



Running OpenLDAP with Delphi

Max Kleiner
Wouter Paesen

www.softwareschule.ch/download/openldap_delphi.zip

Contents

LDAP and Directory Services

OpenLDAP installation

OpenLDAP configuration

OpenLDAP tools

LDAP security

Using SSL/TLS

Using Delphi

Distributed Directory Services

LDAP Replication

What is LDAP ?

Lightweight Directory Access Protocol

A lightweight Protocol used to Access Directory Services (X500)

LDAP is an open standard defined by the Internet Engineering Task Force (IETF). The complete definition can be found in RFC3377, RFC2251, RFC2252, RFC2253, RFC2254, RFC2255, RFC2256, RFC2829 and RFC2830

What's a Directory Service ?

A *directory service* is a system for storing and retrieving information in a tree-like structure with the following key properties:

- Optimized for reading

- Distributed storage model

- Extensible data storage types

- Advanced search capabilities

- Consistent replication possibilities

LDAP is not a database !!! (Example child - parent)

Why Lightweight ?

LDAP is a subset of the X.500 Directory Access Protocol.

X.500

Is very complex and *heavyweight*.

Uses full OSI compliant connections (7 layers)

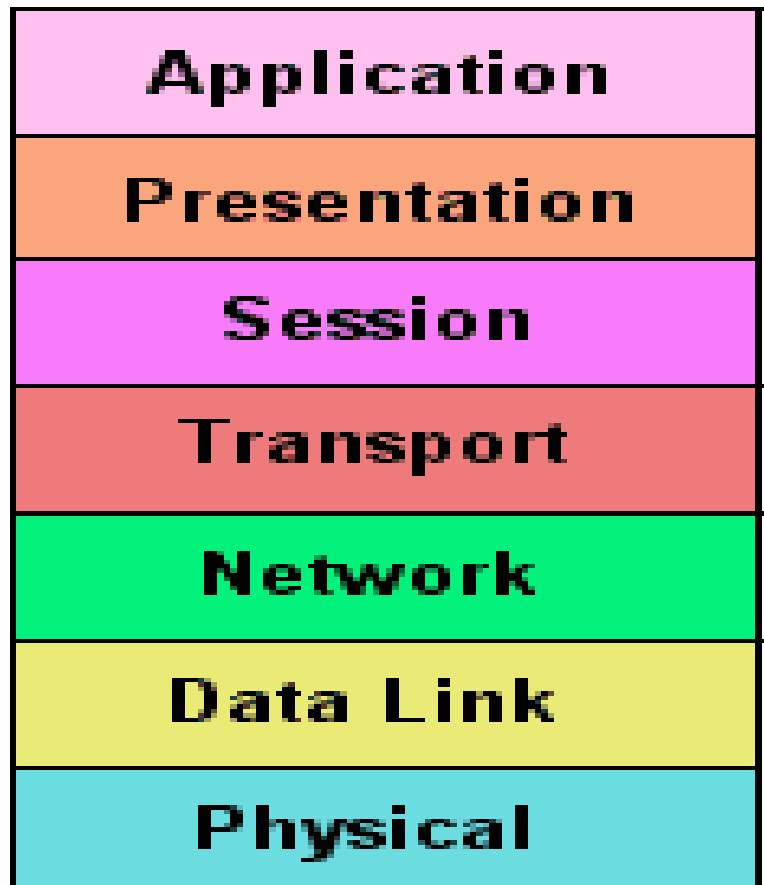
LDAP

Strives to be highly functional without being bloated with functionalities.

