LDAP HOWTO

Luiz Ernesto Pinheiro Malere, malere@yahoo.com v1.0, 20 June 1999

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.

Contents

1 Introduction **2**

1.1 What's LDAP ? . . . 2

1.2 What's a Directory Service ? . . . 2

1.3 How does LDAP work ? . . . 3

1.4 LDAP backends, objects and attributes . . . 3

1.5 New Versions of this Document . . . 4

1.6 Opinions and Sugestions . . . 5

1.7 Copyright and Disclaimer . . . 5

2 Installing the LDAP Server **5**

2.1 Downloading the package . . . 5

2.2 Unpacking the server . . . 5

2.3 Configuring the software . . . 6

2.4 Building the server . . . 6

3 Configuring the LDAP Server **7**

3.1 Configuration File Format . . . 7

3.2 Global Options . . . 7

3.3 General Backend Options . . . 9

3.4 LDBM Backend-Specific Options . . . 11

3.5 Access Control Examples . . . 12

4 Running the LDAP Server **13**

4.1 Command Line Options . . . 13

4.2 Running slapd as a Stand Alone Daemon . . . 14

4.3 Running slapd from inetd . . . 14

5 Database Creation and Maintenance **15**

5.1 Creating a Database online 15

5.2 Creating a Database offline 16

5.3 More on the LDIF format 17

5.4 The ldapsearch, ldapdelete and ldapmodify utilities 19

6 Aditional Informations and Features **21**

6.1 Roaming Access 21

6.2 Netscape Address Book 23

6.3 Killing the LDAP server 24

6.4 Logs 24

7 References **25**

1 Introduction

The main purpose of this document is to setup 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 avaiable 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 like 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 3 (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.ddtc.dimes.tudelft.nl/~malere/LDAP-Linux-HOWTO.html <http://www.ddtc.dimes.tudelft.nl/~malere/LDAP-Linux-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 `<mailto:malere@yahoo.com>`

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

1.7 Copyright and Disclaimer

The LDAP Linux HOWTO is Copyrighted 1999 by Luiz Ernesto Pinheiro Malere. It can be distributed freely. It cannot be modified. 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 `<linux-howto@metalab.unc.edu>`

2 Installing the LDAP Server

Four steps are necessary to install the server : Download the package, Unpack the software, Configure the Makefiles and Build the server.

2.1 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 aditional documentation avaiable for it. This document supposes that you are using the OpenLDAP server.

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

http://www.openldap.org `<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 `<ftp://terminator.rs.itd.umich.edu/ldap>`

To write this document, I used the OpenLDAP latest stable version and OpenLDAP 1.2.3 on a Slackware Linux machine with kernel 2.0.36.

2.2 Unpacking the server

Now that you have the tar gziped 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 -
```

2.3 Configuring the software

There are several options that you should like to customize so you can build the best software to 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, and run directly:

```
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

2.4 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 3 (next) section to see how to configure the operation of your LDAP server

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.

In this directory you will also find the files *slapd.oc.conf* and *slapd.at.conf* which are included on the slad.conf file (see include option on 3.2 (section 3.2)) and that hold respectively the objectclasses and attributes definitions for the LDAP database backend. Next comes a description of the general format of the config file, followed by a detailed description of each config file option.

3.1 Configuration File Format

The slapd.conf file consists of a series of global configuration options that apply to slapd as a whole (including all database backends), followed by zero or more database backend definitions that contain information specific to a backend instance.

Global options can be overridden in a backend (for options that appear more than once, the last appearance in the slapd.conf file is used). 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:

```
# comment - these options apply to every database
<global config options>
# first database definition & config options
database <backend 1 type>
<config options specific to backend 1>
# second database definition & config options
database <backend 2 type>
<config options specific to backend 2>
# subsequent database definitions & config options
...
```

Configuration line arguments 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 '\', (e.g. '\\d')..

The distribution contains an example configuration file that will be installed in the configuration prefix directory. Also provided are slapd.at.conf, which contains many commonly used attribute definitions, and slapd.oc.conf, which contains many commonly used object class definitions.

3.2 Global Options

Options described in this section apply to all backends, unless specifically overridden in a backend definition. Option arguments that should be replaced by actual text are shown in brackets <>.

