Das Lightweight Directory Access Protocol

Autor: Thomas Bendler (*project@bendler-net.de*)
Autor: Steffen Dettmer (*steffen@dett.de*)
Layout: Torsten Hemm (*T.Hemm@gmx.de*)
Layout: Steffen Dettmer (*steffen@dett.de*)

Dieses Dokument beschreibt die Basis Installation und Konfiguration des "OpenLDAP" Paketes, welches einen objektorientierten Verzeichnisdienst bereitstellt.

Inhaltsverzeichnis

1 Prolog

Die von mir beschriebene Installation bezieht sich auf die aktuelle Debian-Distribution Version 2.2 (Potato). Um eine möglichst distributionsunabhängige Installation zu beschreiben, beziehe ich mich auf die Installation des Tarballs unter `/usr/local`. Sollte ein distributionsabhängiges Binär-Paket verwendet werden, sind ggf. die Pfade anzupassen.

1.1 Lizenz

1.2 Bezugsquellen

Die aktuelle Version des "OpenLDAP"-Paketes findet sich unter ftp://www.openldap.org/pub/OpenLDAP/ Distributionsabhängige Pakete finden sich in der Regel auf den beigelegten CDs respektive auf den FTP-Servern des jeweiligen Distributors.

1.3 Literatur

Da ich weder Lust noch Zeit habe, das "OpenLDAP"-Paket bis ins letzte Detail zu beschreiben, verweise ich bei weiterführenden Fragen auf die gelistete Literatur.

1.3.1 Bücher

* Jens Banning, "LDAP unter Linux - Netzwerkinformationen in Verzeichnisdiensten verwalten", Addison-Wesley, geb., 250Seiten, DEM 79,90, ISBN 3-8273-1813-0
* "Implementing LDAP" by Mark Wilcox
* "Programming Directory-Enabled Applications with Lightweight Directory Access Protocol" by Howes and Smith
* "Understanding and Deploying LDAP Directory Servers" by Howes, Smith and Good

1.3.2 RFC's

* 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

1.3.3 Adressen im Internet

* OpenLDAP HomePage
  http://www.openldap.org

* Einführung im Linux Magazin
  http://www.linux-magazin.de/ausgabe/1998/09/LDAP/ldap.html
* slapd und slurpd Administrator's Guide
  http://www.umich.edu/~dirsvcs/ldap/doc/guides
* Introducing to Directory Service (X.500)
  http://www.nic.surfnet.nl/surfnet/projects/x500/introducing
* Linux Directory Service unter
  http://www.rage.net/ldap/

1.4 Haftung

Für die hier vorgestellten Verfahren übernehme ich keine Haftung. Sollten sich Fehler eingeschlichen haben oder Verfahren nicht funktionieren, bitte ich um Feedback (Adresse siehe unten).

1.5 Feedback

Bei Fragen und Kommentaren zu diesem Dokument sowie bei Anregungen und Verbesserungsvorschlägen wenden Sie sich bitte an den Maintainer *Steffen Dettmer*

2 Eine kleine Einführung in LDAP

2.1 Was ist LDAP?

LDAP ist die Abkürzung von "**L**ightweight **D**irectory **A**ccess **P**rotocol". Das LDAP entstand ursprünglich als Front-End für den X.500-Verzeichnisdienst. Da X.500 als kompletter OSI-Stack implementiert ist, war es nicht möglich, diesen Verzeichnisdienst flächendeckend zu implementieren. LDAP ist ein Verzeichnisdienst, der auf dem TCP/IP-Protokoll basiert und somit ressourcenschonender für die Netzwerk-Infrastruktur ist. Obwohl LDAP nur einen Teil der Funktionen des DAP zur Verfügung stellt, reicht es aus, um die fehlenden Funktionen vollständig zu emulieren. Basis für LDAP sind die im Abschnitt **Literatur** aufgeführten RFC's.

