Впервые опубликовано на developerWorks 11.03.2008

Подготовка к экзамену LPI 301: Тема 302. Установка и разработка

Профессионал Linux высокого уровня (LPIC-3)

Компиляция и установка OpenLDAP

В этом разделе описывается материал по теме 302.1 экзамена на профессионала Linux высокого уровня (LPIC-3) 301. Эта тема обладает весом 3.

Из этого раздела вы узнаете, как:

- Компилировать и настраивать OpenLDAP из исходного кода
- Узнаете о базах данных бэк-энда OpenLDAP
- Управлять демонами OpenLDAP
- Устранять ошибки, возникающие во время установки

OpenLDAP - это приложение с открытым исходным кодом, реализующее сервер LDAP и связанные с ним инструменты. Поскольку это приложение с открытым исходным кодом, вы можете загрузить его исходный код бесплатно. Проект OpenLDAP не распространяет двоичный код, но большинство основных дистрибутивов делают пакеты самостоятельно. Из этого руководства вы узнаете, как устанавливать OpenLDAP и из исходного кода, и из пакетов.

Компиляция из исходного кода

Сначала нужно загрузить последнюю версию OpenLDAP с сайта проекта (ссылку для скачивания можно найти в разделе Ресурсы). У этого проекта обычно есть две актуальные версии: одна стабильная и одна тестовая. Это руководство было написано с использованием стабильных версий 2.3.30 и 2.3.38. Если вы будете выполнять приведенные здесь примеры, обратите внимание, что названия директорий могут различаться в зависимости от используемой вами версии.

Чтобы извлечь исходный код из загруженного архива, введите: `tar -xzf openldap-stable-20070831.tgz`. Эта команда разархивирует загруженные файлы в папку. Войдите в эту новую папку с помощью команды `cd openldap-2.3.38` (подставив свою версию OpenLDAP).

Теперь вы находитесь в директории с исходным кодом. Теперь вам нужно настроить и собрать среду для вашей системы, после чего собрать программное обеспечение. Для выполнения этих действий в OpenLDAP используется сценарий `configure`. Введите `./configure --help` для просмотра доступных параметров. Некоторые из них определяют, куда будут устанавливаться файлы (например, `--prefix`); , другие определяют функции OpenLDAP, которые будут доступны в сборке. В листинге 1 перечислены функции и значения, присваиваемые по умолчанию.

Листинг 1. Параметры настройки, относящиеся к функциям OpenLDAP

```
SLAPD (Standalone LDAP Daemon) Options:

  --enable-slapd          enable building slapd [yes]
    --enable-aci          enable per-object ACIs (experimental) [no]
    --enable-cleartext    enable cleartext passwords [yes]
    --enable-crypt        enable crypt(3) passwords [no]
    --enable-lmpasswd     enable LAN Manager passwords [no]
    --enable-spasswd      enable (Cyrus) SASL password verification [no]
    --enable-modules      enable dynamic module support [no]
    --enable-rewrite      enable DN rewriting in back-ldap and rwm overlay
[auto]
    --enable-rlookups     enable reverse lookups of client hostnames [no]
    --enable-slapi        enable SLAPI support (experimental) [no]
    --enable-slp          enable SLPv2 support [no]
    --enable-wrappers     enable tcp wrapper support [no]

SLAPD Backend Options:
    --enable-backends     enable all available backends no|yes|mod
    --enable-bdb          enable Berkeley DB backend no|yes|mod [yes]
    --enable-dnssrv       enable dnssrv backend no|yes|mod [no]
    --enable-hdb          enable Hierarchical DB backend no|yes|mod [yes]
    --enable-ldap         enable ldap backend no|yes|mod [no]
    --enable-ldbm         enable ldbm backend no|yes|mod [no]
      --enable-ldbm-api   use LDBM API auto|berkeley|bcompat|mdbm|gdbm [auto]
      --enable-ldbm-type  use LDBM type auto|btree|hash [auto]
    --enable-meta         enable metadirectory backend no|yes|mod [no]
    --enable-monitor      enable monitor backend no|yes|mod [yes]
    --enable-null         enable null backend no|yes|mod [no]
    --enable-passwd       enable passwd backend no|yes|mod [no]
    --enable-perl         enable perl backend no|yes|mod [no]
    --enable-relay        enable relay backend no|yes|mod [yes]
    --enable-shell        enable shell backend no|yes|mod [no]
    --enable-sql          enable sql backend no|yes|mod [no]

SLAPD Overlay Options:
    --enable-overlays     enable all available overlays no|yes|mod
    --enable-accesslog    In-Directory Access Logging overlay no|yes|mod [no]
    --enable-auditlog     Audit Logging overlay no|yes|mod [no]
    --enable-denyop       Deny Operation overlay no|yes|mod [no]
    --enable-dyngroup     Dynamic Group overlay no|yes|mod [no]
    --enable-dynlist      Dynamic List overlay no|yes|mod [no]
    --enable-lastmod      Last Modification overlay no|yes|mod [no]
    --enable-ppolicy      Password Policy overlay no|yes|mod [no]
    --enable-proxycache   Proxy Cache overlay no|yes|mod [no]
    --enable-refint       Referential Integrity overlay no|yes|mod [no]
    --enable-retcode      Return Code testing overlay no|yes|mod [no]
    --enable-rwm          Rewrite/Remap overlay no|yes|mod [no]
    --enable-syncprov     Syncrepl Provider overlay no|yes|mod [yes]
    --enable-translucent  Translucent Proxy overlay no|yes|mod [no]
    --enable-unique       Attribute Uniqueness overlay no|yes|mod [no]
    --enable-valsort      Value Sorting overlay no|yes|mod [no]

SLURPD (Replication Daemon) Options:
  --enable-slurpd         enable building slurpd [auto]

Optional Packages:
  --with-PACKAGE[=ARG]    use PACKAGE [ARG=yes]
  --without-PACKAGE       do not use PACKAGE (same as --with-PACKAGE=no)
  --with-subdir=DIR       change default subdirectory used for installs
  --with-cyrus-sasl       with Cyrus SASL support [auto]
  --with-fetch            with fetch(3) URL support [auto]
```

```
  --with-threads          with threads [auto]
  --with-tls              with TLS/SSL support [auto]
  --with-yielding-select  with implicitly yielding select [auto]
  --with-odbc             with specific ODBC support iodbc|unixodbc|auto [auto]

  --with-gnu-ld           assume the C compiler uses GNU ld [default=no]
  --with-pic              try to use only PIC/non-PIC objects [default=use
                          both]
  --with-tags[=TAGS]
                          include additional configurations [automatic]
```