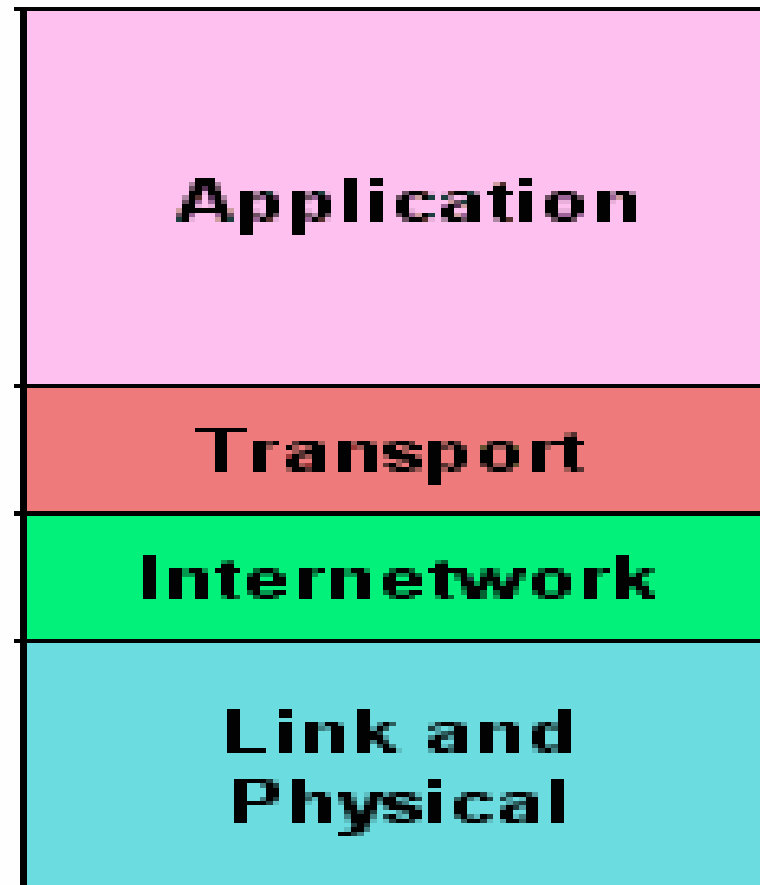
Uses TCP/IP connections (less complicated)

OSI Model - TCP/IP

OSI Model



TCP / IP



Typical LDAP Setup

LDAP Client



LDAP protocol



LDAP Server



Storage backend



LDAP Distinguished Name

Each entry has a distinguished name

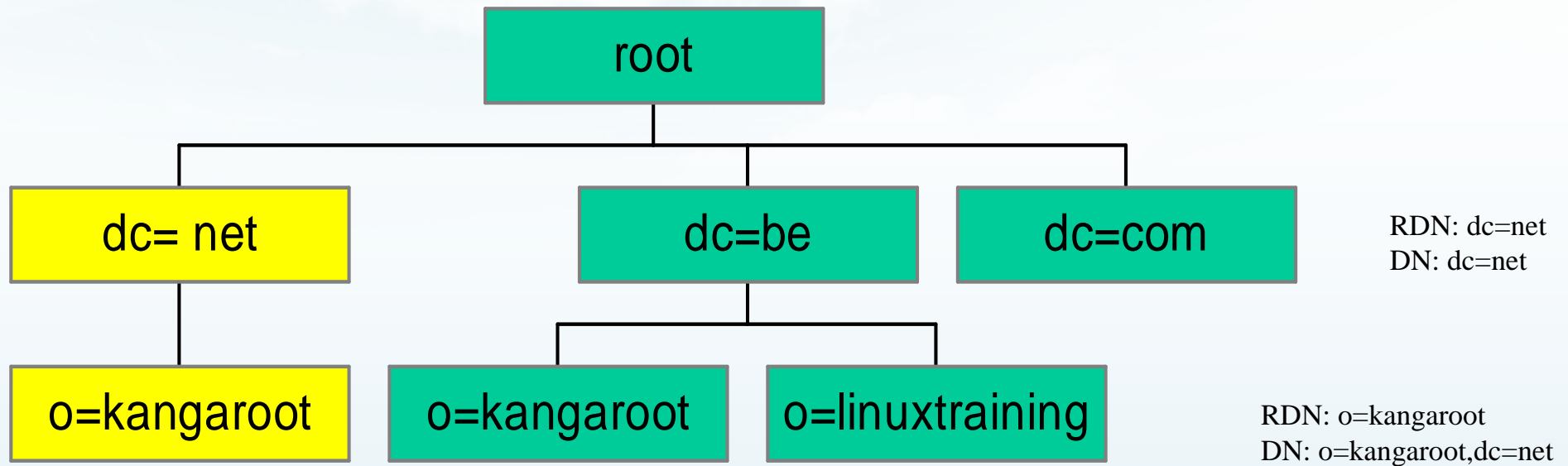
In it **hierarchy level** (horizon): Relative Distinguished Name (RDN)

All RDNs from root onwards build the Distinguished Name (DN)

No two entries in the same hierarchy level can have the same RDN

No two entries in the directory can have the same DN

LDAP Directory Information Tree



When **not** to use LDAP

When

A more specialised directory service is available
(Filesystem, DNS, ...)