2.2 Welche Informationen kann LDAP zur Verfügung stellen?

LDAP speichert seine Informationen in einer Baumhierarchie. Diese Hierarchie kann diverse Informationen enthalten. Einen Überblick verschafft RFC 2307, in dem mögliche Inhalte der LDAP Hierarchie spezifiziert sind:

* Benutzer
* Gruppen
* IP-Dienste
* IP-Protokolle
* RPC's
* NIS-Netzwerkgruppen
* Boot-Informationen
* Einhängepunkte für Dateisysteme
* IP-Hosts und Netzwerke
* RFC 822 konforme Mail-Aliase

2.3 Strukturierung der Informationen

Hat man Daten in Datenbanken, so ist es wichtig, diese Informationen zu strukturieren. Besonders, wenn viele verschiedene Clients auf die Datenbank zugreifen wollen, zum Beispiel Netscape, Outlook und andere, muss die Struktur genau definiert sein. Wenn ein Mailclient eine eMail-Adresse braucht, muss er wissen, wie er diese bekommt.

Es gibt viele Definitionen, an die man sich halten muss, soll am Ende auch etwas funktionieren. Bei LDAP werden Objekte mit Eigenschaften verwendet. Jedes Objekt hat zunächst einen eindeutigen Namen, an dem es von allen anderen unterschieden werden kann ("distinguished name", kurz: DN). Die Eigenschaften eines Objekts hängen davon ab, zu welcher Klasse es gehört (es kann sogar zu mehreren Klassen gehören).

2.3.1 Klassen von Objekten

Es sind nun Klassen für Personen definiert. Zu einer Person ("person") gehören zwingend `objectClass` (die Objektklasse selbst), `sn` (der Nachname) und `cn` (commonName, etwa: üblicher Name, hier wird üblicherweise Vor- und Nachname verwendet). Zusätzlich gibt es optionale Attribute, die nicht unbedingt angegeben werden müssen. Diese sind hier `description` (beliebige Beschreibung), `seeAlso` (verweist auf ein anderes Objekt), `telephoneNumber` (Telefonnummer), `userPassword` (ein Password). Da häufig noch mehr Attribute mit einer Person verknüpft sind, gibt es auch eine Objektklasse organizationalPerson. Diese hat die gleichen geforderten Eigenschaften wie Person, aber erlaubt viele optionale Eigenschaften, wie zum Beispiel Felder der Adresse und eine FAX-Nummer. Es gibt noch mehr Klassen, zum Beispiel `newPilotPerson` (die als optionale Eigenschaft eine eMail-Adresse `mail` einführt) und natürlich Klassen für Organisationen/Firmen, Abteilungen, Bilder, Dokumente, Geräte und so weiter.

2.3.2 Eigenschaften von Klassen

Wenn man also irgendwo eine Person im Verzeichnis hat, ist klar, dass diese ein `cn` haben muss, und eine Telefonnummer haben kann (und dass diese genau `telephoneNumber` heißt). Soll ein Programm eine eMail-Addresse suchen, muss es nur nachschauen, ob es ein Attribut `mail` gibt. Es kann eben nicht sein, dass diese Eigenschaft `email` oder anders heißt. `mail` ist vorgeschrieben, und nichts anderes.

An diesem Beispiel kann man zeigen, dass eine natürliche Person zu mehreren Klassen gehört: `person`, `organizationalPerson` und `newPilotPerson`. Eine weitere Klasse ist `top`. Im Prinzip gehört so ziemlich jedes Objekt auch zur Klasse `top`, die lediglich vorschreibt, dass die Eigenschaft `objectClass` gesetzt sein muss (was bei allen Personenklassen ohnehin gefordert ist).

Durch das Verwenden der Klassen definiert man, welche Eigenschaften vorhanden sein müssen, und welche vorhanden sein können. Eine Person darf zum Beispiel keine Farbtiefe haben, ein Bild hingegen schon. Verwendet man mehrere Klassen, so muss das entsprechende Objekt alle von mindestens einer Klasse geforderten Eigenschaften haben, und kann alle insgesamt erlaubten Eigenschaften haben.

2.3.3 Typen von Eigenschaften

Den Eigenschaften sind Typen zugeordnet. Es gibt Typen, die eine Zeichenkette enthalten, und andere, die eine Telefonnummer enthalten. Diese Typen definieren weiterhin, wie Werte verglichen (und damit sortiert und gesucht) werden. Zeichenketten beispielsweise kann man abhängig von Groß- und Kleinschreibung vergleichen oder auch nicht. Bei Telefonnummer spielen gewisse Füllzeichen möglicherweise keine Rolle. Passwörter hingegen müssen genau übereinstimmen.