В листинге 1 вы можете видеть, что по умолчанию многие функции выключены, например, метакаталоги и модули. Кроме того, многие параметры отмечены как "auto", что означает, что функция включается, если в системе установлены нужные библиотеки. Вместо того чтобы полагаться на такое автоматическое поведение, лучше составить список нужных функций и включить их. Если каких-то библиотек не хватает, то вы получите ошибку во время компиляции, а не когда-либо позже.

Некоторым конфигурационным опциям могут быть переданы значения no, yes или mod. no отключает опцию, yes статически привязывает опцию к конечному двоичному файлу, а mod собирает опцию в отдельной библиотеке общего пользования. Библиотеки общего пользования загружаются на сервер во время работы. По умолчанию модули привязываются статически; то есть они являются частью двоичного кода и неотделимы от него. Если вы хотите использовать динамические модули, вам также нужно использовать параметр `--enable-modules`. Преимущество динамических модулей состоит в том, что вы можете проверить различные опции, не раздувая чрезмерно двоичный файл, и поставлять модули отдельно.

В листинге 2 показана строка конфигурации, применяемая в Fedora 7, которая включает многие полезные функции. В большинстве своём выбранные опции позволят использовать функции, которые будут нужны в следующих руководствах, например `--enable-slurpd` и `--enable-multimaster`для репликации, и `--enable-meta` для метакаталогов. Другие опции включают различные бэк-энды, например, ldab, bdb, null и monitor.

Листинг 2. Пример конфигурации файла

```
./configure --enable-plugins --enable-modules --enable-slapd --enable-slurpd \
   --enable-multimaster --enable-bdb --enable-hdb --enable-ldap --enable-ldbm \
   --enable-ldbm-api=berkeley --enable-meta --enable-monitor --enable-null \
   --enable-shell --enable-sql=mod --disable-perl \
   --with-kerberos=k5only --enable-overlays=mod --prefix=/tmp/openldap
```

В листинге 2 включаются дополнительные модули и несколько бэк-эндов, включая бэк-энды, построенные на языке SQL и файлы базы данных Berkeley Database. Бэк-энды являются способом хранения и получения данных, используемым OpenLDAP.

В листинге 2 также собираются демон автономной работы `slapd` и демон репликации `slurpd`. Также в целях тестирования включаются оверлеи, которые облегчают настройку данных бэк-энда. Поскольку установка является тестовой, префикс изменен на `/tmp/openldap`, поэтому результирующий двоичный код будет расположен в папке `/tmp/openldap/libexec`.

При выполнении сценарий `configure` проверяет наличие необходимых библиотек и

создаёт среду сборки. Если выполнение `configure` завершается успешно, скомпилируйте OpenLDAP, выполнив `make depend; make`.

После завершения успешной компиляции кода вы можете установить OpenLDAP, выполнив команду `make install`. Она скопирует весь двоичный код, страницы man и библиотеки в соответствующие места в `/tmp/openldap`.

Установка из пакетов

Если предыдущий раздел о компиляции из исходного кода напугал вас, вы не одиноки. Компиляция из исходного кода - процесс трудоёмкий и может быть осложнён отсутствием нужных библиотек. Если у вас недостаточно большой опыт разработки на C, вам может быть сложно интерпретировать ошибки, появляющиеся в процессе сборки. К счастью, в большинстве дистрибутивов пакет OpenLDAP поставляется в виде готового двоичного кода с предварительно настроенной конфигурацией. Обычно в этих пакетах есть все необходимые функции.