A undefined mass of data, blob's, need to be stored,
here LDAP as a directory service is not the preferred
way

LDIF

LDAP Interchange Format is described in RFC 2849 (<http://www.rfc-archive.org/getrfc.php?rfc=2849>)

LDIF is a plain text representation for storing LDAP configuration information and directory contents.

An LDIF file contains:

- A collection of entries separated from each other by blank lines;

- A mapping of attribute names to values;

- A collection of directives that instruct the parser how to process the information.

Very simple syntax:

- Comments start with a pound character (#) on position 1 and continues to the end of the current line.

- Attributes are on the lefthand side of the colon (:) and values on the righthand side. The colon is separated from the value by a space.

- The dn attribute uniquely identified the DN of the entry

LDIF (example)

```
dn: dc=be  
objectClass: top  
objectClass: dcObject  
dc: be
```

```
dn: dc=net  
objectClass: top  
objectClass: dcObject  
dc: net
```

```
dn: o=kangaroot, dc=net  
objectClass: organisation  
o: kangaroot
```

LDAP Attributes

Information in the LDAP DIT is stored in attributes

Attributes

- Can be multi-valued

- Have syntax rules

- Have matching rules

- Can be required or optional

- Are defined in the schema definition

- DIT=Directory Information Tree

LDAP ObjectClass Attribute

Each LDAP entry must contain an ObjectClass attribute that:

- Defines which attributes an entry must and may contain

- Can be multi-valued

 - (Base/Object/Attribut/Type/Value1/Value2...)

LDAP ObjectClasses

Are uniquely identified by an OID (IANA)

Can be one of three types:

- Structural Object Class

- Auxiliary Object Class

- Abstract Object Class

Defines:

- Attributes

- Encoding syntaxes

- Matching rules

LDAP

Authentication

Session authentication necessary for establishing privileges

Authentication token stored in the *userPassword* attribute of the *person* objectclass

4 types of authentication defined in LDAP:

- Anonymous authentication

- Simple authentication (sends the ldap passwords in cleartext to the directory service. This is dangerous in an untrusted network)

- Simple authentication over SSL/TLS

- Simple authentication and Security Layer (SASL)

Directory Service Implementations



SunOne - Sun Microsystems



eDirectory - Novell/SUSE



Active Directory - Microsoft



OpenLDAP

Why OpenLDAP

Open source implementation

<http://www.openldap.org>

Fully LDAP v3 compliant

Available on multiple platforms

Real TCP/IP & OpenSSL Integration

Why not Finally

OpenLDAP has virtually no documentation and no comments in source code

If you have some let us have it!!!

It's hard to find examples for beginners
(exception: <http://www.delphi-jedi.org/apilibrary.html>)

Software requirements

Support for POSIX threads

(Available in GNU/Linux)

Database Manager library with DBM support

(Berkeley DB - <http://www.sleepycat.com>)

SSL/TLS libraries

(<http://www.openssl.org>)

Only if simple authentication over SSL/TLS is required

SASL libraries from Carnegie Mellon University

(<http://asg.web.cmu.edu/sasl/sasl-library.html>, Release 2.1)

Only if SASL authentication is required

OpenLDAP Installation, Methods

Installing from packages (deb, rpm)

Installing from source (tar.gz)

Install-openldap-windows.exe (2.0.27 ILEX build)

OpenLDAP Installation, Packages

A package is pre-compiled software together with pre- and post-install scripts.