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

This option 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.

attribute <name> [<name2>] { bin | ces | cis | tel | dn }

This option associates a syntax with an attribute name. By default, an attribute is assumed to have syntax cis. An optional alternate name can be given for an attribute. The possible syntaxes and their meanings are :

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

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

This option specifies the default access to grant requesters not matched by any other access line (take a look
Access Control examples down). Note that an access level implies all lesser access levels (e.g., write access
implies read, search and compare).
Default:
defaultaccess read

include <filename>

This option 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 fil
format. You can use this option to include the files that contain the objectclass and attribute definitions of
your backend database. The LDAP software package comes with the files slapd.oc.conf and slapd.at.conf

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

loglevel <integer>

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

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
This will cause lots and lots of debugging information to be syslogged.
Default:
loglevel 256
```

objectclass <name> [requires <attrs>] [allows <attrs>]

```
This option defines the schema rules for the given object class. Used in conjunction with the schemacheck
 option.
```

referral <url>

```
This option specifies the referral to pass back when slapd cannot find a local database to handle a request.
Example:
referral ldap://ldap.itd.umich.edu
This will refer non-local queries to the LDAP server at the University of Michigan. 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.
```

schemacheck { on | off }

```
This option turns schema checking on or off. If schema checking is on, entries added or modified will be
checked to ensure they obey the schema rules implied by their object class(es) as defined by the correspondin
objectclass option(s). If schema checking is off this check is not done.
Default:
schemacheck off
```

sizelimit <integer>

```
This option specifies the maximum number of entries to return from a search operation.
Default:
sizelimit 500
```

srvtab <filename>

```
This option specifies the srvtab file in which slapd can find the kerberos keys necessary for authenticating
clients using kerberos. This option is only meaningful if you are using kerberos authentication, which must
be enabled at compile time by including the appropriate definitions in the Make-common file.
Default:
srvtab /etc/srvtab
```

timelimit <integer>

```
This option 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

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

database <databasetype>

This option marks the beginning of a new database instance definition. <databasetype> should be one of ldbm, shell, or passwd, depending on which backend will serve the database.
Example:
database ldbm
This marks the beginning of a new LDBM backend database instance definition.

lastmod { on | off }

This option controls whether slapd will automatically maintain the modifiersName, modifyTimestamp, creatorsNam and createTimestamp attributes for entries.
Default:
lastmod off

readonly { on | off }

This option puts the database into "read-only" mode. Any attempts to modify the database will return an "unwilling to perform" error.
Default:
readonly off

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

This option 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 option in the slave slapd's config file. Since DNs are likely to contain embedded spaces, the entire "binddn=<DN>" string should be enclosed in quotes.
bindmethod is either simple or kerberos, depending on whether simple password-based authentication or kerberos authentication is to be used when connecting to the slave slapd. Simple authentication requires a valid passwo be given. Kerberos authentication requires a valid srvtab file.
The credentials= parameter, which is only required if using simple authentication, gives the password for binddn on the slave slapd.
The srvtab= parameter, which is only required if using kerberos, specifies the filename which holds the kerber key for the slave slapd. If omitted, /etc/srvtab is used.

replogfile <filename>

This option specifies the name of the replication log file to which slapd will log changes. The replication lo is typically written by slapd and read by slurpd. Normally, this option 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 runnin In this case, you will need to periodically truncate the file, since it will grow indefinitely otherwise.

rootdn <dn>

This option specifies the DN of an entry that is not subject to access control or administrative limit restric for operations on this database.
Example:
rootdn "cn=Manager, o=U of M, c=US"

rootkrbname <kerberosname>

This option specifies a kerberos name for the DN given above that will always work, regardless of whether an entry with the given DN exists or has a krbName attribute. This option is useful when creating a database and also when using slurpd to provide replication service.
Example:
rootkrbname admin@umich.edu

rootpw <password>

This option 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 option is useful when creating a database and also when using slurpd to provide replication service. Avoid having cleartext password on this option. At least provide an crypto (you can use an entry of Unix /etc/passwd file) password. Slapd supports other types of encryption methods too.
Example:
rootpw secret
rootpw {crypto}encrypted_password_here

suffix <dn suffix>

This option 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 "o=University of Michigan, c=US"
Queries with a DN ending in "o=University of Michigan, c=US" 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 option is only applicable in a slave slapd. It specifies the DN allowed to make changes to the replica (typically, this is the DN slurpd binds as when making changes to the replica).

3.4 LDBM Backend-Specific Options

Options 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 option specifies the size in entries of the in-memory cache maintained by the LDBM backend database instance.
Default:
cachesize 1000

dbcachesize <integer>

This option 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 option 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

directory <directory>

```
This option specifies the directory where the LDBM files containing the database and associated indexes live.
Default:
directory /usr/tmp
```

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

```
This option specifies the indexes to maintain for the given attribute. If only an <attrlist> is given,
all possible indexes are maintained.
Example:
index cn
index sn,uid eq,sub,approx
index default none

This example causes all indexes to be maintained for the cn attribute; equality, substring, and approximate
indexes for the sn and uid attributes; and no indexes for all other attributes.
```

mode <integer>

```
This option specifies the file protection mode that newly created database index files should have.
Default:
mode 0600
```

3.5 Access Control Examples

The access control facility presented on 3.2 (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 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 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).