Дистрибутивы, построенные на RPM

Для загрузки с сервера и установки пакетов RedHat (RPM) в системах Fedora и CentOS используется инструмент `yum`. Команда `yum list` позволит узнать, какие пакеты доступны, необязательное регулярное выражение позволит отфильтровать список пакетов. В листинге 3 показан поиск всех пакетов, содержащих слово `openldap`.

Листинг 3. Определение доступных пакетов с помощью `yum`

```
# yum list \*openldap\*
Loading "installonlyn" plugin
Setting up repositories
Reading repository metadata in from local files
Installed Packages
openldap.i386                           2.3.30-2.fc6           installed
openldap-clients.i386                   2.3.30-2.fc6           installed
openldap-devel.i386                     2.3.30-2.fc6           installed
openldap-servers.i386                   2.3.30-2.fc6           installed
openldap-servers-sql.i386               2.3.30-2.fc6           installed
Available Packages
compat-openldap.i386                    2.3.30_2.229-2.fc6    updates
```

В таких больших приложениях, как OpenLDAP, клиентская и серверная часть обычно разбиваются на два отдельных пакета. Также вы можете найти несколько библиотек, обеспечивающих совместимость (чтобы могли работать приложения, связанные со значительно более старыми версиями данного программного обеспечения). Чтобы установить пакет, выполните команду `yum install`, указав название пакета, например, `yum install openldap-clients openldap-servers`; в результате будут загружены и установлены и клиентский, и серверный пакеты со всеми зависимостями.

В Red Hat Enterprise Linux команда поиска пакетов `openldap` будет иметь вид `up2date --showall | grep openldap`. Для установки пакетов укажите в качестве аргумента up2date их названия, например, `up2date openldap-clients openldap-servers`.

Для того, чтобы убедиться в том, что сервер OpenLDAP запускается при загрузке системы,

выполните команду `chkconfig ldap on`.

Дистрибутивы, построенные на базе Debian

В дистрибутивах, построенных на базе Debian, например, Ubuntu, для установки пакетов используются инструменты Advanced Packaging (APT). Сначала выполните команду `apt-cache search openldap` для поиска пакетов OpenLDAP, как показано в листинге 4.

Листинг 4. Перечень доступных пакетов OpenLDAP в Ubuntu Linux

```
notroot@ubuntu:~$ apt-cache search openldap
libldap2 - OpenLDAP libraries
libldap2-dev - OpenLDAP development libraries
python-ldap - A LDAP interface module for Python. [dummy package]
python-ldap-doc - Documentation for the Python LDAP interface module
python2.4-ldap - A LDAP interface module for Python 2.4
ldap-utils - OpenLDAP utilities
libldap-2.2-7 - OpenLDAP libraries
slapd - OpenLDAP server (slapd)
```

В листинге 4 показано несколько доступных пакетов. Сервер реализован в пакете `slapd`, а все необходимые зависимости будут загружаться в процессе установки. Выполните команду `sudo apt-get install slapd` для установки сервера. Также будет полезно включить пакет `ldap-utils`, содержащий клиент, работающий в командной строке.

Настройка программного обеспечения

После установки OpenLDAP его необходимо настроить. Для тестирования достаточно указать всего несколько параметров; но в реальных условиях (и на экзамене LPIC 3) вы должны хорошо разбираться во всевозможных параметрах.

Работой OpenLDAP управляют два конфигурационных файла; по умолчанию оба располагаются в папке /etc/openldap/. Первый файл, ldap.conf, управляет общим поведением клиентов LDAP. Конфигурационный файл для серверов LDAP называется slapd.conf. Несмотря на название, в slapd.conf также содержится конфигурация для `slurpd`, демона репликации. В этой статье основное внимание уделяется slapd.conf, в частности, его разделу, касающемуся демона `slapd`.

Формат slapd.conf очень прост, за одним ключевым словом следует один или несколько аргументов, на которые распространяются следующие условия:

- Ключевое слово должно начинаться в нулевой колонке — перед ним не должно быть пробелов.
- Если в аргументе содержатся пробелы, его необходимо заключить в двойные кавычки ("").
- Если строка начинается с пробела, она считается продолжением предыдущей строки.
- Регистр ключевых слов значения не имеет, но он может быть важен для аргументов, в зависимости от того, какие ключевые слова используются.

Как и в большинстве инструментов UNIX®, символ решетки (#) обозначает комментарий. Всё, что написано после решетки, игнорируется.

Файл slapd.conf разделен на две части: глобальные параметры и параметры базы данных бэк-энда. Хотя порядок указания директив жёстко не задаётся, выбирать место для новых директив следует с осторожностью, поскольку некоторые из них могут изменять порядок обработки последующих директив. Например, если было указано ключевое слово `backend` или `database` , параметр считается глобальным. После того, как будет прочтена директива `database`, все остальные параметры будут относиться к базе данных. Это будет продолжаться до тех пор, пока не будет считана еще одна директива `database`, после которой следующие директивы будут применяться к новой базе данных.

