LDAP Linux HOWTO

Table of Contents

Table of Contents

LDAP Linux HOWTO

Luiz Ernesto Pinheiro Malere, `malere@yahoo.com`

v1.05, 06 July 2001

Information about installing, configuring, running and maintaining a LDAP (Lightweight Directory Access Protocol) Server on a Linux machine is presented on this document. There are also details about how to create LDAP databases, how to update and delete information on the database, how to implement roaming access and how to use Netscape Address Book. This document is mostly based on the University of Michigan LDAP information pages and on the OpenLDAP Administrator's Guide.

1. Introduction

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 run a slapd which provides directory service for your local domain only, without replication, so without slurpd.

This is a simple configuration for the server, good for starting but easy to upgrade to another configuration later if you want. The information presented on this document represents a nice initialization on using the LDAP protocol. Possibly after reading this document you would 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.

1.1 What's LDAP ?

LDAP is a client−server protocol for accessing a directory service. It was initially used as a front−end to X.500, but can also be used with stand− alone and other kinds of directory servers.

1.2 What's a Directory Service ?

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. As a consequence, directories don't usually implement the complicated transaction or roll−back schemes that regular databases use for doing high−volume complex updates. Directory updates are typically simple all−or−nothing changes, if they are allowed at all.

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.

1.3 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 which 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.

1.4 LDAP backends, objects and attributes

Slapd comes with three different backend databases you can choose from. They are LDBM, a high−performance disk−based database; SHELL, a database interface to arbitrary UNIX commands or shell scripts; and PASSWD, a simple password file database.

In this document I assume that you choose the LDBM database.

The LDBM database works by assigning a compact four−byte unique identifier to each entry in the database. It uses this identifier to refer to entries in indexes. The database consists of one main index file, called id2entry, which maps from an entry's unique identifier (EID) to a text representation of the entry itself. Other index files are maintained as well.

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. An 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 LDBM 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.

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 (its 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:

- 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

Go to the first paragraph of section 3 to know where the objectclass and attribute definitions lay on your system.

1.5 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.mobilesoft.com.br/HOWTO/LDAP−HOWTO.html

for new versions of this HOWTO.

1.6 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 !

1.7 History of Releases

This section lists the releases of this document, sorted by date. Each release carries the changes introduced on the earlier version, plus newer additions and corrections:

v1.0: 20 June 1999, Initial version.

v1.01: 15 February 2000, added the following sections:

- LDAP Migration Tools
- Authentication using LDAP
- Graphical LDAP tools
- RFCs

v1.02: 13 September 2000, correction of typos and addition of the following section:

- History of Releases

v1.03: 28 September 2000, presenting OpenLDAP 2.0, which comprises LDAPv3, defined on the RFC2251.

v1.04: 28 February 2001, correction of more typos and update on the following sections:

- Roaming Access
- Authentication using LDAP

v1.05: 22 June 2001, correction of long lines that were causing inconsistences on the PDF version of the document.

1.8 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, Joshua Go, great volunteer on the LDP project and Hugo van der Kooij for his tips on the Roaming Access section.

1.9 Copyright and Disclaimer

If you have any kind of sugestion, please send me an email (I will update the document if the sugestion proceeds).

If you want a translation, for example to Portuguese, you can send me an email about it too.

No liability for the contents of this document can be accepted. I have no responsability about the consequences of following the steps provided in this document.

If you have questions, please contact, the Linux HOWTO coordinator, at

linux−howto@metalab.unc.edu

---

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 and Build the server.

2.1 Pre−Requirements

To be fully LDAPv3 compliant, OpenLDAP clients and servers require installation of some aditional packages. In my particular case I also installed OpenLdap v2.0.11 on a out−of−box RedHat 2.2.15 distribution. My intention was to figure out if the build scripts would complain about the pre−required packages. They didn't ! Anyway, this is not the rule, you might still need to obtain and install these aditinal packages to successfully build OpenLDAP v2.xx:

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.

Database Software

OpenLDAP's slapd primary database backend, LDBM, requires a compatible database package for entry storage. LDBM is compatible with Sleepycat Software's BerkeleyDB (recommended) or with the Free Software Foundation's GNU Database Manager (GDBM). If neither of these packages are available at configure time, you will not be able build slapd with primary database backend support.

If your operating system doesn't provide one of these two packages, it's necessary to obtain one of them and install it.