- Use only packages for your architecture (e.g. i386)
- Use only packages for your distribution (e.g. Debian, Red Hat, Fedora, SuSE)
- Use only packages for the version of your distribution (e.g. Red Hat v8.0 or Red Hat v9.0)

Advantages:

- Easy to install
- Easy to maintain

Disadvantages:

- Not optimized for your environment

OpenLDAP Installation, Source

Compile OpenSSL

```
# cd /usr/src/  
# wget http://www.openssl.org/source/openssl-0.9.7i.tar.gz  
# tar zxvf openssl-09.7i.tar.gz  
# cd openssl-0.9.7i  
# ./config shared --openssldir=/usr/local  
# make  
# make install
```

for 0.9.8g:

<http://sourceforge.net/projects/delphiwebstart>
sourceforge.net/delphiwebstart/dws_https_lib_1_9.zip

OpenLDAP

Installation, Source

Compile OpenLDAP

```
# cd /usr/src/  
# wget ftp://ftp.openldap.org/pub/OpenLDAP/openldap-  
release/openldap-2.3.18.tgz  
# tar zxvf openldap-2.3.18.tgz  
# cd openldap-2.3.18  
# ./configure  
# make  
# make install
```

OpenLDAP Configuration

slapd.conf

ASCII file with some simple rules:

- Blank lines and lines beginning with a pound sign (#) are ignored.
- Parameters and associated values are separated by whitespace characters (space or tab)
- A line with a blank space in the first column is considered to be a continuation of a previous one. There is no need for a line continuation character such as a backslash.

2 sections:

- Global section
- Database section(s)

OpenLDAP Configuration, Schema

Define which schemas are supported by the server. By default schemas are:

- corba.schema
- core.schema (Basic LDAPv3 attributes and objects)
- cosine.schema
- inetorgperson.schema (Mostly used for addressbooks)
- java.schema (java serialisation)
- misc.schema (a.o. Sendmail schemas)
- nis.schema (for using LDAP with NIS)
- openldap.schema

slapd.conf

```
## Include the minimum schema required.  
include /usr/local/etc/openldap/schema/core.schema
```

OpenLDAP Configuration, Database

```
# define the storage backend to use
database bdb

# the root of our ldap tree
suffix "dc=softwareschule, dc=ch"

# definition of the database manager
rootdn "cn=manager,dc= dc=softwareschule, dc=ch"
rootpw {SSHA}2akslaicAvwc+DhCrXUFlhgWsbBJPLxy

# where the database will be stored
directory /var/ldap/softwareschule.ch

# file permissions on the database files
mode 0600

# indexes on the data
index objectClass eq
      index cn pres,eq
```

OpenLDAP Configuration, Indexes

OpenLDAP can maintain different indexes to speed up directory searches.

Defined in Database section:

index attribute **type**

Type is one (or more) of:

approx (phonetic match)

eq (exact match, as defined by matching rules)

pres (value present or not)

sub (substring matching)

OpenLDAP Configuration, ACL, Who

Or Who has Access to What?

Who:

*****: any connected user, including anonymous

self: DN of connected user

anonymous: unauthenticated users

users: all authenticated users

regular expression: matches as DN or an SASL identity

OpenLDAP Configuration, ACL, What

Or Who has Access to What?

What:

regular expression defining the target DN.

LDAP search filter

comma-separated list of attributes
(*attrs=attributeList*)

OpenLDAP Configuration, ACL, Level

Or Who has Access to What?

Level:

write

read

search

compare

auth

none

OpenLDAP Configuration, ACL, Example

```
# Restrict userPassword to be used for authentication only, but allow  
# users to modify their own passwords
```

```
access to attrs=userPassword
```

```
    by self write
```

```
    by * auth
```

```
# Simple ACL granting read access to the world
```

```
access to *
```

```
    by * read
```

**ACL's have a first match wins policy, so the more specific
ACL's should be specified first !!!**

Certificate Matching Tasks

Parse certificates/CRLs, extract fields (key usage etc.)

Define way of storing indexes on embedded fields

Add extracted fields to new indexes

Define syntaxes for AVAs

Update client to present new filter request

Perform new filtering, find the certificates or CRLs that match, then return them

AVA Notation

Certificate Base Field LDAP AVA Notation

Version number	<code>userCertificate.version</code>
Serial number	<code>userCertificate.serialNumber</code>
Algorithm Identifier	<code>userCertificate.signature.algorithm</code>
Issuer DN	<code>userCertificate.issuer</code>
Start validity time	<code>userCertificate.validity.notBefore</code>
End validity time	<code>userCertificate.validity.notAfter</code>
Subject DN	<code>userCertificate.subject</code>
Subject's public key	<code>userCertificate.subjectPublicKey</code>
Subject's PK alg id	<code>userCertificate.subjectPublicKey.algorithm</code>
Issuer unique id	<code>userCertificate.issuerUID</code>
Subject unique id	<code>userCertificate.subjectUID</code>

How You Can Help - User Requirements

Which fields should we extract and index on?

