LDAP Linux HOWTO

Luiz Ernesto Pinheiro Malère

LDAP Linux HOWTO

Luiz Ernesto Pinheiro Malère

Publication date v1.10, 2007-03-18

Abstract

Information about installing, configuring, running and maintaining a LDAP (Lightweight Directory Access Protocol) Server on a Linux machine is presented on this document. The document also presents details about how to create LDAP databases, how to add, how to update and how to delete information on the directory. This paper is mostly based on the University of Michigan LDAP information pages and on the OpenLDAP Administrator's Guide.

Table of Contents

List of Tables

Chapter 1. Introduction

This document is no longer being updated, for the latest documentation, please refer to: OpenLDAP Administrator's Guide [http://www.openldap.org/doc/admin23/]

The main purpose of this document is to set up and use a LDAP Directory Server on your Linux machine.You will learn how to install, configure, run and maintain the LDAP server. After you also learn how you can store, retrieve and update information on your Directory using the LDAP clients and utilities. The daemon for the LDAP directory server is called *slapd* and it runs on many different UNIX platforms.

There is another daemon that cares for replication between LDAP servers. It's called slurpd and for the moment you don't need to worry about it. In this document you will run a slapd which provides directory service for your local domain only, without replication, so without slurpd. Complete information about replication is available at: OpenLDAP Administrator's Guide [http://www.openldap.org/doc/admin23/replication.html]

The local domain setup represents a simple choice for configuring your server, good for starting and easy to upgrade to another configuration later if you want. The information presented on this document represents a nice initialization on using the LDAP server. Possibly after reading this document you will feel encouraged to expand the capabilities of your server and even write your own clients, using the already available C, C++ and Java Development Kits.

What's LDAP ?

LDAP stands for Lightweight Directory Access Protocol. As the name suggests, it is a lightweight client-server protocol for accessing directory services, specifically X.500-based directory services. LDAP runs over TCP/IP or other connection oriented transfer services. LDAP is defined in RFC2251 [ftp://ftp.isi.edu/in-notes/rfc2251.txt] "The Lightweight Directory Access Protocol (v3).

A directory is similar to a database, but tends to contain more descriptive, attribute-based information. The information in a directory is generally read much more often than it is written. Directories are tuned to give quick-response to high-volume lookup or search operations. They may have the ability to replicate information widely in order to increase availability and reliability, while reducing response time. When directory information is replicated, temporary inconsistencies between the replicas may be OK, as long as they get in sync eventually.

There are many different ways to provide a directory service. Different methods allow different kinds of information to be stored in the directory, place different requirements on how that information can be referenced, queried and updated, how it is protected from unauthorized access, etc. Some directory services are local, providing service to a restricted context (e.g., the finger service on a single machine). Other services are global, providing service to a much broader context.

How does LDAP work ?

LDAP directory service is based on a client-server model. One or more LDAP servers contain the data making up the LDAP directory tree or LDAP backend database. An LDAP client connects to an LDAP server and asks it a question. The server responds with the answer, or with a pointer to where the client can get more information (typically, another LDAP server). No matter what LDAP server a client connects to, it sees the same view of the directory; a name presented to one LDAP server references the same entry it would at another LDAP server. This is an important feature of a global directory service, like LDAP.

LDAP backends, objects and attributes

The LDAP server daemon is called *Slapd*. *Slapd* supports a variety of different **database backends** which you can use.

They include the **primary choice BDB**, a high-performance transactional database backend; LDBM, a lightweight DBM based backend; SHELL, a backend interface to arbitrary shell scripts and PASSWD, a simple backend interface to the passwd(5) file.

BDB utilizes Sleepycat [http://www.sleepycat.com/] Berkeley DB 4. LDBM utilizes either Berkeley DB [http://www.sleepycat.com/] or GDBM [http://www.gnu.org/software/gdbm/].

BDB transactional backend is suited for multi-user read/write database access, with any mix of read and write operations. BDB is used in applications that require:

- Transactions, including making multiple changes to the database atomically and rolling back uncommitted changes when necessary.
- Ability to recover from systems crashes and hardware failures without losing any committed transactions.

In this document I assume that you choose the **BDB database.**

To import and export directory information between LDAP-based directory servers, or to describe a set of changes which are to be applied to a directory, the file format known as LDIF, for LDAP Data Interchange Format, is typically used. A LDIF file stores information in object-oriented hierarchies of entries. The LDAP software package you're going to get comes with an utility to convert LDIF files to the BDB format

A common LDIF file looks like this:

```
dn: o=TUDelft, c=NL
o: TUDelft
objectclass: organization
dn: cn=Luiz Malere, o=TUDelft, c=NL
cn: Luiz Malere
sn: Malere
mail: malere@yahoo.com
objectclass: person
```

As you can see each entry is uniquely identified by a distinguished name, or DN. The DN consists of the name of the entry plus a path of names tracing the entry back to the top of the directory hierarchy (just like a tree).

In LDAP, an **object class** defines the collection of **attributes** that can be used to define an entry. The LDAP standard provides these basic types of object classes:

- Groups in the directory, including unordered lists of individual objects or groups of objects.
- Locations, such as the country name and description.
- Organizations in the directory.
- People in the directory.

An entry can belong to more than one object class. For example, the entry for a person is defined by the *person* object class, but may also be defined by attributes in the inetOrgPerson, groupOfNames, and

organization objectclasses. The server's object class structure (it's schema) determines the total list of required and allowed attributes for a particular entry.

Directory data is represented as attribute-value pairs. Any specific piece of information is associated with a descriptive attribute.

For instance, the commonName, or cn, attribute is used to store a person's name . A person named Jonas Salk can be represented in the directory as

```
cn: Jonas Salk
```

Each person entered in the directory is defined by the collection of attributes in the *person* object class. Other attributes used to define this entry could include:

```
givenname: Jonas
surname: Salk
mail: jonass@airius.com
```

Required attributes include the attributes that must be present in entries using the object class. All entries require the objectClass attribute, which lists the object classes to which an entry belongs.

Allowed attributes include the attributes that may be present in entries using the object class. For example, in the person object class, the cn and sn attributes are required. The description, telephoneNumber, seeAlso, and userpassword attributes are allowed but are not required.

Each attribute has a corresponding syntax definition. The syntax definition describes the type of information provided by the attribute, for instance:

- bin binary.
- ces case exact string (case must match during comparisons).
- cis case ignore string (case is ignored during comparisons).
- tel telephone number string (like cis but blanks and dashes `- ' are ignored during comparisons).
- dn distinguished name.

Note: Usually objectclass and attribute definitions reside on schema files, on the subdirectory *schema* under the OpenLDAP installation home.

New versions of this document

This document may receive corrections and updates based on the feedback received by the readers. You should look at:

http://www.tldp.org/HOWTO/LDAP-HOWTO.html

for new versions of this HOWTO.

Opinions and Sugestions

If you have any kind of doubt about some information avaiable on this document, please contact me on the following email address: malere@yahoo.com

If you have commentaries and/or sugestions, please let me know too!

Acknowledgments

This Howto was result of an internship made by me on the TUDelft University - Netherlands. I would like to thank the persons that encouraged me to write this document: *Rene van Leuken* and *Wim Tiwon*. Thank you very much. They are also Linux fans, just like me.

I would like to thank also Thomas Bendler, author of the German Ldap-Howto, for his contributions to my document and Joshua Go, great volunteer on the LDP project.

Karl Lattimer deserves a prize, for his great contribution on SASL related issues.

And thanks my Lord!

Copyright and Disclaimer

Chapter 2. Installing the LDAP Server

Five steps are necessary to install the server:

- Install the pre-required packages (if not already installed).
- Download the server.
- Unpack the software.
- Configure the Makefiles.
- Build the server.

Pre-Requirements

To be fully LDAPv3 compliant, OpenLDAP clients and servers require installation of some additional packages. For writing this document, I've used a Mandrake 9.0 box with a 2.4.20 Kernel, manually installing the Berkeley BDB package and SASL libraries.