4 Running the LDAP Server

Slapd can be run in two different modes, stand-alone or from inetd(8). Stand-alone operation is recommended, especially if you are using the LDBM backend. This allows the backend to take advantage of caching and avoids concurrency problems with the LDBM index files. If you are running only a PASSWD or SHELL backend, running from inetd is an option.

4.1 Command Line Options

Slapd supports the following command-line options.

-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 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
65535 enable all debugging

Debugging levels are additive. That is, if you want to trace function calls and watch the config file being
processed, you would set level to the sum of those two levels (in this case, 65). Consult the <ldap.h> file
for more details.
Note that slapd must have been compiled with -DLDAP_DEBUG defined for any debugging information beyond the
two stats levels to be available.
```

-f <filename>

```
This option specifies an alternate configuration file for slapd.
```

-i

```
This option tells slapd that it is running from inetd instead of as a stand-alone server. On the following sec
about running slapd from inetd you will find more details.
```

-p <port>

```
This option specifies an alternate TCP port on which slapd should listen for connections. The default port
is 389.
```

4.2 Running slapd as a Stand Alone Daemon

In general, slapd is run like this:

$(ETCDIR)/slapd [<option>]*

where ETCDIR has the value you gave in the Make-common file or configure script during the pre-build configuration, and <option> is one of the options described above. Unless you have specified a debugging level, slapd will automatically fork and detach itself from its controlling terminal and run in the background. Any of the options given above can be given to slapd to point it at a different configuration file, listen on another port, etc.

See this example of starting slapd :

$(ETCDIR)/slapd -f /home/malere/myslapd.conf -d 255

4.3 Running slapd from inetd

First, make sure that running from inetd(8) is a good idea. If you are using the LDBM backend, it is not. If you are in a high-volume environment, the overhead of running from inetd also makes it a bad idea. Otherwise, you may proceed with the two steps necessary.

Step 1 is to add a line like this to your /etc/services file:

```
ldap 389 # ldap directory service
```

Step 2 is to add a line like this to your /etc/inetd.conf file:

```
ldap stream tcp nowait nobody $(ETCDIR)/slapd slapd -i
```

where ETCDIR has the value you gave it in the Make-common file or configure script during pre-build configuration. Finally, send inetd a HUP signal, and you should be all set.

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 3 (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 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.

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

o=TUDelft
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 3 (section 3) for more information on this option.

Once you've configured things to your liking, you create the indexes by running the ldif2ldbm program:

ldif2ldbm -i <inputfile> -f <slapdconfigfile> [-d <debuglevel>] [-j <integer>] [-n <databasenumber>] [-e <etcdir>]

The arguments have the following meanings:

-i <inputfile>

Specifies the LDIF input file containing the entries to add in text form.

-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 4.1 (section 4.1)).

-j <integer>

An optional argument that specifies that at most <integer> processes should be started in parallel when building the indexes. The default is 1. If set to a value greater than one, ldif2ldbm will create at most that many subprocesses at a time when building the indexes. A separate subprocess is created to build each attribute index. Running these processes in parallel can speed things up greatly, but beware of creating too many processes, all competing for memory and disk resources.

-n <databasenumber>

An optional argument that specifies the configuration file database for which to build indices. The first database listed is "1", the second "2", etc. By default, the first ldbm database in the configuration file is used.

-e <etcdir>

An optional argument that specifies the directory where ldif2ldbm can find the other database conversion tools it needs to execute (ldif2index and friends). The default is the installation directory set on the configure script. Look an example of using the ldif2ldbm command :

/usr/local/sbin/ldif2ldbm -i new_entries -f myslapd.conf

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:

```
[<id>]
dn: <distinguished name>
<attrtype>: <attrvalue>
<attrtype>: <attrvalue>
...
```

where <id> is the optional entry ID (a positive decimal number). Normally, you would not supply the <id>, allowing the database creation tools to do that for you. The ldbmcat program, however, produces an LDIF format that includes <id> so that new indexes created will be consistent.

A line may be continued by starting the next line with a single space or tab character. e.g.,

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

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

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

If an <attrvalue> contains a non-printing character, or begins with a space or a colon ':', the <attrtype> is followed by a double colon and the value is encoded in base 64 notation. e.g., the value " begins with a space" would be encoded like this:

```
cn:: IGJlZ2lucyB3aXRoIGEgc3BhY2U=
```

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