BerkeleyDB is available from Sleepycat Software's download page http://www.sleepycat.com/download.html. There are several versions available. At the time of this writing, the latest release, version 3.1, is recommended.

GDBM is available from FSF's download site ftp://ftp.gnu.org/pub/gnu/gdbm. At the time of this writing, version 1.8 is the latest release.

Threads

OpenLDAP is designed to take advantage of threads. OpenLDAP supports POSIX pthreads, Mach CThreads, and a number of other varieties. *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.

2.2 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 2.0.4 version of the OpenLDAP package. My operating system is a Slackware Linux with kernel 2.2.13.

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−20000704.tgz. The latest development version was openldap−2.0.4.tgz.

2.3 Unpacking the server

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.

Then use the following command:

```
tar xvzf openldap−stable.tgz
```

You can use this command too, as well:

```
gunzip openldap−stable.tgz | tar xvf −
```

2.4 Configuring the software

There are several options that you would like to customize so you can build the best software for your site.

To configure the software you just need 2 steps:

- Edit the file *ldapconfig.h.edit*, located on the subdirectory *include* beneath the directory where you unpacked the software.
- Run the configure script (if you are a tough guy, you can also edit the Make−common file instead of running the configure script :^)

In the file include/ldapconfig.h.edit you can set options like the location of the slapd and slurpd daemons. The file itself is well commented and it's default settings also reflect the most common administrator choices so, if you are in a hurry you can skip this step:

```
        vi include/ldapconfig.h.edit
```

The OpenLDAP server sources are distributed with a configuration script for setting options like instalation 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 prepar to build things on the default common location. So just type:

```
        ./configure
```

And watch the output to see if all went well

2.5 Building the server

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

```
        make depend
```

After build the server, 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 check the platform specific hints, they are located in the path doc/install/hints under the directory you unpacked the software.

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

```
        su
        make install
```

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

The binary of the OpenLdap 2.0 server is called *slapd*. OpenLdap 2.0 was officially released on August, 30th and it comprises Ldap protocol v3, as defined on the RFC 2251.

The main features of OpenLDAP 2.0 are:

- LDAPv2 and LDAPv3 Support (RFC2251−2256,2829−2831)
- Maintenance of interoperability with existing clients
- IPv4 and IPv6 support

- Strong Authentication (SASL) (RFC2829)
- Start TLS (RFC2830)
- Language Tags (RFC2596)
- DNS−based service location (RFC2247+"locate" I−D)
- Enhanced Standalone Server
- Named References/ManageDsaIT ("nameref" I−D)
- Enhanced Access Control subsystem
- Thread pooling
- Preemptive threading support
- Multiple listener support
- LDIFv1 (RFC2849)
- Improved platform/subsystem detection

Note: There will be a document on the Linux Documentation Project (LDP) called LDAP Implementation HOWTO. This document will be a great resource for those who want to explore the new features of OpenLDAP 2.0. The date for it's release is around December 2000.

On the latest versions of the OpenLDAP package, it's also possible to test the recently built binaries. The package comes with a test script, which you can run using the command:

```
make test
```

If anything goes wrong with the script you can just abort it hitting Ctrl−C. In my case, the script stopped working before it's total completion. Anyway I still could see some successfull messages about my OpenLDAP configuration.

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 on *slapd.conf*. For a complete list, see slapd.conf(5) manual page. The configuration file directives are separated into global, backend−specific and data−specific categories. Here you will find descriptions of directives, together with their default values (if any) and with examples of their use.

3.1 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. 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 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.

3.2 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 Access Control examples for more details.
```

attributetype <RFC2252 Attribute Type Description>

```
This directive defines an attribute type.
```

defaultaccess { none | compare | search | read | write }

```
This directive specifies the default access to grant requesters when no access
directives have been specified. Any given access level implies all lesser
access levels (e.g., read access implies search and compare but not write).

Default:
defaultaccess read
```

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:

−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
64 configuration file processing
128 access control list processing
256 stats log connections/operations/results
512 stats log entries sent
1024 print communication with shell backends
2048 print 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.
```

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
```

3.3 General Backend Options

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 ldbm, shell, passwd, or other supported backend type.
```

3.4 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 ldbm, shell, passwd, or other supported database
       type.

       Example:
       database ldbm

       This marks the beginning of a new LDBM 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

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

This directive specifies a replication site for this database. The host= parameter specifies 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) is used.