Некоторые глобальные параметры будут рассматриваться в следующих руководствах серии 301, например, посвящённых управлению доступом и репликации. Ниже приведено описание наиболее часто используемых директив.

Параметры сервера (глобальные)

Некоторые параметры сервера ограничивают работу, выполняемую процессом `slapd` , что позволяет предотвратить недостаток ресурсов. `conn_max_pending` принимает целое число, определяющее количество анонимных запросов, которые могут находиться в очереди в любой момент времени. Более подробно с механизмом связывания сервера LDAP вы познакомитесь в следующих руководствах серии 301; если говорить просто, вы можете подавать запросы серверу после ввода имени пользователя и пароля (аутентифицированный сеанс) или без имени и пароля (анонимный сеанс). Запросы, превышающие предел, установленный `conn_max_pending`, будут отбрасываться сервером. Параметр `conn_max_pending_auth` работает так же, как и `conn_max_pending`, но ограничивает аутентифицированные сессии.

Параметр `idletimeout` (в секундах) сообщает `slapd`, как долго будут удерживаться неактивные клиенты прежде, чем будут отключены. Если этому параметру установлено значение 0, отключение не производится.

Параметр `sizelimit` ограничивает количество результатов поиска в одном запросе, а `timelimit` ограничивает время, затрачиваемое сервером на поиск. Эти параметры могут принимать целочисленные значения, ключевое слово `unlimited` или более сложные жёсткие и мягкие ограничения. По умолчанию устанавливается мягкое ограничение по времени и по количеству результатов, однако, если клиент запросил большее число строк или большее время, может быть применено жёсткое ограничение. Например, `sizelimit sizesoft=400 size.hard=1000` указывает, что по умолчанию возвращается 400 строк. Клиент может подать запрос, увеличивающий этот предел до 1 000. Такой формат может быть применен к группам пользователей, позволяя одним пользователям или приложениям выполнять большие запросы, а другим - только маленькие

Когда клиент выполняет поиск по дереву, он обычно указывает узел (который называется *базой поиска* или просто *базой*), с которого будет начинаться поиск—в нем будут находиться базы поиска всех отличительных имен (Distinguished Names, DN) результатов. Это позволяет ускорить поиск (поскольку теперь необходимо просматривать меньшее количество узлов) и упростить реализацию клиента (поскольку поиск только по части дерева - простой, но эффективный фильтр). Если клиент не указывает конкретную базу, используется значение `defaultsearchbase`. Рекомендуется устанавливать значение этого параметра, чтобы избежать возможных проблем с неверно настроенными клиентами. В зависимости от структуры используемого вами дерева LDAP будет уместно использовать либо контейнер пользователя, либо корень дерева.

Различные функции, которые будет поддерживать сервер, например, поддержка ранних версий и требования клиентов к безопасности, определяются тремя командами. Это команды `allow`, `disallow` и `require`. После каждой команды указывается ряд ключевых слов с пробелами, которые включают, отключают функцию или делают её обязательной. Ключевые слова показаны в таблице 3.

Таблица 3. Ключевые слова, используемые с `allow`, `disallow` и `require`

Команда(ы)	Ключевое слово	Описание	Значение по умолчанию
`allow`	`bind_v2`	Установка функции позволяет подключаться клиентам, использующим устаревшую версию протокола LDAPv2. В документации по OpenLDAP постоянно подчёркивается, что OpenLDAP не предоставляет полной поддержки LDAPv2, поэтому такие запросы могут привести к непредвиденным результатам.	Запрещено
`allow`	`bind_anon_cred`	Позволяет клиенту связывание с паролем, но без DN. Если включен этот параметр, клиент может выполнять анонимное связывание.	Запрещено
`allow`	`bind_anon_dn`	Позволяет клиенту выполнять связывание с DN, но без пароля, обычно это связано с неправильной конфигурацией клиента. Если включен этот параметр, клиент может выполнять анонимное связывание.	Запрещено
`allow, disallow`	`update_anon`	Позволяет выполнять связывание анонимно, что происходит, когда клиент подключается к серверу LDAP без DN и пароля.	Разрешено
`disallow`	`bind_simple`	Позволяет выполнять простую аутентификацию (передача имени пользователя и пароля в незашифрованном виде) вместо более защищенных методов, например Simple Authentication and Security Layer (SASL).	Разрешено
`require`	`bind`	Обязывает всех клиентов связываться с каталогом с помощью операции **bind** перед выполнением каждой операции.	Не требуется
`require`	`LDAPv3`	Определяет необходимость использования протокола LDAPv3. Обратите внимание, что эта команда может конфликтовать с allow bind_v2.	Не требуется
`require`	`authc`	Требует аутентификации, в противоположность анонимному связыванию.	Не требуется
`require`	`SASL`	Обязывает использовать метод SASL для подключения к серверу	Не требуется
`require`	`strong`	Обязывает использовать безопасный метод аутентификации. Это может быть либо SASL, либо простая аутентификация по защищённому каналу.	Не требуется

`require none`	Эта опция снимает все требования, обычно используется в случае, если нужно ослабить требования определенной базы данных, указывая эту команду в разделе базы данных файла slapd.conf. Если нужно не очистить список требований для базы данных, а изменить их, необходимо указать none перед добавлением новых требований, даже если эти требования уже указаны в глобальном разделе.	Не применимо

Несмотря на то, что некоторые команды из таблицы 3 определяют тип входа в систему, соединение всё ещё подчиняется правилам управления доступом. Например, при анонимном входе к части дерева может быть предоставлен доступ только для чтения. Включение или выключение различных методов аутентификации определяется природой вашего приложения и возможностями ваших клиентов.