Es gibt nun also Objekte (die zu bestimmten Klassen gehören). Diese werden nun anderen Objekten untergeordnet (beziehungsweise werden anderen Objekte viele zugeordnet, dies ist die richtige Reihenfolge). Diese baumartige Struktur kann man (mit etwas Phantasie) auch in der Realität finden: In Ländern gibt es Firmen, in Firmen gibt es Abteilungen und in Abteilungen letztlich Personen. So wird das in LDAP auch gesehen. Die Baumstruktur wird hier "Directory Information Tree" genannt, kurz **DIT**. Es gibt Länder (also Objekte der Klasse `country` [Land]) mit u.A. der Eigenschaft `c` (kurz für country), Firmen (`organisation`s mit der Eigenschaft `o`), Abteilungen (Organisations Einheit, `organisationalUnit` mit der Eigenschaft `ou`). Hierbei enthalten diese Objekte normalerweise viele weitere Eigenschaften; eine Firma hat zum Beispiel eine Postanschrift.

2.3.4 Schema

Ein Schema ist eine Sammlung von Strukturdefinitionen. Dazu gehören die Schreibungen vieler Klassen und der verwendeten Typen. Es gibt verschiedene Schemata, und es ist möglich, eigene zu definieren (oder bestehende zu erweitern) was jedoch nicht interoperabel mit anderen Diensten sein muss. Das ist außerhalb der Betrachtungen dieses Dokumentes.

2.3.5 Zusammenfassung

Der LDAP-Server speichert seine Informationen in einer baumartigen Struktur. Diese wird auch "Directory Information Tree" genannt, kurz **DIT**.

Zum Speichern benutzt der LDAP-Server Objekte, die er mit Attributen versehen kann. Dadurch kann man die Struktur flexibel an die eigenen Bedürfnisse anpassen. Das RFC 2256 spezifiziert die Standard-Objekte des LDAP-Servers. Man wird zwar von niemandem gezwungen, diese Vorgaben auch zu benutzen. Um aber eine möglichst große Konformität zu erzielen, sollte man diese Vorgaben einhalten.

3 Technische Daten des LDAP Server

Der Zugriff auf den LDAP-Server erfolgt über das LDAP-Protokoll via TCP/IP. Per Default lauscht der `slapd` ("Stand-alone LDAP Daemon": Der LDAP-Dienst) auf dem Port 389. Dies ist im RFC 1777 spezifiziert.

4 Installation des OpenLDAP

Beschrieben wird im folgenden die Installation des OpenLDAP in der Version 1.2.1. Die Installation zukünftiger Releases sollte nicht grundlegend von der hier vorgestellten Methode abweichen. Sollte dies trotzdem der Fall sein, werde ich das in zukünftigen Versionen dieses Dokumentes berücksichtigen.

4.1 Quellen für den OpenLDAP-Server

Der Quellcode der aktuellen Version des OpenLDAP-Servers in einem komprimierten Archiv findet sich auf der Homepage der OpenLDAP Foundation. Die aktuellen Quellen können von ftp://ftp.OpenLDAP.org/pub/OpenLDAP/openldap-release.tgz bezogen werden.
Eine einfachere Möglichkeit der Installation bieten sogenannte rpm/deb-Archive. Dies sind bereits kompilierte Pakete, die auf die Besonderheiten der jeweils eingesetzten Distribution zugeschnitten sind. Die jeweilige Installationsprozedur entnehmen Sie bitte Ihrem Handbuch.

4.2 Installation des OpenLDAP-Servers

Haben Sie den OpenLDAP mit Hilfe der distributionseigenen rpm/deb-Archive installiert, können Sie diesen Abschnitt auslassen.