OpenSSL TLS Libraries

The OpenSSL TLS libraries are normally part of the base system or compose an optional software component. The official OpenSSL url is: http://www.openssl.org

Kerberos Authentication Services

OpenLDAP clients and servers support Kerberos-based authentication services. In particular, OpenLDAP supports SASL/GSSAPI authentication mechanism using either Heimdal or MIT Kerberos V packages. If you desire to use Kerberos-based SASL/GSSAPI authentication, you should install either Heimdal or MIT Kerberos V. Heimdal Kerberos is available from http://www.pdc.kth.se/heimdal MIT Kerberos is available from http://web.mit.edu/kerberos/www

The use of strong authentication services, such as those provided by Kerberos, is highly recommended.

Cyrus's Simple Authentication and Security Layer Libraries

Cyrus's SASL libraries are normally part of the base system or compose an optional software component. Cyrus SASL is available from http://asg.web.cmu.edu/sasl/sasl-library.html. Cyrus SASL will make use of OpenSSL and Kerberos/GSSAPI libraries if preinstalled. By the time of this writing, I've used Cyrus SASL 2.1.17.

Database Software

Slapd's primary database backend, BDB, requires Sleepycat Software Berkeley DB [http://www.sleepycat.com], version 4. If not available at configure time, you will not be able to build slapd with primary database backend.

Your operating system may provide Berkeley DB, version 4, in the base system or as an optional software component. If not, there are several versions available at Sleepycat [http://www.sleepycat.com/download.html]. At the time of this writing, the latest release, version 4.2.52, is recommended. OpenLDAP's slapd LDBM backend supports a variety of database managers, like Berkeley DB (version 3) and GDBM. GDBM is available from FSF's [http://www.fsf.org/] download site ftp://ftp.gnu.org/pub/gnu/gdbm/.

Threads

Threads support are almost guaranteed to be part of your base Linux system. OpenLDAP is designed to take advantage of threads. OpenLDAP supports POSIX pthreads, Mach CThreads, and a number of other varieties. The *configure* script will complain if it cannot find a suitable thread subsystem. If this occurs, please consult the Software - Installation - Platform Hints section of the OpenLDAP FAQ: http://www.openldap.org/faq/.

TCP Wrappers

Slapd supports TCP wrappers (IP level access control filters) if preinstalled. Use of TCP wrappers or other IP-level access filters (such as those provided by an IP-level firewall) is recommended for servers containing non-public information.

Downloading the Package

There are two free distributed LDAP servers: University of Michigan LDAP server and OpenLDAP server. There's also the Netscape Directory Server, which is free only under some conditions (educational institutions get it free, for example). The OpenLDAP server is based on the latest version of the University of Michigan Server and there are mailing lists and additional documentation available for it. This document assumes that you are using the OpenLDAP server.

It's latest tar gzipped version is avaiable on the following address:

http://www.openldap.org

If you want to get the latest version of University of Michigan Server, go to this address:

ftp://terminator.rs.itd.umich.edu/ldap

To write this document, I used the 2.2.5 version of the OpenLDAP package. My operating system is a Mandrake Linux 9.0 with kernel 2.4.20.

On the OpenLDAP site you can always find the latest development and stable versions of the OpenLDAP server. By the time this document was updated, the latest stable version was openldap-stable-20031217.tgz (version 2.1.25). The latest development version was also openldap-2.2.5.tgz.

Unpacking the Software

Now that you have the tar gzipped package on your local machine, you can unpack it.

First copy the package to a desirable directory, for example /usr/local. Next use the following command:

```
tar xvzf openldap-2.2.5.tgz
```

You can use this command too, as well:

```
gunzip openldap-2.2.5.tgz | tar xvf -
```

Configuring the Software

The OpenLDAP server sources are distributed with a configuration script for setting options like installation directories, compiler and linker flags. Type the following command on the directory where you unpacked the software:

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

This will print all options that you can customize with the configure script before you build the software. Some usefull options are --prefix=pref , --exec-prefix=eprefix and --bindir=dir, for setting instalation directories. Normally if you run configure without options, it will auto-detect the appropriate settings and prepare to build things on the default common location. So just type:

```
./configure
```

And watch the output to see if all went well

Tip: Sometimes you need to pass specific options to your configure script, like for example --with-tls (for enabling slapd to use a secure channel: LDAPS://). In this case, you might have your SSL/TLS libraries residing on a non-standard directory of your system. You can make the configure script aware of the libraries location changing you environment with the *env* command. Example: suppose you've installed the openssl package under /usr/local/openssl. The following command will build slapd with SSL/TLS support:

```
env CPPFLAGS=-I/usr/local/openssl/include \
      LDFLAGS=-L/usr/local/openssl/lib \
      configure --with-tls ...
```

You can specify the following environment variables with the *env* command before the configure script:

- CC: Specify alternative C Compiler.
- CFLAGS: Specify additional compiler flags.
- CPPFLAGS: Specify C Preprocessor flags.
- LDFLAGS: Specify linker flags.
- LIBS: Specify additional libraries.

Building the Server

After configuring the software you can start building it. First build the dependencies, using the command:

```
make depend
```

Build the server after that, using the command:

```
make
```

If all goes well, the server will build as configured. If not, return to the previous step to review the configuration settings. You should read the INSTALL and README files located in the directory where you unpacked the software. Also, check the configure script specific hints, they are located in the path doc/install/configure under the directory you unpacked the software.

To ensure a correct build, you should run the test suite (it only takes a few minutes):

```
make test
```

Tests which apply to your configuration will run and they should pass. Some tests, such as the replication test, may be skipped.

Now install the binaries and man pages. You may need to be superuser to do this (depending on where you are installing things):

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

That's all, now you have the binary of the server and the binaries of several other utilities. Go to the Chapter 3, *Configuring the LDAP Server* section to see how to configure the operation of your LDAP server.

Chapter 3. Configuring the LDAP Server

Once the software has been installed and built, you are ready to configure it for use at your site. All slapd runtime configuration is accomplished through the *slapd.conf* file, installed in the prefix directory you specified in the configuration script or by default in /usr/local/etc/openldap.

This section details the commonly used configuration directives in *slapd.conf*. For a complete list, see the slapd.conf(5) manual page. The configuration file directives are separated into **global**, **backend specific** and **database specific**. Here you will find descriptions of directives, together with their default values (if any) and examples of use.

Configuration File Format

The slapd.conf file consists of three types of configuration information: global, backend specific, and database specific. Global information is specified first, followed by information associated with a particular backend type, which is then followed by information associated with a particular database instance.

Global directives can be overridden in a backend and/or database directives, backend directives can be overridden by database directives.

Blank lines and comment lines beginning with a '#' character are ignored. If a line begins with white space, it is considered a continuation of the previous line (even if the previous line is a comment). The general format of slapd.conf is as follows:

```
# global configuration directives
<global config directives>

# backend definition
backend <typeA>
<backend-specific directives>

# first database definition & config directives
database <typeA>
<database-specific directives>

# second database definition & config directives
database <typeB>
<database-specific directives>

# second "typeA" database definition & config directives
database <typeA>
<database-specific directives>

# subsequent backend & database definitions & config directives
...
```

A configuration directive may take arguments. If so, they are separated by white space. If an argument contains white space, the argument should be enclosed in double quotes "like this". If an argument contains a double quote or a backslash character `\', the character should be preceded by a backslash character `\'.

The distribution contains an example configuration file that will be installed in the /usr/local/etc/openldap directory. A number of files containing schema definitions (attribute types and object classes) are also provided in the /usr/local/etc/openldap/schema directory.

Global Directives

Directives described in this section apply to all backends and databases unless specifically overridden in a backend or database definition. Arguments that should be replaced by actual text are shown in brackets <>.

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

This directive grants access (specified by <accesslevel>) to a set of entries and/or attributes (specified by <what>) by one or more requesters (specified by <who>). See the the section called “Access Control Examples” examples for more details.

Important: If no access directives are specified, the default access control policy, access to * by * read, allows all both authenticated and anonymous users read access.

```
attributetype <RFC2252 Attribute Type Description>
```

This directive defines an attribute type. Check the following URL for more details: Schema Specification [http://www.openldap.org/doc/admin22/schema.html]

```
idletimeout <integer>
```

Specify the number of seconds to wait before forcibly closing an idle client connection. An idletimeout of 0, the default, disables this feature.

```
include <filename>
```

This directive specifies that slapd should read additional configuration information from the given file before continuing with the next line of the current file. The included file should follow the normal slapd config file format. The file is commonly used to include files containing schema specifications.