Если вы хотите поддерживать высокий уровень готовности, включите `gentlehup`. При включении этой команды `slapd` перестаёт прослушивать сеть после получения сигнала `SIGHUP`, но не отключает открытые соединения. После этого может быть запущен новый экземпляр `slapd`, как правило, с обновлённой конфигурацией.

Для того, чтобы получить более подробный журнал, измените значение `loglevel`. В качестве параметра этой команды указывается целое число, несколько целых чисел или ряд ключевых слов, которые включают ведение журнала для определенной функции. Полный список ключевых слов и значений можно найти в страницах справки slapd.conf. Например, отслеживанию соединения соответствуют значение 8 и ключевое слово conns, а синхронизации - значение 4096 и ключевое слово `sync`. Для того, чтобы включить ведение журнала по двум этим позициям, нужно выполнить `logging 5004`, `logging 8 4096` или `logging conns sync`, результат будет одинаковым.

Если вы скомпилировали OpenLDAP из исходного кода, вы, возможно, уже включили некоторые модули. Также, возможно, вы загрузили дополнительные модули из менеджера пакетов, например, пакет `openldap-server-sql` содержит модуль бэк-энда SQL. Параметры `modulepath` и `moduleload` используются для загрузки динамических модулей в `slapd`. `modulepath` указывает директорию (или перечень директорий), содержащих общие библиотеки, и каждый экземпляр `moduleload` указывает загружаемый модуль. Версию модуля и расширение указывать не нужно, поскольку `slapd` ищет общие библиотеки. Например, для библиотеки `back_sql-2.3.so.0.2.18` нужно указать `moduleload back_sql`. Также можно указать в `moduleload` полный путь к библиотеке (без версии и расширения), например `moduleload /usr/share/openldap/back_sql`.

Некоторым сценариям нужно, чтобы идентификатор процесса хранился в определенном файле. `pidfile` говорит `slapd`, куда записывать свой идентификатор процесса.

Параметры схемы

Несколько команд позволяют вам добавить к дереву элементы схемы, либо посредством включения файла схемы, либо путём определения объектов в slapd.conf.

Для того, чтобы добавить новый файл схемы на сервер, используйте команду `include`, после которой укажите полный путь к файлу схемы (обычно они располагаются в /etc/openldap/schema). Если одна схема ссылается на другую (например, `inetOrgPerson` наследуется из `organizationalPerson`), нужно указать все необходимые файлы в правильном порядке, сначала включая базовые объекты. OpenLDAP обрабатывает файлы схемы в порядке их включения, поэтому этот порядок важен.

Новые элементы схемы можно добавить напрямую через slapd.conf с помощью команд `attributetype` и `objectclass` для классов атрибутов и объектов соответственно. Этот способ аналогичен указанию информации в файле схемы и включению его с помощью команды `include`. Подобным же образом можно определить и идентификаторы объектов (OID), используя `objectidentifier`.

Бэк-энды и базы данных

Бэк-энды и базы данных - это два отдельных, но очень тесно связанных между собой понятия. База данных представляет собой часть дерева, например, `dc=ertw, dc=com`. Бэк-энд описывает метод, используемый `slapd` для доступа к данным. (Дерево `dc=ertw, dc=com` было первым примером в этой серии.)

Во многих случаях бэк-энд является файлом на диске (в некотором формате, более подробно он будет рассматриваться позже); также он может быть методом для получения данных из другого источника, из базы данных SQL, из DNS, или даже с помощью сценария. Каждая база данных обрабатывается одним бэк-эндом, а один и тот же бэк-энд может использоваться несколькими базами данных.

Как уже было отмечено, slapd.conf начинается с глобальных директив. Режим бэк-энда начинается с первого экземпляра директивы `backend` . Все директивы в этом режиме применяются к текущему бэк-энду. Все установленные глобально параметры применяются к бэк-энду, если они не переопределяются затем на уровне бэк-энда. Подобным же образом базы данных настраиваются в рамках, определяемым ключевым словом `database`. База данных привязывается к типу бэк-энда и наследует все глобальные параметры и параметры уровня бэк-энда. Вы также можете переопределить все параметры на уровне базы данных.