PK Certificate

- has 11 base field

- 16 standard X.509 extensions

- 2 PKIX extensions

- an infinite number of private extensions (from Netscape, MS, Entrust, Baltimore etc.)

AC Certificate has 12 X.509 AC extensions

CRL has

- 7 base fields

- 13 standard X.509 extensions

Implementation

For each new extension

- Server will need to know the ASN.1 data type

- Server will need to build a new index

- Client will need to be able to enter the matching information

Implies a configuration option in OpenLDAP

We are proposing to build

- Indexes for every basic field and

- A handful of X.509 and PKIX standard extensions

- The OID and criticality flag of each extension

Which extensions will be the most useful to YOU?

LDAP tools

slapadd, slapcat, slappasswd

ldapsearch

ldapmodify

ldapadd

LdapAdmin.exe

!!! The *slap utilities operate directly on the database files, without connecting to the LDAP server. Because OpenLDAP caches entries it is *dangerous* to use them *when slapd is running* !!!**

DIT Initialisation

```
# cat base.ldif
dn: dc:linuxtraining,dc=be
objectClass: dcObject
objectClass: organizationalUnit
dc: linuxtraining
ou: linuxtraining
#
# /usr/local/bin/slapadd -v -l base.ldif
```

Start slapd (see Running slapd) and verify the tree contents:

```
# /usr/local/bin/ldapsearch -x -b "dc=linuxtraining,dc=be" "(objectclass=*)"
```

OpenLDAP tools common options

Options common to all **ldap*** utilities:

d integer: similar to loglevel in slapd.conf

D “binddn”: DN to bind to when authenticating

f LDIF-file: specifies the file containing LDIF entries to be used during the operation

l: enable SASL interactive mode (prompts for password)

n: do not perform the search, only show what would be done

P [1|2]: which LDAP version to use, 2 or 3

R SASLrealm: SASLrealm to use when authenticating

U username: username to use for SASL authentication

v: verbose mode

w password: password for authentication (**dangerous**)

W: prompt for password

x: use simple authentication

y passwordFile: read password from this file

OpenLDAP adding data

Example:

```
ldapadd -xWD "cn=Manager,dc=linuxtraining,dc=be" -f datafile.ldif
```

Explanation:

x: use simple bind (no SSL or SASL)

W: prompt for password

D: use "cn=Manager,dc=linuxtraining,dc=be" as identity to logon

f: use the specified file as LDIF data to add to the tree

OpenLDAP

ldapsearch usage

`ldapsearch` specific options:

- a How to handle aliases during the search (never, always, search, find)
- A return only attribute names, not values
- b Base DN to search
- l Limit search to the specified time (seconds)
- s Define the scape (syb, base, one)
- S Sort the search data by the values of the specified attributes
- z Specify the maximum number of entries to return

example:

```
ldapsearch -x -b "dc=softwareschule,dc=ch" "(objectclass=*)"
```

OpenLDAP ldapmodify

`ldapmodify` can be used to change the ldap data in the tree. It understand 2 special attributes:

changetype

This attribute defines how the entry should be modified. It can be one of the following:

add: add the entry to the directory

delete: delete the entry from the directory

modify: change or delete attributes

modrdn: modify the RDN of an entry

moddn: modify the DN of an entry

delete

the value(s) of the attribute are names of attributes to be deleted.

LDAP other tools

A range of tools are available for ldap management:

GQ: GTK+ ldap client (<http://biot.com/gq/index2.html>)

Softerra LDAP browser: Windows LDAP client
(<http://www.ldapbrowser.com/>)

phpLdapAdmin: Powerfull web based ldap administration frontend
(<http://phpldapadmin.sourceforge.net/>)

AKBK website: LDAP Schema Viewer
(<http://ldap.akbkhhome.com/index.php>)

LDAPAdmin:

<http://ldapadmin.sourceforge.net/>

LDAP and SSL/TLS (1/2)

For more data security it is possible to setup SSL tunnels.