Note: You should be careful when using this directive - there is no small limit on the number of nested include directives, and no loop detection is done.

```
loglevel <integer>
```

This directive specifies the level at which debugging statements and operation statistics should be syslogged (currently logged to the syslogd(8) LOCAL4 facility). You must have configured OpenLDAP --enable-debug (the default) for this to work (except for the two statistics levels, which are always enabled). Log levels are additive. To display what numbers correspond to what kind of debugging, invoke slapd with -? or consult the table below. The possible values for <integer> are:

Table 3.1. Debugging Levels

Level	Description
-1	enable all debugging

Level	Description
0	no debugging
1	trace function calls
2	debug packet handling
4	heavy trace debugging
8	connection management
16	print out packets sent and received
32	search filter processing
64	configuration file pro-

Level	Description
-1	Debugging
0	No debugging
1	Trace function calls
2	Debug packet handling
4	Heavy trace debugging
8	Connection management
16	Print out packets sent and received
32	Search filter processing
64	Configuration file processing
128	Access control list processing
256	Stats log connections/operations/results
512	Stats log entries sent
1024	Communication with shell backends
2048	Entry parsing debugging

Example:

loglevel 255 or loglevel -1

This will cause lots and lots of debugging information to be syslogged.

Default:

loglevel 256

`objectclass <RFC2252 Object Class Description>`

This directive defines an object class. Check the following URL for more details: Schema Specification [http://www.openldap.org/doc/admin22/schema.html]

`referral <URI>`

This directive specifies the referral to pass back when slapd cannot find a local database to handle a request.

Example:

referral ldap://root.openldap.org

This will refer non-local queries to the global root LDAP server at the OpenLDAP Project. Smart LDAP clients can re-ask their query at that server, but note that most of these clients are only going to know how to handle simple LDAP URLs that contain a host part and optionally a distinguished name part.

`sizelimit <integer>`

This directive specifies the maximum number of entries to return from a search operation.

Default:

sizelimit 500

`timelimit <integer>`

This directive specifies the maximum number of seconds (in real time) slapd will spend answering a search request. If a request is not finished in this time, a result indicating an exceeded timelimit will be returned.

Default:

timelimit 3600

General Backend Directives

Directives in this section apply only to the backend in which they are defined. They are supported by every type of backend. Backend directives apply to all databases instances of the same type and, depending on the directive, may be overridden by database directives.

`backend <type>`

This directive marks the beginning of a backend definition. <type> should be one of bdb or one of other supported backend types listed below:

Table 3.2. Database Backends

Type	Description
bdb	Berkeley DB transac-

Type	Description
tional	back-end
dnssrv	DNS SRV back-end
ldbm	lightweight DBM back-end
ldap	lightweight Directory Access Protocol (Proxy) back-end
meta	Meta Directory back-end
monitor	Monitor back-end
passwd	provides read-only access to

Type	Description
passwd	passwd(5)
perl	Perl programmable backend
shell	Shell (external program) backend
sql	SQL programmable backend

Example:

backend bdb

This marks the beginning of a new BDB backend definition

General Database Directives

Directives in this section apply only to the database in which they are defined. They are supported by every type of database.

```
database <type>
```

This directive marks the beginning of a new database instance definition. <type> should be one of the backend types listed on the previous item.

Example:

database bdb

This marks the beginning of a new BDB backend database instance definition.

```
readonly { on | off }
```

This directive puts the database into "read-only" mode. Any attempts to modify the database will return an "unwilling to perform" error.

Default:

readonly off

```
replica uri=ldap[s]://<hostname>[:<port>] | host=<hostname>[:<port>]
                [bindmethod={simple|kerberos|sasl}]
                ["binddn=<DN>"]
                [saslmech=<mech>]
                [authcid=<identity>]
                [authzid=<identity>]
                [credentials=<password>]
                [srvtab=<filename>]
```

This directive specifies a replication site for this database. The uri= parameter specifies a scheme, a host and optionally a port where the slave slapd instance can be found. Either a domain name or IP address may be used for <hostname>. If <port> is not given, the standard LDAP port number (389 or 636) is used.

Note: host is deprecated in favor of the uri parameter.

uri allows the replica LDAP server to be specified as an LDAP URI such as ldap://slave.example.com:389 or ldaps://slave.example.com:636

The binddn= parameter gives the DN to bind as for updates to the slave slapd. It should be a DN which has read/write access to the slave slapd's database. It must also match the updatedn directive in the slave slapd's config file. Generally, this DN *should not* be the same as the rootdn of the master database. Since DNs are likely to contain embedded spaces, the entire "binddn=<DN>" string should be enclosed in double quotes.

The bindmethod is simple or kerberos or sasl, depending on whether simple password-based authentication or Kerberos authentication or SASL authentication is to be used when connecting to the slave slapd.

Simple authentication should not be used unless adequate integrity and privacy protections are in place (e.g. TLS or IPSEC). Simple authentication requires specification of binddn and credentials parameters.

Kerberos authentication is deprecated in favor of SASL authentication mechanisms, in particular the KERBEROS_V4 and GSSAPI mechanisms. Kerberos authentication requires binddn and srvtab parameters.

SASL authentication is generally recommended. SASL authentication requires specification of a mechanism using the saslmech parameter. Depending on the mechanism, an authentication identity and/or credentials can be specified using authcid and credentials respectively. The authzid parameter may be used to specify an authorization identity.

Check this URL for additional details: Replication with Slurpd [http://www.openldap.org/doc/admin22/replication.html].

```
replogfile <filename>
```

This directive specifies the name of the replication log file to which slapd will log changes. The replication log is typically written by slapd and read by slurpd. Normally, this directive is only used if slurpd is being used to replicate the database. However, you can also use it to generate a transaction log, if slurpd is not running. In this case, you will need to periodically truncate the file, since it will grow indefinitely otherwise.

Check this URL for additional details: Replication with Slurpd [http://www.openldap.org/doc/admin22/replication.html].

`rootdn <dn>`

This directive specifies the DN that is not subject to access control or administrative limit restrictions for operations on this database. The DN need not refer to an entry in the directory. The DN may refer to a SASL identity.

Entry-based Example:

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

SASL-based Example:

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

`rootpw <password>`

This directive can be used to specify a password for the rootdn (when the rootdn is set to a DN within the database).

Example:

rootpw secret

It is also permissible to provide hash of the password in RFC 2307 form. slappasswd may be used to generate the password hash.

Example:

rootpw {SSHA}ZKKuqbEKJfKSXhUbHG3fG8MDn9j1v4QN

The hash was generated using the command slappasswd -s secret.

`suffix <dn suffix>`

This directive specifies the DN suffix of queries that will be passed to this backend database. Multiple suffix lines can be given, and at least one is required for each database definition.

Example:

suffix "dc=example, dc=com"

Queries with a DN ending in "dc=example, dc=com" will be passed to this backend.

Note: When the backend to pass a query to is selected, slapd looks at the suffix line(s) in each database definition in the order they appear in the file. Thus, if one database suffix is a prefix of another, it must appear after it in the config file.

`syncrepl`

This directive is used to keep a replicated database synchronized with the master database, so that the replicated database content will be kept up to date with the master content.

This document doesn't cover in details this directive, because we're configuring a single LDAP Server. For more informations about this directive, please visit : LDAP Sync Replication [http://www.openldap.org/doc/admin22/syncrepl.html].

`updatedn <dn>`

This directive is only applicable in a slave slapd. It specifies the DN allowed to make changes to the replica. This may be the DN slurpd binds as when making changes to the replica or the DN associated with a SASL identity.

Entry-based Example:

updatedn "cn=Update Daemon, dc=example, dc=com"

SASL-based Example:

updatedn "uid=slurpd,cn=example.com,cn=digest-md5,cn=auth"

Check this URL for additional details: Replication with Slurpd [http://www.openldap.org/doc/admin22/replication.html].