Wenn Sie sich die Quellen des OpenLDAP-Servers gezogen haben, müssen Sie diese noch installieren. Zu diesem Zweck kopieren sie die Quellen nach "/usr/local/src/" und entpacken sie die Quellen mit dem Befehl `tar xvfz ./openldap-release.tgz`

Anschließend müssen Sie mit `cd ldap` in das Installationsverzeichnis wechseln. Dort befindet sich die Datei `include/ldapconfig.h.edit`. In ihr kann man den LDAP an die eigenen Bedürfnisse anpassen. In der Regel sollten aber die voreingestellten Werte in Ordnung sein. Das "OpenLDAP" Paket wird per Default nach `/usr/local/` installiert.
Nun geht es ans Übersezten und Installieren des Programmpaketes. Führen Sie dazu folgende Befehle aus:

```
user@linux / # configure
user@linux / # make depend
user@linux / # make
```

Um die Kompilation zu testen, können noch folgende Anweisungen ausgeführt werden:

```
user@linux / # cd test
user@linux / # make
```

Die Installation des Paketes muss als Superuser (root) mit folgendem Befehl erfolgen:

```
user@linux / # su
root@linux / # make install
```

That's it.
Nun sollte der OpenLDAP-Server installiert sein.

4.3 Besonderheiten von RPM-Packeten

Es gibt Unterschiede zwischen dem Orginal-OpenLDAP-Paket und den rpm-Archiven (hier am Beispiel von SuSE).

Die beiden Pakete sind zwar nach der Installation inhaltlich fast identisch, unterscheiden sich aber gravierend in den verwendeten Pfaden. Folgende Übersicht soll die Unterschiede verdeutlichen:

OpenLDAP-Original in der Default-Konfiguration:

* `/usr/local/etc/openldap/` Konfigurationsdateien
* `/usr/local/bin/` Hilfsdateien
* `/usr/local/sbin/` Server
* `/usr/local/src/ldap/doc/` Dokumentation
* `/usr/local/include/` Include-Dateien
* `/usr/local/lib/` Bibliotheken
* `/usr/local/share/` Dateien für X.500 Gateway
* `/usr/local/var/openldap-ldbm/` Datenbankdateien

SuSE rpm-Archive Konfiguration:

* `/etc/openldap/` Konfigurationsdateien
* `/usr/bin/` Hilfsdateien
* `/usr/libexec/openldap/` Server
* `/sbin/init.d/ldap` Startskript
* `/usr/doc/packages/openldap/` Dokumentation, zusätzliche Tools
* `/usr/include/` Include-Dateien
* `/usr/lib/` Bibliotheken
* `/usr/share/openldap/` Dateien für X.500 Gateway

5 Anpassen der Konfigurationsdateien

Mit dem OpenLDAP-Server werden mehrere Konfigurationsdateien ausgeliefert, die teilweise noch an die lokalen Gegebenheiten angepasst werden müssen.

5.1 Liste der Konfigurationsdateien

* `ldap.conf` -- Client Konfiguration
* `ldapfilter.conf` -- Filterregeln
* `ldapsearchprefs.conf` -- Bevorzugte Suchkriterien
* `ldaptemplates.conf` -- Templates für Formulare
* `slapd.conf` -- Server Konfiguration
* `slapd.at.conf` -- Beschreibung der Attribute
* `slapd.oc.conf` -- Beschreibung der Objektklassen
* `*.schema` -- neue Beschreibung der Attribute und Objektklassen

Zusätzlich zu den dem Paket beiliegenden Konfigurationsdateien gibt es nochmal denselben Satz mit der Endung "*.default". Diese kann man getrost löschen oder sich in ein extra Verzeichnis kopieren um evtl. nochmal die möglichen Einstellungen überprüfen zu können.

5.2 Konfigurieren der ldap.conf

In der Datei ldap.conf wird die Basis-Domain für den LDAP-Client festgelegt. Für das folgende Beispiel im Abschnitt "Erstellen eines Beispielverzeichnisses" wird die Basisadresse mit der Domain gleichgesetzt. Das weicht von dem vielleicht intuitiveren "Land-Organisation-Organisationseinheit"-Schema ab, schafft dafür aber "private" Namensräume. Dies löst folgendes Problem: Wenn Firma A Firma B in Ihrem LDAP listet, kann es Herrn Meier aus B im LDAP von Firma A und dem von Firma B geben. Beide Meiers sind der gleiche, haben den gleichen eindeutigen Namen (DN); jedoch sind es verschiedene Objekte (in Firma B kann Herr Meier ein Passwort haben, das Firma A nicht kennt!).

