволяют клиентам и серверам автоматически договориться об использовании определенного механизма аутентификации пользователя. Среди поддерживаемых механизмов - одноразовые пароли, Kerberos, использование SSL/TLS сертификатов и др. Поскольку SASL “встраивается” в сетевой протокол (в виде прослойки, откуда и название), то поддержка SASL должна быть оговорена в спецификациях сетевого протокола и реализована как на стороне клиента, так и сервера. Наиболее широко SASL используется в серверах электронной почты (pop3,imap,smtp) и в этом сегменте практически все современные продукты поддерживают ту или иную реализацию этой прослойки. Есть небольшое руководство на русском языке с описанием принципов работы и рекомендациями по настройке Cyrus-SASL [23].

Для работы с OpenLDAP проще всего (но это не значит, что лучше всего) настроить Cyrus-SASL для проверки паролей по файлу /etc/shadow. Для этого достаточно на стороне сервера запустить демон `saslauthd -a shadow` и отредактировать конфигурационный файл SASL `/usr/lib/sasl2/slapd.conf`, указав там список поддерживаемых протоколом LDAPv3 аутентификационных механизмов (состоящий из одного plain-text механизма) и способ проверки паролей (через демон saslauthd):

```
pwcheck_method:saslauthd
mech_list:plain
```

Следующим шагом нужно настроить контроль доступа пользователей к серверу LDAP. Для этого в конфигурационном файл slapd.conf нужно, во-первых, настроить правила преобразования аутентификационного имени в имя пользователя LDAP, а, во-вторых, собственно прописать ограничения на доступ к каталогу для пользователей.

Первая задача решается с помощью регулярных выражений, задаваемых в глобальной части конфигурационного файла:

```
sasl-secprops noanonymous
authz-regexp
        uid=([^,]*),cn=(plain|gssapi),cn=auth
        cn=$1,ou=People,o=Library
```

Имя пользователя это поле `uid`, следующее за ним `cn` - механизм используемый при аутентификации через SASL, так что это правило действует только для пользователей зарегистрированных через текстовые пароли или с помощью сертификатов Kerberos. Второй вариант оставлен, если вы решите настраивать SASL

следуя рекомендациям [23], в этом случае понадобится только отредактировать список механизмов в slapd.conf, а также дополнительно настроить принципала ldap/<hostname>@<REALM> и экспортировать его ключ в keytab-файл на сервере, где запущен slapd. С помощью параметра sasl-secprops существующий по умолчанию уровень безопасности ослаблен, с тем чтобы разрешить использование простых текстовых паролей[9]. Для отладочных целей крайне полезной является команда ldapwhoami, с помощью которой можно проверить под каким именем вы известны серверу LDAP (т.е. как выглядит ваше имя после применения всех регулярных подстановок).

```
#ldapwhoami -O "noanonymous" -v -U root  -Y plain -W
Enter LDAP Password:
ldap_initialize( <DEFAULT> )
SASL/PLAIN authentication started
SASL username: root
SASL SSF: 0
dn:cn=root,ou=bib,o=library
Result: Success (0)
```

Все консольные программы, входящие в состав OpenLDAP, имеют стандартизованный набор параметров. Чтобы каждый раз не набирать их в командной строке, удобно вынести их в конфигурационный файл ~/.ldaprc. В частности, можно убрать туда такие параметры как HOST/-h - имя LDAP сервера, или SASL_MECH/-Y и SASL_SECPROPS/-O - используемый механизм и уровень безопасности SASL, который на клиентской стороне также должен соответствовать уровню безопасности и списку разрешенных механизмов на стороне сервера.

Хотя параметры командной строки у программы ldap2bibtex несколько отличаются, но используемый ею механизм аутентификации SASL можно указать с помощью ключа -M. Остальные параметры регулирующие настройку SASL настраиваются только правкой пользовательского конфигурационного файла ~/.ldaprc.

После того как мы разобрались с SASL аутентификацией, то рекомендуется удалить запись rootpw из конфигурационного файла и приступить к настройке прав доступа к серверу LDAP.

```
access  to *
        by dn="cn=root,ou=People,o=Library" write
        by users read
access to dn.subtree="ou=Bib,o=Library"
        by users write
        by users read
```

В данном случае с помощью первого правил задан доступ по умолчанию к любым каталогам: на запись - только администратору, и на чтение - любым прошедшим проверку через SASL пользователям. Для ветки расположенной в узле ou=Bib,o=Library чтение и запись разрешены любым зарегистрированным пользователям. Синтаксис access to достаточно гибок, позволяющий тонкую настройку доступа пользователей к подкаталогам, например с помощью регулярных выражений dn.regex, но для решения этих вопросов лучше обратиться к онлайновой документации [10].