`updateref <URL>`

This directive is only applicable in a slave slapd. It specifies the URL to return to clients which submit update requests upon the replica. If specified multiple times, each URL is provided.

Example:

updateref ldap://master.example.net

BDB Database Directives

Directives in this category only apply a BDB database. That is, they must follow a "database bdb" line and come before any subsequent "backend" or "database" line. For a complete reference of BDB configuration directives, see the slapd-bdb manpages (*man slapd-bdb*).

`directory <directory>`

This directive specifies the directory where the BDB files containing the database and associated indexes reside.

Default:

directory /usr/local/var/openldap-data

`sessionlog <sid> <limit>`

This directive specifies a session log store in the syncrepl replication provider server which contains information on the entries that have been scoped out of the replication content identified by <sid>. The first syncrepl search request having the same <sid> value in the cookie establishes the session log store in the provider server. The number of the entries in the session log store is limited by <limit>. Excessive entries are removed from the store in the FIFO order. Both <sid> and <limit> are non-negative integers. <sid> has no more than three decimal digits.

The LDAP Content Synchronization operation that falls into a pre-existing session can use the session log store in order to reduce the amount of synchronization traffic. If the replica is not so outdated that it can be made up-to-date by the information in the session store, the provider slapd will send the consumer slapd the identities of the scoped-out entries together with the in-scope entries added to or modified within the replication content. If the replica status is outdated too much and beyond the coverage of the history store, then the provider slapd will send the identities of the unchanged in-scope entries along with the changed in-scope entries. The consumer slapd will then remove those entries in the replica which are not identified as present in the provider content.

For more informations about syncrepl, please visit : LDAP Sync Replication [http://www.openldap.org/doc/admin22/syncrepl.html].

LDBM Database Directives

Directives in this category only apply to the LDBM backend database. That is, they must follow a "database ldbm" line and come before any other "database" or "backend" line. For a complete reference of LDBM configuration directives, see the slapd-ldbm manpages (*man slapd-ldbm*).

`cachesize <integer>`

This directive specifies the size in entries of the in-memory cache maintained by the LDBM backend database instance.

Default:

cachesize 1000

`dbcachesize <integer>`

This directive specifies the size in bytes of the in-memory cache associated with each open index file. If not supported by the underlying database method, this directive is ignored without comment. Increasing this number uses more memory but can cause a dramatic performance increase, especially during modifies or when building indexes.

Default:

dbcachesize 100000

`dbnolocking`

This option, if present, disables database locking. Enabling this option may improve performance at the expense of data security.

`dbnosync`

This option causes on-disk database contents not to be immediately synchronized with in memory changes upon change. Enabling this option may improve performance at the expense of data security.

`directory <directory>`

This directive specifies the directory where the LDBM files containing the database and associated indexes live.

Default:

directory /usr/local/var/openldap-data

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

This directive specifies the indexes to maintain for the given attribute. If only an <attrlist> is given, the default indexes are maintained.

Example:

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

The first line sets the default set of indexes to maintain to present and equality. The second line causes the default (pres,eq) set of indices to be maintained for the uid attribute type. The third line causes present, equality and substring indices to be maintained for cn and sn attribute types. The fourth line causes an equality index for the objectClass attribute type.

By default, no indices are maintained. It is generally advised that minimally an equality index upon objectClass be maintained.

index objectClass eq

```
mode <integer>
```

This directive specifies the file protection mode that newly created database index files should have.

Default:

mode 0600

Access Control Examples

The access control facility provided by the *access* directive is quite powerful. This section shows some examples of it's use. First, some simple examples:

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

This access directive grants read access to everyone.

The following example shows the use of a regular expression to select the entries by DN in two access directives where ordering is significant.

```
access to dn=".*, o=U of M, c=US"
by * search
access to dn=".*, c=US"
by * read
```

Read access is granted to entries under the c=US subtree, except for those entries under the "o=U of M, c=US" subtree, to which search access is granted. No access is granted to c=US as neither access directive matches this DN.If the order of these access directives was reversed, the U-M-specific directive would never be matched, since all U-M entries are also c=US entries.

Another way to implement the same access controls is:

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

Read access is granted to entries under the dc=com subtree, except for those entries under the dc=example,dc=com subtree, to which search access is granted. No access is granted to dc=com as neither access directive matches this DN. If the order of these access directives was reversed, the trailing directive would never be reached, since all entries under dc=example,dc=com are also under dc=com entries.

Note: Also note that if no access to directive or no "by <who>" clause matches, **access is denied**. That is, every *access to* directive ends with an implicit *by * none* clause and every access list ends with an implicit *access to * by * none* directive.

The next example again shows the importance of ordering, both of the access directives and the "by <who>" clauses. It also shows the use of an attribute selector to grant access to a specific attribute and various <who> selectors.

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

This example applies to entries in the "dc=example,dc=com" subtree. To all attributes except homePhone, an entry can write to itself, entries under example.com entries can search by them, anybody else has no access (implicit by * none) excepting for authentication/authorization (which is always done anonymously). The homePhone attribute is writable by the entry, searchable by entries under example.com, readable by clients connecting from network 10, and otherwise not readable (implicit by * none). All other access is denied by the implicit access to * by * none.

Sometimes it is useful to permit a particular DN to add or remove itself from an attribute. For example, if you would like to create a group and allow people to add and remove only their own DN from the member attribute, you could accomplish it with an access directive like this:

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

The dnattr <who> selector says that the access applies to entries listed in the member attribute. The selfwrite access selector says that such members can only add or delete their own DN from the attribute, not other values. The addition of the entry attribute is required because access to the entry is required to access any of the entry's attributes.

There's plenty of information about Access Control on the OpenLDAP Administrator's Guide. Take a look at: Access Control [http://www.openldap.org/doc/admin22/slapdconfig.html#Access Control] for more information about this subject.

Configuration File Example

The following is an example configuration file, interspersed with explanatory text. It defines two databases to handle different parts of the X.500 tree; both are BDB database instances. The line numbers shown are provided for reference only and are not included in the actual file. First, the global configuration section:

```
1.    # example config file - global configuration section
2.    include /usr/local/etc/schema/core.schema
3.    referral ldap://root.openldap.org
4.    access to * by * read
```

Line 1 is a comment. Line 2 includes another config file which contains core schema definitions. The referral directive on line 3 means that queries not local to one of the databases defined below will be referred to the LDAP server running on the standard port (389) at the host root.openldap.org.

Line 4 is a global access control. It applies to all entries (after any applicable database-specific access controls).

The next section of the configuration file defines a BDB backend that will handle queries for things in the "dc=example,dc=com" portion of the tree. The database is to be replicated to two slave slapds, one on truelies, the other on judgmentday. Indexes are to be maintained for several attributes, and the userPassword attribute is to be protected from unauthorized access.

```
5.     # BDB definition for the example.com
6.     database bdb
7.     suffix "dc=example,dc=com"
8.     directory /usr/local/var/openldap-data
9.     rootdn "cn=Manager,dc=example,dc=com"
10.    rootpw secret
11.    # replication directives
12.    replogfile /usr/local/var/openldap/slapd.replog
13.    replica uri=ldap://slave1.example.com:389
14.            binddn="cn=Replicator,dc=example,dc=com"
15.            bindmethod=simple credentials=secret
16.    replica uri=ldaps://slave2.example.com:636
17.            binddn="cn=Replicator,dc=example,dc=com"
18.            bindmethod=simple credentials=secret
19.    # indexed attribute definitions
20.    index uid pres,eq
21.    index cn,sn,uid pres,eq,sub
22.    index objectClass eq
23.    # database access control definitions
24.    access to attr=userPassword
25.            by self write
26.            by anonymous auth
27.            by dn.base="cn=Admin,dc=example,dc=com" write
28.            by * none
29.    access to *
30.            by self write
31.            by dn.base="cn=Admin,dc=example,dc=com" write
32.            by * read
```

Line 5 is a comment. The start of the database definition is marked by the database keyword on line 6. Line 7 specifies the DN suffix for queries to pass to this database. Line 8 specifies the directory in which the database files will live.