Deshalb hat man sich überlegt, dass Firmen einfach ihren Internet-Domainnamen verwenden können, um ihre Namen zu bilden. Diese werden `dc` (Domain Component, Domainkomponente) genannt. `selflinux.de` besteht zum Beispiel aus den Komponenten `selflinux` und `de`, also `dc=selflinux,dc=de` (Ansonsten würde man `c=DE,o=Bendler Projekte` oder sowas verwenden, was jedoch eigentlich international vergeben werden müßte).

/usr/local/etc/openldap/ldap.conf

```
# /usr/local/etc/openldap/ldap.conf
#
# Thomas Bendler, 19.06.2000, 17:05:06
#
# Bitte beachten Sie auch ldap.conf(5)
# Diese Datei sollte fuer alle lesbar sein
#
BASE dc=selflinux,dc=de
HOST voyager.bendler.net
```

Was bewirkt die hier vorgestellte `ldap.conf`? Mit der Variable BASE wird der standardmäßig abgefragte Teilbaum festgelegt. Hier bedeutet das, alle Anfragen sind unterhalb von `dc=selflinux,dc=de` durchzuführen. Dies wird häufig "Suchbasis" (Searchbase) genannt. Die Variable `HOST` gibt den Server an, der standardmäßig abgefragt wird. Über die Variable PORT kann alternativ auch ein anderer Default Port eingestellt werden.

5.3 Konfiguration der slapd.conf

Die Datei `slapd.conf` enthält die Einträge für die Konfiguration des slapd Standalone-Server. Der slapd beantwortet die LDAP Anfragen der Clients - es ist der LDAP-Server oder Verzeichnis-Server. Für das folgende Beispiel bekommt die Datei folgenden Inhalt:

/usr/local/etc/openldap/slapd.conf

```
# /usr/local/etc/openldap/slapd.conf
#
# Thomas Bendler, 27.06.2000, 15:37:02
# überarbeitet von Steffen Dettmer, 2001
#
# Bitte beachten Sie auch sldapd.conf(5)
#
# Modifizierte Version der slapd.conf aus
# dem Debian OpenLDAP Packetes
# http://www.debian.org/
#

#
# -- Einzubindende Dateien --
#
# Schema und ObjectClass Definitionen
include/usr/local/etc/openldap/slapd.at.conf
include/usr/local/etc/openldap/slapd.oc.conf

# Schema und ObjectClass Definitionen fuer Netscape Roaming
include/usr/local/etc/openldap/netscape_roaming.at.conf
include/usr/local/etc/openldap/netscape_roaming.oc.conf

# Schema for supporting Debian Package Directory entries
#include/usr/local/etc/openldap/debian.at.conf
#include/usr/local/etc/openldap/debian.oc.conf

# Neuere Versionen verwenden nicht mehr Attribut und objectclass
# Dateien, sondern stattdessen Schemata:
# include /usr/local/etc/openldap/schema/core.schema
# include /usr/local/etc/openldap/schema/cosine.schema
# include /usr/local/etc/openldap/schema/inetorgperson.schema

#
# -- Servereinstellungen --
#
# Wenn Schemacheck auf ";on" gesetzt wird, wird bei der Modifizierung
# mit Hilfe von "ldapadd" ueberprueft, ob die Eintraege in Objektklassen
# spezifizert sind.
# In Produktion sollte dies auf "on" gesetzt werden, um zu
# vermeiden, dass "falsche" Eigenschaften (zum Beispiel
# Nachname eines Landes) gesetzt werden können.
schemacheck off

# Wenn lokal nichts gefunden wird frage folgenden Rechner
# Dieser gilt nur bei eingetragener Domain
referral ldap://root.openldap.org

# Prozess ID und Log Level, siehe auch man slapd.conf
pidfile /var/run/slapd.pid
argsfile /var/run/slapd.args
loglevel 256

#
# -- Datenbankeinstellungen --
#
# Welche Datenbank wird benutzt und wo ist sie gespeichert

database ldbm
directory "/usr/local/var/openldap-ldbm"

# Basis Domain (unser "root")
suffix "dc=selflinux,dc=de"

# Aenderungen werden mit Datum versehen
# Achtung, erfordert bestimmte Klassen!
lastmod on

#
# -- Indizierung --
#
# Art der Indizierung in der Datenbank
```

```
index cn pres,eq,approx,sub
index objectclass pres,eq
index default none

#
# -- Zugriffskontrolllisten --
#
#Man kann hier einen Manager global definieren. Das ist zum
#  Beispiel zum Anlegen der initialen Daten sinnvoll
#rootdn          "cn=Manager,dc=selflinux,dc=DE"
#rootpw         geheim>

# Standardrechte gibts nicht
defaultaccess none

# Die Manager (Administratoren) bekommen Zugriff auf das gesamte
# Verzeichnis:

access to *
by group/organizationalRole/roleOccupant="cn=Manager,
  dc=selflinux, dc=de" write
by group/organizationalRole/roleOccupant="cn=Manager,
  dc=selflinux, dc=de" read
by group/organizationalRole/roleOccupant="cn=Manager,
  dc=selflinux, dc=de" search

# Das eigene userPassword kann von einem Benutzer geaendert
# werden. Die anderen Nutzer koennen es vergleichen (d.h. prüfen)
access to attr=userpassword
by self write
by group="cn=Manager,dc=selflinux,dc=de" write
by * compare

# Jeder eingetragene Benutzer darf lesen, der Rest
# darf nichts
access to *
by self write
by dn=".+" read
by * none
```