[9] См. man slapd.conf

[10] См. man slapd.access

Следующий вопрос - это индексация и оптимизации доступа к каталогам, что также решается правкой конфигурационного файла. Вообще-то, по моему скромному мнению, более правильно было бы выносить индексы вместе с соответствующей схемой в отдельный файл, но синтаксис конфигурационного файла не позволяет этого сделать. Действительно, загрузка схем происходит в глобальной секции конфигурационного файла, а индексация осуществляется в локальной секции, специфичной для конкретного каталога, т.е. после ключевого слова `database`. С другой стороны, такое разделение оставляет за администратором сервера LDAP решение по оптимизации объёма на диске, занимаемом индексами, и скоростью обращения к каталогу.

Хотя, размер библиографического каталогу не настолько велик, чтобы вопросы оптимизации были существенными, но на мой взгляд приемлемый уровень быстродействия можно получить используя следующие индексы, которые необходимо поместить в конце конфигурационного файла:

```
index   objectClass     eq

index   bibtexAuthor    approx,sub
index   bibtexTitle     approx,sub
index   bibtexKeywords  pres,approx,sub,eq
index   bibtexYear      eq
index   bibtexEntryTag  eq,sub,pres
```

Есть четыре типа индексации `eq,approx,sub,pres`, применение каждого из них зависит не только от желания администратора, но и от процедур поиска, определённых для индексируемого атрибута. Индексация `approx` работает для нечёткого равенства `~=`. `eq` оптимизирует поиск для фильтров типа точного равенства, не использующих шаблоны в правой части выражений сравнения, при этом для индексируемого атрибута должна быть определена операция `EQUALITY`. Для атрибутов индексируемых `sub` должна быть определена процедура поиска `SUBSTR`, и они оптимизируют фильтры типа точного равенства с шаблонами в правой части. Индекс `pres` проверяет наличие индексируемого атрибута и применимо для выражений с шаблонами специального вида, типа, `(bibtexEntryTag=*)`, используемых, например, с целью выборки из каталога только библиографических записей.

Использованные здесь индексы были выбраны из соображений наиболее частых применений. Поиск по авторам или названиям вряд ли будет производиться по точному соответствию (вместо того, чтобы вспоминать, стоят ли в заголовке инициалы автора или его полное имя, проще набрать одну фамилию с шаблонами в начале и конце). В то же время поиск по году издания, ключу, классу и ключевым словам вполне вероятно будет включать в себя точное соответствие. `objectClass` стоит проиндексировать для запросов типа точного равенства в обязательном порядке, поскольку считается хорошим тоном при любом поиске по каталогу ограничиваться определенными классы объектов.

7 Советы TEXникам.

В этом разделе будут даны несколько советов, которые возможно окажутся полезнымb в основном людям использующим TEX/LATEXдля написания текстов, и BibTeX для цитируемой литературы.

- Помимо стилевых файлов .bst, определяющих вид, в котором ссылка будет свёрстана в списке литературы, оформление ссылки в тексте зависит от дополнительных стилевых пакетов, таких как chicago или natbib. Какой из них применять, главным образом определяется редакционными требованиями. Для гуманитарных дисциплин более привычен Chicago-стиль (“первый автор-год”), а для естественнонаучных и технических работ - стиль natbib (числовые ссылки, пронумерованные по мере появления в тексте). Кроме того пакет natbib позволяет укорачивать ссылки, за счет удаления средних членов в непрерывной последовательности цитат, что тоже требуется в научных публикациях.

- Настоятельно рекомендуется установить пакет консольных утилит bibutils [24]. С помощью этого пакета можно конвертировать библиографические записи между различными форматами Bibtex/Endnote/RIS через промежуточный формат XML. Помимо того, что это бывает полезным и само по себе, некоторые онлайновые библиографические каталоги не экспортируют свои данные в формате bibtex. Пользоваться пакетом достаточно просто, имеющийся файл конвертируется с помощью одной утилиты в промежуточный файл, который затем преобразуется в bibtex-файл и по уже знакомой процедуре помещается в LDAP-каталог. Если же у вас аллергия на консольные утилиты, но нет аллергии на Java, то имеет смысл для той же цели воспользоваться уже упоминавшейся выше программой Jabref.

- ldap2bibtex можно использовать не только для хранения библиографических записей, но и для хранения интернет ссылок. Для этого крайне полезным будут latex-пакеты urlbst [25] и hyperref [26], которые позволяют форматировать поля title и url (или bibtexTitle и bibtexUrl) как гиперссылку. Пакет urlbst добавляет также дополнительную bibtex-запись типа webpage, которая также включена в ldap2bibtex схему, хотя для тех же целей допустимо использовать и запись типа misc.

- Совет негативного свойства. Поскольку bibtex приобрёл значительную популярность среди TEXников, то было несколько попыток использования его как своего рода базу данных для latex. Например, пакет figbib позволяет держать в bib-файлах ссылки на файлы с изображениями, а пакет directory - использовать bibtex в качестве адресной книги. К сожалению, по-разным причинам оба этих пакета не подходят для работы с ldap2bibtex. Вместо них предлагается использовать два стилевых файла imgbib.sty и dirbib.sty, входящие в пакет ldap2bibtex, которые предоставляют аналогичную функциональность.