SSL uses X.509 certificates for creating a secure and authenticated connection.

Certificates can be bought from an established Certificate Authority (VeriSign, Thawte, Comodo, ...)

Self Signed Certificates can be created with the **openssl** tools:

```
/usr/lib/ssl/misc/CA.pl -newcert  
openssl rsa -in newreq.pem -out newkey.pem  
mkdir -p /usr/var/openssl  
cp newkey.pem /usr/var/openssl/slapd-key.pem  
cp newreq.pem /usr/var/openssl/slapd-cert.pem
```

LDAP and SSL/TLS

(2/2)

Configuration of slapd SSL support:

in the global section of /etc/ldap/slapd.conf

TLS options for slapd

TLSCipherSuite HIGH

TLSCertificateFile /usr/var/openldap/slapd-cert.pem

TLSCertificateKeyFile /usr/var/openldap/slapd-key.pem

Now start ldap as:

```
/usr/sbin/slapd -h "ldap:/// ldaps://"
```

LDAP and Delphi

Uses Files

uses Winldap;

The initial developer of the Pascal code is Luk Vermeulen <http://www.delphi-jedi.org/>

Libraries

- winldap.h LDAP client 32 API header file

```
LDAPLib = 'wldap32.dll';  
function ldap_openA; external LDAPLib name  
'ldap_openA';
```

3 Forms

ldap_bindW, ldap_bindA, and ldap_bind

LDAP and Delphi

Uses Files

uses WinLDAP;

Only the UNICODE version of the LM APIs are available on NT.

```
//-----  
----- // Delphi-Portierung von Teilen der  
WinLDAP.H (unvollständig) // Copyright dieser  
Portierung (c) 2008 Jens Geyer und Toolbox-Verlag.  
//-----  
----- unit WinLDAP; {$IFDEF VER130} {$A+}  
{$ELSE} {$A4} {$ENDIF} {$Z4}
```

LDAP Code

```
// open connection

pld := ldap_open(PChar(sServer), iPort);

if Assigned(pld) then
try
    // authenticate anonymously
    LDAPCheck(ldap_simple_bind_s(pld, nil, nil));

    // perform search
    LDAPCheck(ldap_search_s(pld, PChar(sBase), LDAP_SCOPE_SUBTREE,
        PChar(sSearch), nil, 0, plmSearch));

try
    // initialize results
    iRow := 0;
    msResults.Clear;
    slAttribs.Clear;

    // loop thru entries
    plmEntry := ldap_first_entry(pld, plmSearch);
    while Assigned(plmEntry) do begin
```

LDAP and Unicode

```
{ $EXTERNALSYM ldap_openA }
```

```
function ldap_openA(HostName: PAnsiChar; PortNumber:  
    ULONG): PLDAP; cdecl;
```

```
{ $EXTERNALSYM ldap_openW }
```

```
function ldap_openW(HostName: PWideChar; PortNumber:  
    ULONG): PLDAP; cdecl;
```

```
{ $EXTERNALSYM ldap_open }
```

```
function ldap_open(HostName: PChar; PortNumber: ULONG):  
    PLDAP; cdecl;
```

Support ASN.1 and LDAP Data Interchange Format (LDIF). It is hard to say Delphi 2008 have much improved connectivity without supporting these data interchange formats.

LDAP & Unicode in Delphi 2009 Discussion

Now that PChar is an alias to PWideChar, things started falling over because the element type is now 2 bytes.

With the preprocessor symbols `_UNICODE` and `UNICODE`, any Windows C/C++ preprocessor that can be used with the Windows SDK will happily replace every occurrence of `TCHAR` with `WCHAR`; ... which would be the counterpart to `WideChar` in Delphi and is also defined in `Windows.pas`, if memory serves me well.

If you have touched the LSA headers you'll have come accross `LSA_UNICODE_STRING` and `LSA_STRING`, where the latter one uses `PCHAR` clearly in the ANSI sense. LDAP functions also use it excessively.