Lines 9 and 10 identify the database "super user" entry and associated password. This entry is not subject to access control or size or time limit restrictions. Please remeber to encrypt the rootpw using slappasswd.

Example: rootpw {SSHA}Jq4xhhkGa7weT/0xKmaecT4HEXsdqiYA

Lines 11 through 18 are for replication. See the Replication [http://www.openldap.org/doc/admin22/replication.html] link for more information on these directives.

Lines 20 through 22 indicate the indexes to maintain for various attributes.

Lines 24 through 32 specify access control for entries in the this database. As this is the first database, the controls also apply to entries not held in any database (such as the Root DSE). For all applicable entries, the userPassword attribute is writable by the entry itself and by the "admin" entry. It may be used for

authentication/authorization purposes, but is otherwise not readable. All other attributes are writable by the entry and the "admin" entry, but may be read by all users (authenticated or not).

The next section of the example configuration file defines another BDB database. This one handles queries involving the dc=example,dc=net subtree but is managed by the same entity as the first database. Note that without line 39, the read access would be allowed due to the global access rule at line 4.

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

Chapter 4. Running the LDAP Server

The LDAP daemon *slapd* is designed to be run as a stand-alone server. This allows the server to take advantage of caching, manage concurrency issues with underlying databases, and conserve system resources. Running from inetd(8) is not an option.

Command Line Options

Slapd supports a number of command-line options as detailed in the manual page. This section details a few commonly used options:

```
-f <filename>
```

This option specifies an alternate configuration file for slapd. The default is normally /usr/local/etc/openldap/slapd.conf.

```
-h <URLs>
```

This option specifies alternative listener configurations. The default is ldap:/// which implies LDAP over TCP on all interfaces on the default LDAP port 389. You can specify specific host-port pairs or other protocol schemes (such as ldaps:// or ldapi://). For example, -h "ldaps:// ldap://127.0.0.1:667" will create two listeners: one for LDAP over SSL on all interfaces on the default LDAP/SSL port 636, and one for LDAP over TCP on the localhost (loopback) interface on port 667. Hosts may be specified using IPv4 dotted-decimal form or using host names. Port values must be numeric.

```
-n <service-name>
```

This option specifies the service name used for logging and other purposes. The default service name is slapd.

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

This option specifies the local user for the syslog(8) facility. Values can be LOCAL0, LOCAL1, LOCAL2, ..., and LOCAL7. The default is LOCAL4. This option may not be supported on all systems. See the the section called “Logs” for more details.

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

These options specify the user and group, respectively, to run slapd as. user can be either a user name or uid. group can be either a group name or gid.

```
-r directory
```

This option specifies a run-time directory. slapd will chroot(2) to this directory after opening listeners but before reading any configuration files or initializing any backends.

```
-d <level> | ?
```

This option sets the slapd debug level to <level>. When level is a `?' character, the various debugging levels are printed and slapd exits, regardless of any other options you give it. Current debugging levels are:

Table 4.1. Debugging Levels

Level	Description
-1	enable all debugging
0	no debugging
1	trace function calls
2	debug packet handling
4	heavy trace debugging
8	connection management
16	print out packets sent and received
32	search filter processing

Devel	Description
64	configuration file processing
128	access control list processing
256	stats log connections/operations/results
512	stats log entries sent
1024	communication with shell backends
2048	entry parsing de-

Description	Description
debugging	debugging

You may enable multiple levels by specifying the debug option once for each desired level. Or, since debugging levels are additive, you can do the math yourself. That is, if you want to trace function calls and watch the config file being processed, you could set level to the sum of those two levels (in this case, -d 65). Or, you can let slapd do the math, (e.g. -d 1 -d 64). Consult <ldap.h> for more details.

Note: slapd must have been compiled with -DLDAP_DEBUG defined for any debugging information beyond the two stats levels to be available.

Starting the LDAP Server

In general, slapd is run like this:

```
/usr/local/etc/libexec/slapd [<option>]*
```

Where /usr/local/etc/libexec is determined by configure and <option> is one of the options described above (or in slapd(8)). Unless you have specified a debugging level (including level 0), slapd will automatically fork and detach itself from its controlling terminal and run in the background.

Killing the LDAP Server

To kill off slapd safely, you should give a command like this:

```
kill -INT `cat $(ETCDIR)/slapd.pid`
```

Killing slapd by a more drastic method may cause it's databases to be corrupted, as it may need to flush various buffers before it exits. Note that slapd writes its pid to a file called slapd.pid in the directory you configured in slapd.conf file, for example: /usr/local/var/slapd.pid

Slapd will also write its arguments to a file called slapd.args in the directory you configured in slapd.conf file, for example /usr/local/var/slapd.args

Chapter 5. Database Creation and Maintenance

This section tells you how to create a slapd database from scratch. There are two ways to create a database. First, you can create the database on-line using LDAP. With this method, you simply start up slapd and add entries using the LDAP client of your choice. This method is fine for relatively small databases (a few hundred or thousand entries, depending on your requirements). This method works for database types which support updates.

The second method of database creation is to do it off-line using special utilities provided with slapd. This method is best if you have many thousands of entries to create, which would take an unacceptably long time using the LDAP method, or if you want to ensure the database is not accessed while it is being created. Note that not all database types support these utilitites.

Creating a Database online

The OpenLDAP software package comes with an utility called ldapadd, used to add entries while the LDAP server is running. If you choose to create the Database online, you can use the ldapadd tool to add entries (you can also use other clients provided outside the OpenLDAP package to add entries, like the Ldap Browser [http://www.iit.edu/~gawojar/ldap/]). After adding the first entries, you can still use ldapadd to add more entries. You should be sure to set the following configuration options on your sladp.conf file before starting slapd:

```
suffix <dn>
```

As described in the the section called “General Database Directives”, this option says what entries are to be held by this database. You should set this to the DN of the root of the subtree you are trying to create. For example:

```
suffix "o=TUDelft, c=NL"
```

You should be sure to specify a directory where the index files should be created:

```
directory /usr/local/tudelft
```

You need to create this directory with appropriate permissions so that slapd can write to it.

You need to configure slapd so that you can connect to it as a directory user with permission to add entries. You can configure the directory to support a special super-user or root user just for this purpose. This is done through the following two options in the database definition:

```
rootdn <dn>
rootpw <passwd>   /* Remember to use a SHA password here !!! */
```

These options specify a DN and password that can be used to authenticate as the "superuser" entry of the database (i.e., the entry allowed to do anything). The DN and password specified here will always work, regardless of whether the entry named actually exists or has the password given. This solves the chicken-and-egg problem of how to authenticate and add entries before any entries yet exist.

Slapd natively understands if you use a SHA-1 encrypted password on the rootpw directive. I use a Java class that generates SHA-1 passwords, but it's possible to use the command *slappasswd* to generate the passwords:

```
slappasswd -h {SHA}

rootpw    "{SHA}5en6G6MezRroT3XKqkdPOmY/BfQ="
```

For example:

```
        rootdn "cn=Manager,dc=example,dc=com"
        rootpw "{SHA}5en6G6MezRroT3XKqkdPOmY/BfQ="
```

The default output for slappasswd is to generate Secure Hash passwords {SSHA}, in this case you don't need to pass the -h parameter, just call slappasswd directly.

If you are using SASL as a mechanism to authenticate against LDAP, the rootpw line may be discarded. Take a look on the the section called “General Database Directives” and on the the section called “Authentication using LDAP” for more details.

Finally, you should make sure that the database definition contains the index definitions you want:

```
index {<attrlist> | default} [pres,eq,sub,none]
```

For example, to index the cn, sn, uid and objectclass attributes, the following index configuration lines could be used.

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

Note: Note that not all index types are available with all attribute types. Take a look on the the section called “LDBM Database Directives” for examples.