- Если вас интересует лишь способ красивого вывода на печать информации, хранящейся в bib-файле, то следует обратить внимание на пакет bibentry, который вставляет полную библиографическую запись прямо в текст, а не в список литературы.

- Используемые мною библиографические стилевые файлы можно посмотреть на моей домашней странице [27].

8 Нерешённые проблемы.

К нерешённым можно отнести проблемы свойственные LDAP в целом, так и собственные проблемы пакета ldap2bibtex.

Я, скорее всего, во введении несколько оптимистично оценивал шансы LDAP как замену SQL баз данных для хранения информации. В первую очередь это связано с отсутствием у LDAP поддержки чисел с плавающей точкой, что следует из отсутствия стандарта на соответствующий синтаксис. В некоторых случаях можно обойтись и хранением чисел в виде строк или даже в бинарном формате, но вряд ли такое решение будет приемлемо.

Пакет ldap2bibtex как раз является удачным примером ещё и потому, что тут можно полностью обойтись без операций с дробными числами. К недостаткам же самого пакета можно отнести отсутствие поддержки со стороны приложений, где его использование было бы интересным. Например, emacs при работе с bib-файлами и LaTeXв режиме reftex позволяет автоматически искать записи в bib-файлах при наведении курсора на команду `\cite`. Хотелось бы иметь такие же возможности и при работе с библиографическим LDAP каталогом, тем более, что необходимая функциональность для взаимодействия с LDAP в emacs заложена.

Интересна была бы возможность использования библиографического LDAP каталога в текстовых процессорах, не связанных с LaTeX, например в OpenOffice. Для этой цели приходится действовать окружным путем, используя JDBC для доступа к библиографической базе и специальный JDBC-LDAP [28] драйвер для доступа к каталогу. Такой способ реализуем, но результат не слишком впечатляет. Кроме того разработчики OpenOffice в ближайшее время собираются серьезно переработать модуль работы с библиографией, так что, возможно, интерфейс этого модуля существенно поменяется.

Вместе с тем, насколько показывает опыт работы с библиографическим каталогом LDAP небольшой группой сотрудников, возможность использовать единую онлайновую базу данных во многом компенсирует неудобства, возникающие при редактировании текстов. Неплохой приз, особенно если учесть, что пакет ldap2bibtex помог нам разобраться с секретами создания информационно-справочных служб!

Список литературы

[1] Домашняя страница Pierangelo Masarati. Pierangelo Masarati Home page.

[2] ldap2bibtex - старая версия. Original version Ldap2bibtex.

[3] Новая версия ldap2bibtex. New version ldap2bibtex.

[4] Е.Балдин. Латех – компьютерная типография. Введение. *LinuxFormat*, 83(9), 2006.

[5] Tkbibtex - графический интерфейс для редактирования bibtex файлов.

[6] JabRef - домашняя страница.

[7] BibTeX-Mode for GNU-Emacs.

[8] Система для хранения библиографических ссылок в базе данных MySQL/SqlLite. Bibus-Biblio database.

[9] Библиографическая база данных MySQL/PostgreSQL/SqlLite. RefDB bibliography database.

[10] Apache directory server. Alternative LDAP Sever - Apache Directory Server. LDAPv3 sertified!

[11] SUN Open Directory Server. Alternative LDAP Sever - SUN Open Directory Server.

[12] OpenLDAP release. Last version of openldap.

[13] ftp://ftp.andrew.cmu.edu/pub/cyrus-mail/cyrus-sasl-2.1.22.tar.gz. Cyrus SASL Last version.

[14] http://www.oracle.com/technology/software/products/berkeley-db/index.html. Berkley DB download center from Oracle.

[15] Iana - Организация резервирования интеренет адресов. Get the free OID 1.3.6.1.4.1.x from IANA for free!

[16] Поиск уже зарезервированных OID. Search the tree of assigned OID's.

[17] Pierangelo Masarati. Программа для расчета динамики составных тел. MBDyn - MultiBody Dynamics analysis system.

[18] Регистрация организационных имен ANSI.

[19] OpenLDAP Software 2.3 Administrator's Guide. Schema Specification.

[20] Understanding Schema in OpenDS. Understanding Schema in OpenDS.

[21] Michael Donnelly. Customizing your LDAP Directory Schema. Part One: Understanding LDAP Attributes. Customizing your LDAP Directory Schema Part One: Understanding LDAP Attributes.

[22] GQ LDAP-client.

[23] М.Кондрин. Как настроить библиотеку SASL для совместной работы с Kerberos. *Системный Администратор*, 10, 2006.

[24] BibUtils home page.

[25] Urlbst - гиперлинки в bib-файлах.

[26] Hyperref - гипрелинки в latex.

[27] Домашняя страница автора.

[28] JDBC-LDAP Bridge Driver.