OpenLDAP разделяет бэк-энды на три типа:

1. Те, которых хранят данные:
    - bdb—Использует механизм баз данных Berkeley (например, Sleepycat, сейчас принадлежит Oracle)
    - hdb—Улучшенная версия back-ldb, в которой усовершенствована индексация
2. Те, которых передают данные:
    - ldap—Передаёт данные другого сервера LDAP
    - meta—Передаёт данные различных серверов LDAP для различных частей дерева
    - sql—Возвращает данные из базы SQL
3. Те, которых создают данные:
    - Dnssrv—Возвращает внешние ссылки (referral) LDAP на основании данных в записи DNS SRV
    - monitor—Возвращает статистику сервера LDAP
    - null—Тестовый модуль, ничего не возвращает
    - passwd—Возвращает данные из файла паролей
    - perl—Возвращает данные, сформированные сценарием Perl

- shell—Возвращает данные, сформированные сценарием shell

Параметры конфигурации различны для каждого бэк-энда, их можно найти в соответствующих страницах справки (например, `slapd-bdb` для бэк-энда bdb).

Базы данных представляют дерево и содержащиеся в нём данные. Примером базы данных является дерево `dc=ertw,dc=com`. Все данные в этом DN будут храниться так же, как если бы они являлись частью одной базы данных. Кроме того, существует возможность хранения `ou=people,dc=ertw,dc=com` в одной базе данных, а все, что располагается под `dc=ertw,dc=com` - в другой. И, наконец, сервер LDAP может обрабатывать несколько деревьев, например, `dc=ertw,dc=com` и `dc=lpi,dc=org`. У каждой базы данных есть собственный способ обработки запросов посредством собственного бэк-энда.

Для начала режима конфигурирования базы данных укажите `database`, а затем - тип базы данных. Чаще всего используется база данных Berkeley, поэтому `database bdb` создаёт базу данных BDB. Следующая команда - `suffix`, которая указывает корень дерева обслуживаемой базы данных.

`rootdn` и `rootpw` позволяет вам указывать пользователя со всеми привилегиями (*корневой пользователь*) базы данных. К этому пользователю не применяются правила контроля доступа. `rootdn` должен находиться в рамках указанного суффикса и может содержать или не содержать пароль. Будет использоваться пароль, указанный в `rootpw`. Если этот параметр не указан, будет выполняться поиск записи `rootdn` в дереве и будет выполняться аутентификация по атрибуту `userPassword`. Если корневой пользователь не указан, то заданные правила контроля доступа будут применяться ко всем пользователям.

Если вы указываете `lastmod on`, OpenLDAP хранит несколько скрытых атрибутов (называемых *операционными атрибутами*), например, имя лица, создавшего запись, и когда она была изменена. Некоторые из этих атрибутов необходимы для репликации, поэтому будет разумно оставить параметр `lastmod` включенным (как установлено по умолчанию). Эти операционные атрибуты не показываются клиентам, если их не запросить специально.

Также с помощью команды `restrict` можно ограничить действия, которые можно выполнять с базой данных. В качестве параметров этой команды указываются параметры, соответствующие операциям LDAP, например `add`, `bind`, `compare`, `delete`, `rename` или `search`. Для того, чтобы запретить пользователям удалять узлы дерева, используется команда `restrict delete`. Если в дереве содержатся пользователи, но по каким-то причинам вы не хотите давать им возможность подключения к дереву, используйте `restrict bind`. Кроме того, доступны параметры `read` и `write`, которые не запрещают все действия, а блокируют чтение и запись в дерево соответственно. Также вы можете использовать команду `readonly` и сделать базу данных доступной только для чтения.

Различные элементы дерева могут содержаться в различных базах данных. При правильной настройке OpenLDAP объединяет вместе все части. База данных, содержащая в себе другие базы, называется *главной базой данных*; базы данных, содержащиеся в ней, называются *зависимыми базами данных*. Сначала определите зависимую базу данных и добавьте в соответствующую ей строку команду `subordinate`. После этого определите главную базу данных. В такой конфигурации OpenLDAP может работать с несколькими базами данных как с одной, при этом часть данных будет храниться локально, а другая часть запрашивается из других источников (в особом случае, когда все данные находятся на удалённых серверах LDAP, используется метакаталог). Обратите внимание, что если вы сначала определите главную базу данных, а потом зависимую, вы получите сообщение об ошибке – попытке

переопределить часть дерева. В листинге 5 показано дерево dc=ertw, dc=com, разбитое на главную и зависимую базы данных.

Листинг 5. Настройка главной и зависимой базы данных

```
# Subordinate
database bdb
suffix "ou=people,dc=ertw, dc=com"
rootdn "cn=Sean Walberg,ou=people,dc=ertw,dc=com"
rootpw mysecret
directory       /var/db/openldap/ertw-com-people
subordinate

# Superior
database bdb
suffix "dc=ertw, dc=com"
rootdn "cn=Sean Walberg,dc=ertw,dc=com"
rootpw mysecret
directory       /var/db/openldap/ertw-com
```

Также обратите внимание, что здесь настраиваются два rootdn. Если вы хотите установить пароль, в базе должен быть определен rootdn. Для того, чтобы сформировать дерево, нужно назначить вторую учётную запись root, которая позволит определить запись dc=ertw, dc=com, а первый root будет определять организационную единицу для сотрудника и все нижестоящие объекты. После добавления пользователей вы можете входить под различными пользователями, получая доступ ко всему дереву.