Distributed Directories, Child

Distribution is done by an entry with the *referralClass* as an objectclass (*Subordinate knowledge references*)

```
dn: ou=sales,dc=linuxtraining,dc=be
```

```
ou: sales
```

```
objectClass: extensibleObject
```

```
objectClass: referral
```

```
ref: ldap://ldap2.linuxtraining.be/ou=sales,dc=linuxtraining,dc=be
```

LDAP Replication (1/3)

OpenLDAP replication is handled by **slurpd**

slapd stores all ldap changes in a file identified by the *repllogfile* directive

slurpd monitors this file and pushes all changes to the replica server

LDAP Replication (2/3)

Master server setup:

```
repllogfile      /var/ldap/slapd.repllog  
replica          host=replica1.linuxtraining.be:389  
                 suffix="dc=linuxtraining,dc=be"  
                 binddn="cn=replica,dc=linuxtraining,dc=be"  
                 credentials=password  
                 bindmethod=simple  
                 tls=yes
```

LDAP Replication (3/3)

Slave server setup:

```
updatedn      "cn=replica,dc=plainjoe,dc=org"  
udateref     ldap://ldap.linuxtraining.be
```

(The rest of the config is identical to the master configuration)

Start the replication daemon:

```
/usr/sbin/slurpd
```

LDAP Schema

The schema holds a central importance, which is hidden from users

The schema determines the type of data a directory holds

Modifying the schema can extend the functionality of a directory

The schema components are highly interdependent

Schema is defined in RFC2252

LDAP Schema, Object Classes

An object class has many properties:

OID: Unique object identifier for the class

Name: Name used to refer to the class

Description: Brief description of what the class represents

Superior Class: Which object class the class is based on (*SUP*)

Class Category: one of *abstract*, *auxilliary*, *structural*

Mandatory Attributes: attributes that must have a value (*MUST*)

Optional Attributes: attributes that may have a value (*MAY*)

LDAP Schema, Object Classes

An example:

```
subschema OBJECT-CLASS ::= {  
    SUBCLASS OF { top }  
    KIND auxiliary  
    MAY CONTAIN { ditStructureRules | nameForms | ditContentRules |  
        objectClasses | attributeTypes | matchingRules | matchingRuleUse }  
    ID 2.5.20.1 }
```

LDAP Schema, Object Classes

Another example:

```
objectclass (2.16.840.1.113730.3.2.2
```

```
    NAME 'inetOrgPerson'
```

```
    DESC 'RFC2798: Internet Organizational Person'
```

```
    SUP organizationalPerson
```

```
    STRUCTURAL
```

```
    MAY (
```

```
        audio $ businessCategory $ carLicense $ depretementNumber $
```

```
        displayName $ employeeNumber $ employeeType $ givenName $
```

```
        homePhone $ homePostalAddress $ initials $ jpegPhotot $
```

```
        labeledURI $ mail $ manager $ mobile $ o $ pager $
```

```
            x500uniqueIdentifier $ preferredLanguage $
```

```
        userSMIMECertificate $ userPKCS12 )
```

```
)
```

LDAP Schema, Attribute Types

Define the syntax and properties of attributes:

OID: Unique object identifier of the attribute

Name: Name used to refer to the attribute

Description: Brief description of what the attribute

Superior Class: Which attribute types the attribute is based on (*SUP*)

Equality matching rule: How to match an asserted value

Order matching rule: How to determine the order related to an asserted value

Substring matching rule: How to match the attribute to an asserted string (with wildcards)

Syntax: The kind and form of the data allowed in the attribute value.

Number of allowed values, if *SINGLE-VALUE* is specified one value is allowed

Syntax rules and matching rules are defined in RFC2252, uses ASN.1 (Abstract Syntax Notation)

LDAP Schema, Attribute Types

attributetype (2.16.840.1.113730.3.1.1

NAME 'carLicense'

DESC 'RFC2798: vehicle license or registration plate'

EQUALITY caseIgnoreMatch

SUBSTR caseIgnoreSubstringsMatch

SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

attributetype (0.9.2342.192.00300.100.1.60

NAME 'jpegPhoto'

DESC ' RFC2798: a JPEG image'

SYNTAX 1.3.6.1.4.1.1466.115.121.1.28)

References

O'Reilly's "LDAP Directories Explained: An Introduction and Analysis"

O'Reilly's "LDAP System Administration"

<http://www.openldap.org>

LDAP - Eine Einführung

<http://www.mitlinx.de/ldap/>

Sourcen:

www.softwareschule.ch/download/openldap_delphi.zip