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, typically given as a rootdn in the slave's config file. It must also match the updatedn directive in the slave slapd's config file. 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 mech 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.

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.

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@EXAMPLE.COM"
```

rootpw <password>

```
This directive specifies a password for the DN given above that will always
work, regardless of whether an entry with the given DN exists or has a
password. This directive is deprecated in favor of SASL based authentication.

Example:
rootpw 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.
```

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(8) 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@EXAMPLE.COM"
```

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

3.5 LDBM Backend−Specific 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" line.

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 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−ldbm
```

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

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

mode <integer>

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

Default:
mode 0600
```

3.6 Other Backend Databases

slapd supports a number of backend database types besides the default LDBM:

- ldbm: Berkeley or GNU DBM compatible backend
- passwd: Provides read−only access to /etc/passwd
- shell: Shell (extern program) backend
- sql: SQL Programmable backend

Take a look on the slapd.conf(5) manpage for details.

3.7 Access Control Examples

The access control facility presented on section 3.2 is quite powerful. This section shows some examples of its use. First, some simple examples:

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

This access directive grants read access to everyone. If it appears alone it is the same as the following defaultaccess line.

```
defaultaccess read
```

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=University of Michigan, c=US" subtree, to which search access is granted. 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.

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

```
access to dn=".*, o=U of M, c=US" attr=homePhone
by self write
by dn=".*, o=U of M, c=US" search
by domain=.*\.umich\.edu read
by * compare
access to dn=".*, o=U of M, c=US"
by self write
by dn=".*, o=U of M, c=US" search
by * none
```

This example applies to entries in the "o=U of M, c=US" subtree. To all attributes except homePhone, the entry itself can write them, other U−M entries can search by them, anybody else has no access. The homePhone attribute is writable by the entry, searchable by other U−M entries, readable by clients connecting from somewhere in the umich.edu domain, and comparable by everybody else.

Sometimes it is usefull 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 too 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.

Note that the attr=member construct in the <what> clause is a shorthand for the clause "dn=* attr=member" (i.e., it matches the member attribute in all entries).

Note: Take a look on OpenLDAP Administrator's Guide at http://www.openldap.org to learn more about Access Control on Ldap.

3.8 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 LDBM 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 containing 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 is used only if no database access controls match or when the target objects are not under the control of any database (such as the Root DSE).

The next section of the configuration file defines an LDBM 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.    # ldbm definition for the example.com
 6.    database ldbm
 7.    suffix "dc=example, dc=com"
 8.    directory /usr/local/var/openldap
 9.    rootdn "cn=Manager, dc=example, dc=com"
10.    rootpw secret
11.    # replication directives
12.    replogfile /usr/local/var/openldap/slapd.replog
13.    replica host=slave1.example.com:389
14.            binddn="cn=Replicator, dc=example, dc=com"
```

```
        15.             bindmethod=simple credentials=secret
        16.     replica host=slave2.example.com
        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,approx,sub
        22.     index objectClass eq
        23.     # ldbm access control definitions
        24.     access to attr=userPassword
        25.             by self write
        26.             by anonymous auth
        27.             by dn="cn=Admin,dc=example,dc=com" write
        28.             by * none
        29.     access to *
        30.             by dn="cn=Admin,dc=example,dc=com" write
        31.             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.

Lines 11 through 18 are for replication. Line 11 specifies the replication log file (where changes to the database are logged − this file is written by slapd and read by slurpd). Lines 12 through 14 specify the hostname and port for a replicated host, the DN to bind as when performing updates, the bind method (simple) and the credentials (password) for the binddn. Lines 15 through 18 specify a second replication site.

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

Lines 24 through 31 specify access control for entries in the database. For all 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 "admin" entry and may be read by authenticated users.

The next section of the example configuration file defines another LDBM database. This one handles queries involving the dc=example,dc=net subtree. Note that without line 37, the read access would be allowed due to the global access rule at line 4.

```
        32.     # ldbm definition for example.net
        33.     database ldbm
        34.     suffix "dc=example, dc=net"
        35.     directory /usr/local/var/ldbm−example−net
        36.     rootdn "cn=Manager, dc=example, dc=com"
        37.     access to * by users read
```

---

4. Running the LDAP Server

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.