Was bewirkt die hier vorgestellte slapd.conf?

Die include-Anweisungen bewirken ein Einbinden der angegebenen Dateien. In diesem Fall werden Objektklassen (oc) und deren Attribute (at) beziehungsweise die SChemata-Dateien eingelesen. Die "Netscape Roaming"-Dateien gehören nicht zum Standard-Umfang des LDAP-Tarballs. Sie können auf meiner Homepage unter http://www.bendler-net.de/projects/config/ldap-config.tar.gz runtergeladen werden.

Mit dem schemacheck wird überprüft, ob modifizierte oder neu installierte Daten den Regeln der Objektklassen entsprechen. In produktiven Systemem sollte das immer auf "on" stehen. Dabei werden leider nur Objektklassen geprüft, die LDAP in einer Konfigurationsdatei wie zum Beispiel slapd.oc.conf bekannt sind. Bei allen anderen Klassen akzeptiert der Server leider alle Attribute.

Ist der LDAP-Server nicht in der Lage, eine Anfrage zu beantworten, fragt er den unter `referral` angegebenden LDAP-Server.

Die Zeilen `pidfile` und `argsfile` sind für den laufenden Betrieb (pid=prozess id, args=argumente).

Mit dem Schlüsselwort `database` wird festgelegt, welches Datenbankformat bzw. welche Datenbank benutzt wird. Es sind auch Abfragen von anderen Datenbanken möglich. Im `directory` wird spezifiziert, wo die Datenbankdateien zu finden sind bzw. angelegt werden sollen. Unter Umständen muss dieses Verzeichnis noch nachträglich angelegt werden wenn z.B. ein kompiliertes Paket installiert wird, in dem das Verzeichnis nicht angelegt wird (war bei SuSE 6.2 der Fall, das hat sich mittlerweile jedoch geändert).

Die unter `suffix` angegebene Struktur legt fest, welche Anfragen über die lokale Datenbank beantwortet werden

können. Der Suffix legt sozusagen den Namensraum des Verzeichnisses fest - hiermit wird also die "Wurzel" / "Root" festgelegt.

Mit Hilfe der `index`-Anweisung wird der Datenbank mitgeteilt, wie sie ihre Indizes anlegen soll.

Zum Schluß werden noch die Zugriffsrechte auf dem LDAP-Server festgelegt. Standardmäßig erhält jeder Benutzer Lesezugriff. Die einzelnen Zugriffskontrolllisten (ACL, "Access Control List") entnehmen Sie bitte der Konfigurationsdatei.