Once you have configured things to your liking, start up slapd, connect with your LDAP client, and start adding entries. For example, to add the TUDelft entry followed by a Postmaster entry using the ldapadd tool, you could create a file called /tmp/newentry with the contents:

```
o=TUDelft, c=NL
objectClass=organization
description=Technical University of Delft Netherlands

cn=Postmaster, o=TUDelft, c=NL
objectClass=organizationalRole
cn=Postmaster
description= TUDelft postmaster - postmaster@tudelft.nl
```

and then use a command like this to actually create the entry:

```
ldapadd -f /tmp/newentry -x -D "cn=Manager, o=TUDelft, c=NL" -w secret
```

The above command assumes that you have set rootdn to "cn=Manager, o=TUDelft, c=NL" and rootpw to "secret" (maybe SHA-1 encrypted in slapd.conf). If you don't want to type the password on the command line, use the -W option for the ldapadd command instead of -w "password". You will be prompted to enter the password:

```
ldapadd -f /tmp/newentry -x -D "cn=Manager, o=TUDelft, c=NL" -W
```

```
Enter LDAP Password:
```

Creating a Database offline

The second method of database creation is to do it off-line, using the slapd database tools described below. This method is best if you have many thousands of entries to create, which would take an unacceptably long time using the LDAP method described above. These tools read the slapd configuration file and an input LDIF file containing a text representation of the entries to add. For database types which support the tools, they produce the database files directly (otherwise you must use the on-line method above). There are several important configuration options you will want to be sure and set in the config file database definition first:

```
suffix <dn>
```

As described in the preceding section, this option says which entries are to be held by this database. You should set this to the DN of the root of the subtree you are trying to create. For example:

```
suffix "o=TUDelft, c=NL"
```

You should be sure to specify a directory where the index files should be created:

```
directory /usr/local/tudelft
```

Finally, you need to specify which indexes you want to build. This is done by one or more index options.

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

For example:

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

This would create presence, equality and substring indexes for the cn, sn, and uid attributes and an equality index for the objectClass attribute. See the configuration file section for more information on this option.

Once you've configured things to your liking, you create the primary database and associated indexes by running the slapadd(8) program:

```
slapadd -l <inputfile> -f <slapdconfigfile> [-d <debuglevel>]
 [-n <integer>|-b <suffix>]
```

The arguments have the following meanings:

```
-l <inputfile>
```

Specifies the LDIF input file containing the entries to add in text form (Take a look on the next section).

```
-f <slapdconfigfile>
```

Specifies the slapd configuration file that tells where to create the indexes, what indexes to create, etc.

```
-d <debuglevel>
```

Turn on debugging, as specified by <debuglevel>. The debug levels are the same as for slapd. See the the section called “Command Line Options” for more details.

```
-n <databasenumber>
```

An optional argument that specifies which database to modify. The first database listed in the configuration file is 1, the second 2, etc. By default, the first database in the configuration file is used. Should not be used in conjunction with -b.

```
-b <suffix>
```

An optional argument that specifies which database to modify. The provided suffix is matched against a database suffix directive to determine the database number. Should not be used in conjunction with -n.

Sometimes it may be necessary to regenerate indices (such as after modifying slapd.conf(5)). This is possible using the slapindex(8) program. slapindex is invoked like this:

```
slapindex -f <slapdconfigfile> [-d <debuglevel>] [-n <databasenumber>|-b <suffix>]
```

Where the -f, -d, -n and -b options are the same as for the slapadd(1) program. slapindex rebuilds all indices based upon the current database contents.

The slapcat program is used to dump the database to an LDIF file. This can be useful when you want to make a human-readable backup of your database or when you want to edit your database off-line. The program is invoked like this:

```
slapcat -l <filename> -f <slapdconfigfile> [-d <debuglevel>] [-n <databasenumber>|
```

where -n or -b is used to select the database in the slapd.conf(5) specified using -f. The corresponding LDIF output is written to standard output or to the file specified using the -l option.

More on the LDIF Format

The LDAP Data Interchange Format (LDIF) is used to represent LDAP entries in a simple text format. The basic form of an entry is:

```
#comment
dn: <distinguished name>
<attrdesc>: <attrvalue>
<attrdesc>: <attrvalue>
...
```

Lines starting with a '#' character are comments. An attribute description (attrdesc) may be a simple attribute type like cn or objectClass or 1.2.3 (an OID associated with an attribute type) or may include options such as cn;lang_en_US or userCertificate;binary.

A line may be continued by starting the next line with a single space or tab character. For example:

```
dn: cn=Barbara J Jensen, dc=example, dc=
 com
cn: Barbara J
    Jensen
```

is equivalent to:

```
dn: cn=Barbara J Jensen, dc=example, dc=com
cn: Barbara J Jensen
```

Multiple attribute values are specified on separate lines. e.g.,

```
cn: Barbara J Jensen
cn: Babs Jensen
```

If an <attrvalue> contains non-printing characters or begins with a space, a double colon (':'), or a less than ('<'), the <attrdesc> is followed by a double colon and the base64 encoding of the value. For example, the value " begins with a space" would be encoded like this:

```
cn:: IGJlZ2lucyB3aXRoIGEgc3BhY2U=
```

You can also specify a URL containing the attribute value. For example, the following specifies the jpegPhoto value should be obtained from the file /path/to/file.jpeg.

```
cn:< file://path/to/file.jpeg
```

Multiple entries within the same LDIF file are separated by blank lines. Here's an example of an LDIF file containing three entries.

```
# Barbara's Entry
dn: cn=Barbara J Jensen, dc=example, dc=com
cn: Barbara J Jensen
cn: Babs Jensen
objectClass: person
sn: Jensen

# Bjorn's Entry
dn: cn=Bjorn J Jensen, dc=example, dc=com
cn: Bjorn J Jensen
cn: Bjorn Jensen
objectClass: person
sn: Jensen
# Base64 encoded JPEG photo
jpegPhoto:: /9j/4AAQSkZJRgABAAAAAQABAAD/2wBDABALD
A4MChAODQ4SERATGCgaGBYWGDEjJR0oOjM9PDkzODdASFxOQ
ERXRTc4UG1RV19iZ2hnPk1xeXBkeFxlZ2P/2wBDARESEhgVG

# Jennifer's Entry
dn: cn=Jennifer J Jensen, dc=example, dc=com
cn: Jennifer J Jensen
cn: Jennifer Jensen
objectClass: person
sn: Jensen
# JPEG photo from file
jpegPhoto:< file://path/to/file.jpeg
```

Notice that the jpegPhoto in Bjorn's entry is base 64 encoded and the jpegPhoto in Jennifer's entry is obtained from the location indicated by the URL.

Trailing spaces are not trimmed from values in an LDIF file. Nor are multiple internal spaces compressed. If you don't want them in your data, don't put them there.

The ldapsearch, ldapdelete and ldapmodify utilities

ldapsearch - ldapsearch is a shell accessible interface to the ldap_search(3) library call. Use this utility to search for entries on your LDAP database backend.

The synopsis to call ldapsearch is the following (take a look at the ldapsearch man page to see what each option means):

```
ldapsearch  [-n]  [-u]  [-v]  [-k]
[-K]  [-t]  [-A] [-B] [-L]
[-R] [-d debuglevel] [-F sep] [-f file]
[-x] [-D binddn]  [-W]  [-w bindpasswd]
[-h ldaphost]  [-p ldapport]   [-b searchbase]
[-s base|one|sub]
[-a never|always|search|find] [-l timelimit]
[-z sizelimit] filter [attrs...]
```

ldapsearch opens a connection to an LDAP server, binds, and performs a search using the filter *filter*. The filter should conform to the string representation for LDAP filters as defined in RFC 1558. If ldapsearch finds one or more entries, the attributes specified by *attrs* are retrieved and the entries and values are printed to standard output. If no attrs are listed, all attributes are returned.

```
ldapsearch -x -b 'o=TUDelft,c=NL' 'objectclass=*'

ldapsearch -b 'o=TUDelft,c=NL' 'cn=Rene van Leuken'

ldasearch -u -b 'o=TUDelft,c=NL' 'cn=Luiz Malere' sn mail
```

The -b option stands for searchbase (initial search point), the -u option stands for userfriendly output information and the -x option is used to specify simple authentication.

ldapdelete - ldapdelete is a shell accessible interface to the ldap_delete(3) library call. Use this utility to delete entries on our LDAP database backend.