4.1 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.
```

−u user −g group

```
These options specify the user and group, respectively, to run 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:

−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
64  configuration file processing
128  access control list processing
256  stats log connections/operations/results
512  stats log entries sent
1024  print communication with shell backends
2048  print entry parsing 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.
```

4.2 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.

4.3 Killing the LDAP server

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

kill −TERM `cat $(ETCDIR)/slapd.pid`

Killing slapd by a more drastic method may cause its LDBM 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

You can change the location of this pid file by changing the SLAPD_PIDFILE variable in include/ldapconfig.h.edit

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

You can change the location of the args file by changing the SLAPD_ARGSFILE variable in include/ldapconfig.h.edit.

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).

The second method of database creation is to do it off−line, using the index generation tools. 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.

5.1 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. 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 section 3, 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 <directory>

For example:

directory /usr/local/tudelft

You need to make it so you can connect to slapd as somebody with permission to add entries. This is done through the following two options in the database definition:

rootdn <dn>

rootpw <passwd> /* Remember to use crypto or 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.

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

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

index {<attrlist> | default} [pres,eq,approx,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

index objectclass pres,eq

index default none

Once you have configured things to your liking, start up slapd, connect with your LDAP client, and start adding entries. For example, to add a 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 −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". 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 −D "cn=Manager, o=TUDelft, c=NL" −W
Enter LDAP Password:
```

5.2 Creating a Database offline

The second method of database creation is to do it off−line, using the index generation 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. They produce the LDBM index files directly. 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 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 <directory>

For example:

directory /usr/local/tudelft

Next, you probably want to increase the size of the in−core cache used by each open index file. For best performance during index creation, the entire index should fit in memory. If your data is too big for this, or your memory too small, you can still make it pretty big and let the paging system do the work. This size is set with the following option:

dbcachesize <integer>

For example:

dbcachesize 50000000

This would create a cache 50 MB big, which is pretty big (at University of Michigan, the database has about 125K entries, and the biggest index file is about 45 MB). Experiment with this number a bit, and the degree of parallelism (explained below), to see what works best for your system. Remember to turn this number back down once your index files are created and before you run slapd.

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,approx

index default none

This would create presence, equality and approximate indexes for the cn, sn, and uid attributes, and no indexes for any other attributes. See the configuration file on section 3 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

Command−Line Options section in Running slapd.

```
       −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 ldbm 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.

There is another program called slapcat that 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>|−b <suffix>]

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.

5.3 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 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.

5.4 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 our LDAP database backend.

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

```
ldapsearch  [−n]  [−u]  [−v]  [−k]
[−K]  [−t]  [−A] [−B] [−L]
[−R] [−d debuglevel] [−F sep] [−f file]
[−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.

Here are some examples of use of ldapsearch:

```
ldapsearch −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) and the −u option stands for userfriendly output information.

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 mean):

```
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 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=NL' −W
```

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 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).

---

6. Additional Information and Features

In this section you will find information about the Netscape Address Book, a LDAP client that can be used to query your Directory. Also presented are details on how to implement Roaming Access using the Netscape Navigator, version 4.5 or above and your LDAP server. The purpose of introducing these features here is more for giving people an idea about the capabilities of the LDAP protocol. To finish you will see some information about authentication using LDAP, LDAP migration tools, LDAP graphical tools, slapd logs and about safely killing the slapd process.

6.1 Roaming Access

The goal of Roaming Access is that wherever you are on the Net, you can retrieve your bookmarks, preferences, mail filters, etc. using Netscape Navigator and a LDAP server. This is a very nice feature. Imagine that wherever you access the Web, you can have your own settings on the browser. If you will travel and you need to access that currency site that is stored on your local bookmarks, don't worry. Upload the bookmarks and other configuration files to a LDAP server and you can retrieve them all later, independent of the place you will be.

To implement Roaming Access you have to follow these steps:

- Include a new schema file on your slapd.conf configuration file
- Set the modification field at the database section of your slapd.conf configuration file
- Change you Ldif file adding profile entries for the users that want to use Roaming Access
- Configure Netscape Navigator to use the LDAP server as a Roaming Access Server
- Restart the LDAP server with the new settings.

− Including a new schema file: Copy and paste the section bellow and save it as a text file with a .schema extension. Usually you would save it in the directory /usr/local/etc/openldap/schema. If you prefer, the file can be downloaded from: http://home.kabelfoon.nl/~hvdkooij/mull.schema. Remember that your slapd.conf file should include the core.schema definitions file, using the line:

```
include /usr/local/etc/schema/core.schema

#       This schema requires that the core schema is loaded

# Used to store Netscape Roaming Profile information into OpenLDAP v2.
# This stores the actual profile name into the database.
attributeType ( 1.3.6.1.4.1.7081.1.1.1
         NAME 'nsLIProfileName'
         DESC 'Store Netscape Roaming Profile name'
         EQUALITY caseIgnoreMatch
         SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )

# Used to store Netscape Roaming Profile information into OpenLDAP v2.
attributeType ( 1.3.6.1.4.1.7081.1.1.2
         NAME 'nsLIPrefs'
         DESC 'Store Netscape Roaming Profile preferences'
         EQUALITY caseExactIA5Match
         SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )

# Used to store Netscape Roaming Profile information into OpenLDAP v2.
attributeType ( 1.3.6.1.4.1.7081.1.1.3
         NAME 'nsLIElementType'
         DESC ''
         EQUALITY caseIgnoreMatch
         SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )

# Used to store Netscape Roaming Profile information into OpenLDAP v2.
attributeType ( 1.3.6.1.4.1.7081.1.1.4
         NAME 'nsLIData'
         DESC 'Store the actual data blocks'
         EQUALITY bitStringMatch
         SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )

# Used to store Netscape Roaming Profile information into OpenLDAP v2.
```

```
attributeType ( 1.3.6.1.4.1.7081.1.1.5
         NAME 'nsLIVersion'
         DESC 'Store Netscape Roaming Profile version'
         EQUALITY integerMatch
         SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 )

# Used to store Netscape Roaming Profile information into OpenLDAP v2.
# This is the base holder of the Roaming Profile and must be created before
# you try to store information into the LDAP database.
objectClass ( 1.3.6.1.4.1.7081.1.2.1
         NAME 'nsLIProfile'
         DESC 'Base holder of the NetScape Roaming Profile'
         SUP top
         MUST ( objectClass $ nsLIProfileName )
         MAY ( nsLIPrefs $ uid $ owner )
         )

# Used to store Netscape Roaming Profile information into OpenLDAP v2.
# This object class will store the actual data.
objectClass ( 1.3.6.1.4.1.7081.1.2.2
         NAME 'nsLIProfileElement'
         DESC 'Contains the actual Roaming Profile data'
         SUP top
         MUST ( objectClass $ nsLIElementType )
         MAY ( owner $ nsLIData $ nsLIVersion )
         )

# EOF
```

− Setting the modification field: To make sure Netscape can compare your local copy of the profile data against the LDAP server, you need to set modification times in the database. A simple line added in the database section of your slapd.conf file will be sufficient. Just add:

```
lastmod on
```

− Changing your Ldif file: Each user that wish to try the Roaming Access feature of Netscape needs a profile entry on the Ldif file. Look an example of a simple LDIF file with profiles entries:

```
dn: o=myOrg,c=NL
o: myOrg
objectclass: organization

dn: cn=seallers,ou=People,o=myOrg,c=NL
cn: seallers
userpassword: myPassword
objectclass: top
objectclass: person

dn: nsLIProfileName=seallers,ou=Roaming,o=myOrg,c=NL
nsLIProfileName: seallers
owner: cn=seallers,ou=People,o=myOrg,c=NL
objectclass: top
objectclass: nsLIProfile
```

This entries can be added using the ldapadd program. Probably in your case you will only need to add the entry correspondent to the roaming profile (dn: nsLIProfileName=...).

− Configuring Netscape Navigator: The next step is to configure Netscape to enable the Roaming Access against your LDAP server. Just follow the sequence:

Go to Menu Edit => Preferences => Roaming User

Now you have to first enable Roaming Access for this profile, clicking on the checkbox corresponding to this option.

Fill the username box with an appropriate value, this must be identical with the nsLIProfileName= part from the User profile entry of the LDIF file. Example: seallers

Pull down the arrow of the Roaming User option on the left side of the Preferences Window to see the suboptions of Roaming Access.

Click on Server Information, enable the option LDAP Server and fill the boxes with the following information:

```
Address: ldap://myHost/nsLIProfileName=$USERID,ou=Roaming,o=myOrg,c=NL

User DN: cn=$USERID,ou=People,o=myOrg,c=NL
```

IMPORTANT: Netscape automatically substitutes the $USERID variable for the name of the profile you selected before running the browser. So if you selected the profile seallers, it will substitute $USERID for seallers, if you selected profile gonzales, it will substitute $USERID for gonzales. If you are not familiar with profiles, run the Profile Manager application that comes on the Netscape Comunicator suite. It's an application designed to satisfy the multiple users of a browser on the same machine, so each one can have their own settings on the browser.

The final step is to restart the server. Take a look on the section 4.2 to see how you do that safely and on section 4 to see how to start it again.

6.2 Netscape Address Book

Once you have your LDAP server up and running, you can access it with many different clients (e.g. ldapsearch command line utility). A very interesting one is the Netscape Address Book. It's available from version 4.x of Netscape but you have to use the 4.5 or above version for a stable interoperation with your LDAP server.

Just follow the sequence:

Open Netscape Navigator −> Go to Communicator Menu −> Address Book

The Netscape Address Book will be launched with some default LDAP directories. You have to add your own LDAP directory too!

Go to File Menu −> New Directory

Fill the boxes with your server information. For example:

− Description: TUDelft

− LDAP Server: dutedin.et.tudelft.nl

− Server Root: o=TUDelft, c=NL

The default LDAP port is 389. Don't change it, unless you changed this option while building your server.

Now, make simple queries to your server, using the box Show Names Containing, or advanced queries, using the Search for button.

6.3 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. Apply also 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.

6.4 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. SASL use will be presented on the next version of this Howto as the installation of the Cyrus SASL library is not yet trivial.

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 Authentication Module).

Since the beginnings of UNIX, authenticating a user has been accomplished via the user entering a password and the system checking if the entered password corresponds to the encrypted official password that is stored in /etc/passwd. That was in the beginning. Since then, a number of new ways for authenticating users became popular, including more complicated replacements for the /etc/passwd file and hardware devices called Smart cards. The problem is that each time a new authentication schema is developed, it requires all the necessary programs (login, ftpd etc...) to be rewritten to support it. PAM provides a way to develop programs that are independent of authentication scheme. These programs need "authentication modules" to be attatched to them at run−time in order to work.

The authentication module for LDAP is available as a tar ball on the following address:

http://www.padl.com/pam_ldap.html

Here I assume that your Linux distribution is already PAM prepared. If not take a look at this URL: http://www.kernel.org/pub/linux/libs/pam. Various Linux distributions use different standard settings related to PAM. Usually, the PAM configuration files reside on the `/etc/pam.d/` directory. There you can find a file for each service running on your box. As an example, if you want to use the LDAP server for logging users in after your Linux boot up, you should make your Linux PAM compatible (as described in the beginning of this paragraph), install the LDAP PAM module and edit a file called login in the PAM configuration directory (/etc/pam.d/) with the following content:

```
#%PAM−1.0
auth        required     /lib/security/pam_securetty.so
auth        required     /lib/security/pam_nologin.so
auth        sufficient   /lib/security/pam_ldap.so
auth        required     /lib/security/pam_unix_auth.so try_first_pass
account     sufficient   /lib/security/pam_ldap.so
account     required     /lib/security/pam_unix_acct.so
password    required     /lib/security/pam_cracklib.so
password    required     /lib/security/pam_ldap.so
password    required     /lib/security/pam_pwdb.so use_first_pass
session     required     /lib/security/pam_unix_session.so
```

6.5 Graphical LDAP tools

- Kldap

  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 from the application and download it at:

  http://www.mountpoint.ch/oliver/kldap

- GQ

  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/

6.6 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/ldalog
```

This will use the default user LOCAL4 for the syslog facility. If you are not familiar with the sintax of this line, take a look at the man pages of syslog, syslog.conf and syslogd. If you want to change the default user or to specify the level of the logs generated, 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. They can help you tremendously in solving problems with queries, updates, binding, etc.

---

7. References

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

7.1 URLs

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/index.html
- University of Michigan LDAP Documentation Page:

  http://www.umich.edu/~dirsvcs/ldap/doc/
- OpenLDAP Administrator's Guide:

  http://www.openldap.org/doc/admin

- Manually Implementing Roaming Access:

  http://help.netscape.com/products/client/communicator/manual_roaming2.html
- Customizing LDAP Settings for Communicator 4.5:

  http://developer.netscape.com/docs/manuals/communicator/ldap45.htm
- Introducing to Directory Service (X.500):

  http://www.nic.surfnet.nl/surfnet/projects/x500/introducing/
- Linux Directory Service:

  http://www.rage.net/ldap/

7.2 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

7.3 RFCs

The RFCs 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