Если вы используете бэк-энд bdb, вам также нужно указать, где хранятся файлы базы данных, с помощью команды directory. Каждый экземпляр базы данных должен быть размещен в отдельной директории.

Настройка новой базы данных очень проста, поскольку достаточно ввести всего несколько команд. Сложности начинаются при попытке настроить бэк-энд; эта тема будет раскрыта в следующем руководстве серии 301.

Оверлеи

Оверлеями называются расширения базы данных. Добавить функцию базы данных можно не только работая с кодом, но и с помощью оверлея. Например, если нужно записывать в файл журнала все операции записи, достаточно подключить к соответствующей базе данных оверлей auditlog.

Оверлеи работают по принципу стека. После настройки базы данных вы указываете одну или несколько баз данных. После этого с помощью команды overlay, за которой указывается название оверлея, определяются все нужные оверлеи. У каждого оверлея есть свои параметры.

В случае, если задаётся несколько оверлеев, их запуск производится в порядке, обратном порядку определения. Доступ к базе данных осуществляется только после выполнения всех оверлеев. После того, как данные будут получены из базы, оверлеи запускаются снова в том же порядке, и только после этого slapd возвращает данные клиенту.

На каждом шаге оверлеи могут выполнять различные действия - ведение журнала,

изменение запроса или ответа, прекращение обработки.

Разработка для LDAP с применением Perl/C++

Из этого раздела вы узнаете, как:

- Использовать модуль `Net::LDAP` Perl
- Писать сценарии Perl для связывания, поиска и изменения каталогов
- Разрабатывать программное обеспечение на C/C++

Несмотря на то, что в OpenLDAP реализованы клиенты командной строки, часто бывает полезно использовать информацию LDAP в своих собственных сценариях. Perl - очень популярный язык сценариев. В Perl есть модуль `Net::LDAP`, используемый для подключения к серверу LDAP и работы с ним.

Первое знакомство

`Net::LDAP` не поставляется с Perl, но он может поставляться вместе с вашим дистрибутивом в качестве пакета.

Если пакета `Net::LDAP` в вашем дистрибутиве нет, вы можете загрузить его из сети полного архива Perl (CPAN). Выполните команду `perl -MCPAN -e "install Net::LDAP"` с правами root, она загрузит и установит пакет `Net::LDAP` и все зависимости.

Использование Net::LDAP

Использовать пакет `Net::LDAP` очень просто:

1. Создайте новый объект `Net::LDAP`.
2. Свяжитесь с нужным сервером.
3. Выполните операцию LDAP.

Создание нового объекта

Как обычно, экземпляр модуля `Net::LDAP` в Perl создаётся с помощью метода `new`. Все остальные операции будут выполняться на этом экземпляре. Для работы `new` требуется как минимум, название сервера, к которому вы хотите подключиться. Например:

```
my $ldap = Net::LDAP->new('localhost') or die "$@";
```

В этом примере с помощью метода `new` создаётся новый объект `Net::LDAP`, и ему передаётся строка `localhost`. Результат записывается в переменную `$ldap`. Если функция не выполнится, программа будет остановлена, и на экран будет выведено сообщение, описывающее проблему. $@ - это внутренняя переменная Perl, в которой содержится состояние последней операции.

Вы можете продолжать выполнять операции LDAP на новом объекте `Net::LDAP`. Каждая функция возвращает объект `Net::LDAP::Message`, в котором содержатся состояние операции, возможные сообщения об ошибках и данные, полученные от сервера.

Подключение к дереву

Первая операция, которую необходимо выполнить - подключиться к дереву, или *bind*. В листинге 6 показана операция подключения и соответствующая обработка ошибок.

Листинг 6. Код Perl для подключения к дереву

```
my $message = $ldap->bind(

        "cn=Sean Walberg,ou=people,dc=ertw,dc=com",
        password=>"test" );

if ($message->code() != 0) {
        die $message->error();
}
```

Листинг 6 начинается с вызова метода `bind` созданного ранее объекта. Первый параметр функции - DN, под которым вы подключаетесь. Если DN не указывается, будет создано анонимное подключение. Остальные параметры указываются в формате `key=>value`; чаще всего будет использоваться пароль.

Каждый метод `Net::LDAP` возвращает объект `Net::LDAP::Message`, в котором содержится результат работы функции. Код ошибки можно получить с помощью метода `code`. Код `0` означает успешное выполнение, поэтому в листинге 6 работа программы останавливается, если результат отличен от нуля. Обратите внимание, что ошибка извлекается из `$message->error`, а не из $@, как в предыдущем примере. Причина в том, что это не ошибка Perl, а внутренняя ошибка `Net::LDAP`.