The synopsis to call ldapdelete is the following (take a look at the ldapdelete man page to see what each option means):

```
ldapdelete   [-n]   [-v]  [-k]  [-K]
[-c]  [-d debuglevel]  [-f file]  [-D binddn]
[-W]  [-w passwd] [-h ldaphost] [-p ldapport]
[dn]...
```

ldapdelete opens a connection to an LDAP server, binds, and deletes one or more entries. If one or more dn arguments are provided, entries with those Distinguished Names are deleted. Each dn should be a string-represented DN as defined in RFC 1779. If no dn arguments are provided, a list of DNs is read from standard input (or from file if the -f flag is used).

Here are some examples of the use of ldapdelete:

```
ldapdelete 'cn=Luiz Malere,o=TUDelft,c=NL'

ldapdelete -v 'cn=Rene van Leuken,o=TUDelft,c=NL' -D 'cn=Luiz Malere,o=TUDelft,c=N
```

The -v option stands for verbose mode, the -D option stands for Binddn (the dn to authenticate against) and the -W option stands for password prompt.

ldapmodify - ldapmodify is a shell accessible interface to the ldap_modify(3) and ldap_add(3) library calls. Use this utility to modify entries on our LDAP database backend.

The synopsis to call ldapmodify is the following (take a look at the ldapmodify man page to see what each option mean):

```
ldapmodify   [-a]  [-b]  [-c]  [-r]
[-n]  [-v]  [-k]  [-d debuglevel]
[-D binddn]  [-W]  [-w passwd]
[-h ldaphost] [-p ldapport] [-f file]

ldapadd [-b] [-c] [-r] [-n]
[-v]  [-k]  [-K]  [-d debuglevel]
[-D binddn]  [-w passwd]  [-h ldaphost]
[-p ldapport] [-f file]
```

ldapadd is implemented as a hard link to the ldapmodify tool. When invoked as ldapadd the -a (add new entry) flag of ldapmodify is turned on automatically. ldapmodify opens a connection to an LDAP server, binds, and modifies or adds entries. The entry information is read from standard input or from file through the use of the -f option.

Here are some examples of the use of ldapmodify:

Assuming that the file /tmp/entrymods exists and has the contents:

```
dn: cn=Modify Me, o=University of Michigan, c=US
changetype: modify
replace: mail
mail: modme@terminator.rs.itd.umich.edu
-
add: title
title: Grand Poobah
-
add: jpegPhoto
jpegPhoto: /tmp/modme.jpeg
-
delete: description
-
```

The command:

```
ldapmodify -b -r -f /tmp/entrymods
```

will replace the contents of the "Modify Me" entry's mail attribute with the value "modme@terminator.rs.itd.umich.edu", add a title of "Grand Poobah", and the contents of the file /tmp/modme.jpeg as a jpegPhoto, and completely remove the description attribute.

The same modifications as above can be performed using the older ldapmodify input format:

```
cn=Modify Me, o=University of Michigan, c=US
mail=modme@terminator.rs.itd.umich.edu
+title=Grand Poobah
+jpegPhoto=/tmp/modme.jpeg
-description
```

And plus the command bellow:

```
ldapmodify -b -r -f /tmp/entrymods
```

Assuming that the file /tmp/newentry exists and has the contents:

```
dn: cn=Barbara Jensen, o=University of Michigan, c=US
objectClass: person
cn: Barbara Jensen
cn: Babs Jensen
sn: Jensen
title: the world's most famous manager
mail: bjensen@terminator.rs.itd.umich.edu
uid: bjensen
```

The command:

```
ldapadd -f /tmp/entrymods
```

will add the entry with dn: cn=Barbara Jensen, o=University of Michigan, c=US if it's not already present. If an entry with this dn already exists, the command will point out the error and will not overwrite the entry.

Assuming that the file /tmp/newentry exists and has the contents:

```
dn: cn=Barbara Jensen, o=University of Michigan, c=US
changetype: delete
```

The command:

```
ldapmodify -f /tmp/entrymods
```

will remove Babs Jensen's entry.

The -f option stands for file (read the modification information from a file instead of standard input), the -b option stands for binary (any values starting with a '/' on the input file are interpreted as binaries), the -r stands for replace (replace existing values by default).

Chapter 6. Additional Information and Features

In this section you will find additional information and useful references for topics like authentication, logs and LDAP clients. At the end of the section there are also some very nice generic URL's and book recomendations about the subject LDAP.

LDAP Migration Tools

The LDAP Migration Tools are a collection of Perl scripts provided by PADL Software Ltd. They are used to convert configuration files to the LDIF format. I recommend reading the license terms before using them, even being free. If you plan to use your LDAP server to authenticate users, this tools may be very useful. Use the Migration Tools to convert your NIS or password archives to the LDIF format, making these files compatible with your LDAP Server. Also apply these Perl scripts to migrate users, groups, aliases, hosts, netgroups, networks, protocols, RPCs and services from existing nameservices (NIS, flat files and NetInfo) to the LDIF format.

To download the LDAP Migration Tools and get more information, go to the following address: http://www.padl.com/tools.html .

The package comes with a README file and the name of the script files are intuitive. Take a first look on the README file and then start applying the scripts.

Another recommended URL with migration tools is:

http://dataconv.org/apps_ldap.html

Authentication using LDAP

To access the LDAP service, the LDAP client first must authenticate itself to the service. That is, it must tell the LDAP server who is going to be accessing the data so that the server can decide what the client is allowed to see and do. If the client authenticates successfully to the LDAP server, then when the server subsequently receives a request from the client, it will check whether the client is allowed to perform the request. This process is called access control.

In LDAP, authentication is supplied in the "bind" operation. Ldapv3 supports three types of authentication: anonymous, simple and SASL authentication. A client that sends a LDAP request without doing a "bind" is treated as an anonymous client. Simple authentication consists of sending the LDAP server the fully qualified DN of the client (user) and the client's clear-text password. This mechanism has security problems because the password can be read from the network. To avoid exposing the password in this way, you can use the simple authentication mechanism within an encrypted channel (such as SSL), provided that this is supported by the LDAP server.

Finally, SASL is the Simple Authentication and Security Layer (RFC 2222). It specifies a challenge-response protocol in which data is exchanged between the client and the server for the purposes of authentication and establishment of a security layer on which to carry out subsequent communication. By using SASL, LDAP can support any type of authentication agreed upon by the LDAP client and server. The Cyrus-SASL package is available at the following URL: http://asg.web.cmu.edu/sasl/sasl-library.html.

Further on authenticating users to access information from your Directory Tree, your LDAP server can authenticate users from other services too (Sendmail, Login, Ftp, etc.). This is accomplished migrating specific user information to your LDAP server and using a mechanism called PAM (Pluggable Authenti-

cation Module). The authentication module for LDAP is available as a tar ball on the following address: http://www.padl.com/OSS/pam_ldap.html

SASL Configuration: Digest-MD5

I've got LDAP-SASL authentication running using the DIGEST-MD5 mechanism. To accomplish that, I've followed strictly the steps listed bellow:

- Downloaded SleepyCat 4.2.52, compiling and building manually. After downloading, I've just followed the instructions listed on the file docs/index.html under the directory where I've unpacked the .tar.gz bundle.

  After unpacking you can run the suggested:

  ```
  root@rdnt03:/usr/local/BerkeleyDB.4.2/build_unix#../dist/configure
  root@rdnt03:/usr/local/BerkeleyDB.4.2/build_unix#make
  root@rdnt03:/usr/local/BerkeleyDB.4.2/build_unix#make install
  ```

- Downloaded Cyrus SASL 2.1.17, unpacking and following the instructions listed on the document doc/install.html, under the directory where I've unpacked the .tar.gz file. Here there's a point of attention, you need to run the configure script using some env parameters:

  ```
  root@rdnt03:/usr/local/cyrus-sasl-2.1.17#env CPPFLAGS="-I/usr/local/BerkeleyDB.4
  LDFLAGS="-L/usr/local/BerkeleyDB.4.2/lib" ./configure
  ```

  The CPPFLAGS and LDFLAGS environment parameters should point to the respective include and lib directories where Berkeley BDB was installed.