```
dn: cn=Barbara J Jensen, o=University of Michigan, c=US
cn: Barbara J Jensen
cn: Babs Jensen
objectclass: person
sn: Jensen

dn: cn=Bjorn J Jensen, o=University of Michigan, c=US
cn: Bjorn J Jensen
cn: Bjorn Jensen
objectclass: person
sn: Jensen

dn: cn=Jennifer J Jensen, o=University of Michigan, c=US
cn: Jennifer J Jensen
cn: Jennifer Jensen
objectclass: person
sn: Jensen
jpegPhoto:: /9j/4AAQSkZJRgABAAAAAQABAAD/2wBDABALD
A4MChAODQ4SERATGCgaGBYWGDEjJROoOjM9PDkzODdASFxOQ
ERXRTc4UG1RV19iZ2hnPk1xeXBkeFxlZ2P/2wBDARESEhgVG
...
```

Notice that the jpegPhoto in Jennifer Jensen's entry is encoded using base 64. The ldif program that comes with the OpenLDAP package can be used to produce the LDIF format.

NOTE: 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 databse 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 databse 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 databse 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
```

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 Aditional Informations and Features

On this section you will find information about the Netscape Address Book, a LDAP client that can be used to query your Directory. Also is presented details on how to implement Roaming Access using the Netscape Navigator, version 4.5 or above and your LDAP server. There have been a lot of talk on the OpenLDAP mailing lists about the Roaming Access, since this is a feature that is not totally implemented. Most part of the people don't like the way Netscape Navigator operates with the LDAP server while making downloads and uploads to it. So, if after reading this you find that the Roaming Access is not working the way you would like, nevermind, a lot of people passed through this situation already. The purpose of introducing this feature here is more for giving people an idea about the capabilities of the LDAP protocol. To finish you will see some information about killing safely the slapd process and about slapd logs.

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 a 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 travell 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 the Roaming Access you have to follow these steps :

- Change your attributes description file
- Change your objectclass description file
- Change the LDIF file to include profiles
- Configure Netscape Navigator to use the LDAP server as a Roaming Access Server

- Restart the LDAP server with the new settings.

- Changing the attributes file : You need to add new attributes on the attribute list present on the file slapd.at.conf (this is a file you include on your slapd.conf and it's normally located at /usr/local/etc/openldap) :

```
attribute       nsLIPtrURL              ces
attribute       nsLIPrefs               ces
attribute       nsLIProfileName         cis
attribute       nsLIData                bin
attribute       nsLIElementType         cis
attribute       nsLIServerType          cis
attribute       nsLIVersion             cis
```

- Changing the objectclass file : You also have to add some new classes to your slapd.oc.conf (this is another file you include on your slapd.conf and it's normally located at /usr/local/etc/openldap) in order to enable the roaming access :

```
objectclass nsLIPtr
requires
        objectclass
allows
        nsliptrurl,
        owner

objectclass nsLIProfile
requires
        objectclass,
        nsliprofilename
allows
        nsliprefs,
        uid,
        owner

objectclass nsLIProfileElement
requires
        objectclass,
        nslielementtype
allows
        owner,
        nslidata,
        nsliversion

objectclass nsLIServer
requires
        objectclass,
        serverhostname
allows
        description,
        cn,
        nsserverport,
        nsliservertype,
        serverroot
```

- Changing the LDIF file : Now you have to modify your LDIF file, adding profiles entries to each user that

wish to try the Roaming Access feature of Netscape. 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
changetype: add
objectclass: top
owner: cn=seallers,ou=People,o=myOrg,c=NL
objectclass: top
objectclass: nsLIProfile
```

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 the Roaming Access for this profile, clicking on the checkbox correspondent to this option.

- Fill the username box with an appropiate value, for instance john

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

- Click on Server Information and 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, if will substitute $USERID for gonzales. If you are not familiar with profiles, run the Profile Manager aplication that comes on the Netscape Comunicator package. It's an application designed to satisfy the multiple users of a browser on the same machine, so each one can have their on settings on the browser.

The final step is to restart the server, take a look on the 6.3 (section 6.3) to see how you do that safely and on 4 (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 diferent clients (e.g. ldapsearch command line utility). A very interesting one is the Netscape Address Book. It's avaiable 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, at least if 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 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.

6.4 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 at /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 a lot to solve problems with queries, updates, binding, etc.

7 References

I present here the URLs that contain very useful information about LDAP. From this 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* <http://www.umich.edu/dirsvcs/ldap/index.html>
- University of Michigan LDAP Documentation Page : *http://www.umich.edu/~dirsvcs/ldap/doc/* <http://www.umich.edu/dirsvcs/ldap/doc/>
- Manually Implementing Roaming Access *http://help.netscape.com/products/client/communicator/manual_roaming2.ht* <http://help.netscape.com/products/client/communicator/manual_roaming2.html>
- Customizing LDAP Settings for Communicator 4.5 : *http://developer.netscape.com/docs/manuals/communicator/ldap4* <http://developer.netscape.com/docs/manuals/communicator/ldap45.htm>