После успешного подключения можно выполнять любые действия, которые будут выполняться в соответствии с политикой контроля доступа сервера. Для отключения необходимо вызвать метод `unbind`.

Поиск по дереву

Поиск выполняется с помощью метода `search`. Так же, как и в методе `bind`, необходимо передать некоторые параметры и проверить результат запроса. Однако теперь в возвращаемом объекте содержатся данные и его необходимо обработать. Результат работы `search` будет помещён в объект `Net::LDAP::Search`, который наследует все методы `Net::LDAP::Message` (например, `code` и `error`) и содержит методы, которые помогут вам анализировать данные. В листинге 7 показан поиск по дереву.

Листинг 7. Поиск по дереву с помощью `search`

```
$message = $ldap->search(base => "dc=ertw,dc=com", filter=> "(objectClass=*)");

if ($message->code() != 0) {
        print $message->error();
} else {
        foreach my $entry ($message->entries()) {
                print $entry->dn() . ": ";
                print join ", ", $entry->get_value("objectClass");
                print "\n";
        }
}
```

Листинг 7 начинается с вызова метода `search`, которому передаётся два параметра - база и фильтр. База сообщает серверу, с какой точки дерева начинать поиск. Дополнительный параметр `scope` говорит серверу, до каких пор продолжать поиск:

- **base** —Только базовый объект
- **one** —Только объекты, дочерние по отношению к базовому объекту (но не сам базовый объект)
- **sub** —Базовый объект и все его дочерние объекты (используется по умолчанию)

Фильтр представляет собой строку, описывающую интересующие вас объекты. Вы можете выполнять поиск по атрибутам и использовать сложные запросы AND/OR. `objectClass=*` возвращает любой объект.

Результат поиска проверяется и, в случае возникновения проблем, выводится сообщение об ошибке. Поскольку сценарий может исправить ошибку, он просто выводит сообщение и продолжает работу.

Функция `entries` возвращает массив объектов `Net::LDAP::Entry`, в каждом из которых содержится один результат. Сначала выводится DN записи, а затем - все классы объекта. Если же вам нужна текстовая версия всей записи, её выведет метод `dump`.

Добавление новой записи

Запись к дереву добавляется с помощью метода `add`. Необходимо передать функции DN записи, которую вы добавляете, и её атрибуты. Атрибуты представляют собой массив пар значений `key =>`. При наличии нескольких экземпляров одного атрибута значение также будет массивом. В листинге 8 показано добавление записи в дерево.

Листинг 8. Добавление записи с помощью `Net::LDAP`

```
$message = $ldap->add(

        "cn=Fred Flintstone,ou=people,dc=ertw,dc=com",

        attr => [
                cn   => "Fred Flintstone",
                sn   => "Flintstone",
                objectclass => [ "organizationalPerson",
                        "inetOrgPerson" ],
        ]
);

if ($message->code() != 0) {
        print $message->error();
}
```

Первый параметр `add` - либо DN, либо объект `Net::LDAP::Entry`. Если передаётся DN, необходимо передать ссылку на массив с помощью метода `attr`. Даже если в качестве ссылки на ассоциативный массив используется формат `key =>` значение, `Net::LDAP` ожидает, что будет передана ссылка на массив, будьте внимательны!

Дополнительная информация о Net::LDAP

`Net::LDAP` предоставляет интерфейс ко всем функциям LDAP, в том числе `compare`, `delete` и `moddn`. Все они используются подобным же образом и подробно описываются в страницах справки `Net::LDAP`.

Все показанные примеры работают в режиме блокировки, что означает, что выход из функции производится после получения ответа от сервера. Кроме того, допускается работа в асинхронном режиме, при этом указывается функция обратного вызова, которая

вызывается при получении пакета.

С помощью `Net::LDAP` можно использовать в сценариях данные из дерева LDAP. Perl используется в огромном множестве приложений, поэтому возможности интеграции очень широки.

Разработка на C/C++

Использование библиотек C сложнее, чем библиотек Perl. На странице справки ldap(3) подробно описано, как работать с этой библиотекой, а также содержатся ссылки на другие страницы, описывающие каждую из функций. Для того, чтобы использовать библиотеки C LDAP, сначала необходимо включить в код файл ldap.h, например, с помощью инструкции `#include <ldap.h>`. После этого объектные файлы необходимо связать с libldap с помощью опции компоновщика `-lldap`.

Резюме

Из этого руководства вы узнали о том, как установить и настроить автономный сервер OpenLDAP. Для настройки `slapd` используется файл slapd.conf. Необходимо располагать глобальные параметры в начале файла, за ними следует конфигурация бэк-энда и базы данных, поскольку `slapd` зависим от порядка директив. При возникновении сомнений обратитесь к странице справки slapd.conf.

Модуль `Net::LDAP` позволяет работать с сервером LDAP из кода Perl. Сначала создаётся объект, после чего вызываются методы объекта, соответствующие операциям LDAP. Как правило, сначала выполняется `bind`, после чего подаётся нужный запрос. Важно проверить результаты работы функций с помощью функций `code` и `error`.