  After that you can run the suggested:

  ```
  root@rdnt03:/usr/local/cyrus-sasl-2.1.17#make
  root@rdnt03:/usr/local/cyrus-sasl-2.1.17#make install
  root@rdnt03:/usr/local/cyrus-sasl-2.1.17#ln -s /usr/local/lib/sasl2 /usr/lib/sas
  ```

- Finally, I've installed OpenLDAP 2.2.5 using the same directions listed on this document, just running the configure script the same way as SASL's configure:

  ```
  root@rdnt03:/usr/local/openldap-2.2.5#env CPPFLAGS="-I/usr/local/BerkeleyDB.4.2/i
  LDFLAGS="-L/usr/local/BerkeleyDB.4.2/lib" ./configure
  ```

  After that, I've run the suggested:

  ```
  root@rdnt03:/usr/local/openldap-2.2.5#make depend
  root@rdnt03:/usr/local/openldap-2.2.5#make
  root@rdnt03:/usr/local/openldap-2.2.5#make install
  ```

- Next, I've created the sasl user database:

  ```
  root@rdnt03:~# saslpasswd2 -c admin
  ```

  You'll be prompted for a password. Remember that the username should not be a DN (distinguished name). Also remember to use the same password as your admin entry on the directory tree.

- Now, you should set the sasl-regexp directive in the *slapd.conf* file before starting the slapd daemon and testing the authentication. My *slapd.conf* file resides at /usr/local/etc/openldap:

  ```
  sasl-regexp uid=(.*),cn=rdnt03,cn=DIGEST-MD5,cn=auth uid=$1,ou=People,o=Ever
  ```

This parameter is in the format of:

uid=<username>,cn=<realm>,cn=<mech>,cn=auth

The username is taken from sasl and inserted into the ldap search string in the place of $1.Your realm is supposed to be your FQDN (fully qualified domain name), but in some cases it isn't, like mine. To find out what your realm is do:

```
root@rdnt03:~# sasldblistusers2
admin@rdnt03: userPassword
admin@rdnt03: cmusaslsecretOTP
```

In my case, *rdnt03* is indicated as the realm. If it is your FQDN you shouldn't have any problems. I use the following LDIF file:

```
dn: o=Ever
o: Ever
description: Organization Root
objectClass: top
objectClass: organization

dn: ou=Staff, o=Ever
ou: Staff
description: These are privileged users that can interact with Organization produ
objectClass: top
objectClass: organizationalUnit

dn: ou=People, o=Ever
ou: People
objectClass: top
objectClass: organizationalUnit

dn: uid=admin, ou=Staff, o=Ever
uid: admin
cn: LDAP Adminstrator
sn: admin
userPassword: {SHA}5en6G6MezRroT3XKqkdPOmY/BfQ=
objectClass: Top
objectClass: Person
objectClass: Organizationalperson
objectClass: Inetorgperson

dn: uid=admin,ou=People,o=Ever
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
userPassword: {SHA}5en6G6MezRroT3XKqkdPOmY/BfQ=
displayName: admin
mail: admin@eversystems.com.br
uid: admin
cn: Administrator
sn: admin
```

Add the entries to your LDAP directory using the following command:

```
slapadd -c -l Ever.ldif -f slapd.conf -v -d 256
```

- Now, start the *slapd* daemon and run a query using the *ldapsearch* command:

```
root@rdnt03:~# ldapsearch -U admin@rdnt03 -b 'o=Ever' '(objectclass=*)'
SASL/DIGEST-MD5 authentication started
Please enter your password:
SASL username: admin@rdnt03
SASL SSF: 128
SASL installing layers
...
Entries
...
```

That's it ! If you prefer to use SASL with Kerberos V or GSSAPI, there's a useful link at http://www.openldap.org/doc/admin22/sasl.html. This link assumes you've already managed to install and configure the SASL library. The mailing lists will help you get going with this matter: http://asg.web.cmu.edu/sasl/index.html#mailinglists

Graphical LDAP tools

Kldap is a graphical LDAP client written for KDE. Kldap has a nice interface and is able to show all the information tree stored on your Directory. You can check some screenshots of the application and download it at: http://www.mountpoint.ch/oliver/kldap/

KDirAdm is a LDAP Directory management tool written for the KDE Desktop Environment version 2 or later. It aims to provide all of the functionality of most commercial directory management tools: http://www.carillonis.com/kdiradm/

Directory Administrator is the most widely used GNOME application for managing UNIX users and groups on LDAP directory servers. Directory administrator allows you to create and delete users and groups, and manage your users associated address book information, per-server access controls and Sendmail mail routing: http://diradmin.open-it.org/index.php

GQ is another graphical LDAP client with a simpler interface. It was written for GNOME. It also runs under KDE, the same way Kldap runs under GNOME. The address for downloading and getting more information is: http://biot.com/gq/

LDAP Browser/Editor: This tool is fantastic, it has complete administrative and browsing funcionalities. Check it out: Ldap Browser [http://www.iit.edu/~gawojar/ldap/].

Logs

Slapd uses the syslog(8) facility to generate logs. The default user of the syslog(8) facility is LOCAL4, but values from LOCAL0, LOCAL1, up to LOCAL7 are allowed.

In order to enable the generation of logs you have to edit your syslog.conf file, usually located in the /etc directory.

Create a line like this:

```
local4.*     /usr/adm/ldaplog
```

This will use the default user LOCAL4 for the syslog facility. If you are not familiar with the syntax of this line, take a look at the man pages of syslog, syslog.conf and syslogd. If you want to specify the level of the logs generated or to change the default user, you have the following options while starting slapd:

`-s syslog-level`

This option tells slapd at what level debugging statements should be logged to the syslog(8) facility. The level describes the severity of the message, and is a keyword from the following ordered list (higher to lower): emerg, alert, crit, err, warning, notice, info, and debug. Ex: slapd -f myslapd.conf -s debug

`-l syslog-local-user`

Selects the local user of the syslog(8) facility. Values can be LOCAL0, LOCAL1, and so on, up to LOCAL7. The default is LOCAL4. However, this option is only permitted on systems that support local users with the syslog(8) facility.

Now take a look at the logs generated (/usr/adm/ldaplog in the example). They can help you tremendously in solving problems with queries, updates, binding, etc.

Chapter 7. References

On this section you will find additional documentation about LDAP: useful URLs, cool books and definition RFCs.

URL's

Here are the URLs that contain very useful information about LDAP. From these URLs, this HOWTO was made, so if after reading this document you need more specific information, you probably will find here:

- University of Michigan LDAP Page: http://www.umich.edu/~dirsvcs/ldap/
- University of Michigan LDAP Documentation Page: http://www.umich.edu/~dirsvcs/ldap/doc/
- OpenLDAP Administrator's Guide (brother document): http://www.openldap.org/doc/admin
- Linux Directory Service: http://www.rage.net/ldap/
- Red Hat and LDAP: http://www.redhat.com/docs/manuals/linux/RHL-9-Manual/ref-guide/ch-ldap.html
- Mandrake Linux - Using OpenLDAP for Authentication: http://www.mandrakesecure.net/en/docs/ldap-auth.php
- Integrating OpenLDAP with other Open Source projects: ftp://kalamazoolinux.org/pub/pdf/ldapv3.pdf

Books

These are the most popular and useful books about LDAP:

- Implementing LDAP by Mark Wilcox
- LDAP: Programming Directory-Enabled Applications with Lightweight Directory Access Protocol by Howes and Smith
- Understanding and Deploying LDAP Directory Servers by Howes, Smith, and Good

RFC's

The RFCs [http://www.rfc-editor.org/rfc/] that support the LDAP development efforts:

- RFC 1558: A String Representation of LDAP Search Filters
- RFC 1777: Lightweight Directory Access Protocol
- RFC 1778: The String Representation of Standard Attribute Syntaxes
- RFC 1779: A String Representation of Distinguished Names
- RFC 1781: Using the OSI Directory to Achieve User Friendly Naming
- RFC 1798: Connectionless LDAP
- RFC 1823: The LDAP Application Programming Interface

- RFC 1959: An LDAP URL Format
- RFC 1960: A String Representation of LDAP Search Filters
- RFC 2251: Lightweight Directory Access Protocol (v3)
- RFC 2307: LDAP as a Network Information Service