5.4 Attribute und Objektklassen

Wie man bereits in der Konfigurationsdatei slapd.conf sehen kann, werden mehrere Konfigurationsdateien eingebunden, unter anderem `slapd.at.conf` und `slapd.oc.conf` beziehungsweise die `*.schema-Dateien` bei neuren Versionen (mehr dazu im nächsten Abschnitt). Diese Dateien sollten nicht geändert werden. Möchte man eigene Definitionen einbinden sollte das über gesonderte Dateien geschehen, z.B. `slapd.local.at.conf` und `slapd.local.oc.conf`. Die Dateien enthalten die Standard-Objektklassen und die Attribute für die Objektklassen. Die standardmäßig gegebenen Dateien sind für die meisten Anwendungen (Ausnahmen bestätigen die Regel, siehe auch "Netscape Roaming") ausreichend. Für weitere Informationen konsultieren Sie bitte die entsprechenden Manual-Pages.
Sollen die Benutzer ihre Einträge selbst verändern können, so empfielt sich noch eine Anpassung der ldaptemplates.conf an die eigenen Bedürfnisse. Sie stellt die Voreinstellungen zur Verfügung, die Programme geliefert bekommen, die auf den LDAP-Server zugreifen.

5.5 Schemata

Neuere Versionen verwenden Schemata, um Objektklassen und deren Attribute zu definieren. Ein Schemata-Element wird über einen OID, einen Object Identifier, eindeutig bestimmt. Das ist im Westentlichen eine Ziffernfolge, die man in etwa mit der Kapitel-Gliederung eines sehr großen Buches vergleichen kann: Es gibt 1.1 und 1.2, unterhalb von 1.1 dann 1.1.1 und 1.1.2. Es kann dann auch 1.1.2.1.19.241.243.4 geben und so weiter.

Über OIDs kann man - vereinfacht gesprochen - allen möglichen Dingen eine eindeutige Nummer geben. Das sind nicht nur Objektklassen, sondern auch Datentypen und Syntaxregeln. Die Syntax eines DNs ist zum Beispiel im OID 1.3.6.1.4.1.1466.115.121.1.12 definiert, ein "directoryString" (das ist eine Zeichenkette im UTF-8 Zeichensatz) ist definiert als 1.3.6.1.4.1.1466.115.121.1.15.

In einem Schema definiert man sich zunächst Attributtypen. Dazu gibt es die Direktive attributeType. Ein Beispiel für die Verwendung:

/usr/local/etc/openldap/schema/core.schema (Auszug)

```
attributeType ( 2.5.4.41 NAME 'name'
        DESC 'name(s) associated with the object'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{32768} )

attributeType ( 2.5.4.3 NAME ( 'cn' 'commonName' )
        DESC 'common name(s) assciated with the object'
        SUP name )
```

So ist "name" und "cn" oder "commonName" definiert, die uns bereits begegnet sind. Sie haben den OID 2.5.4.41 bzw. 2.5.4.3.

Es gibt also "name". Der muß im Format "Zeichenkette im UTF-8 Zeichensatz" sein (SYNTAX) und bei Vergleichen ist die Groß-/Kleinschreibung egal (EQUALITY caseIgnoreMatch). So kann man sich eigene Attribute sehr flexibel

definieren.

Der OID muß eindeutig sein - es muß also zu jeder Nummer genau ein Attribut oder eine Objektklasse zugeornet sein. Alles, was mit 1.1 beginnt, kann man privat verwenden, das ist sozusagen reserviert, da es offiziell nicht verwendet wird.

Man kann zum Beispiel (für sich selbst) festlegen: 1.1.2 für LDAP OIDs (weil die Netzwerker 1.1.1 schon für ihre SNMP OIDs verwenden). Darunter dann 1.1.2.1 als Prefix für Attribute und 1.1.2.2 für Objektklassen.

Die erste neue, eigene Klasse bekäme damit dann 1.1.2.2.1. Definiert man mehrere Klassen, muß man natürlich verschiedene OIDs verwenden, der nächste könnte 1.1.2.2.2 sein.

Meistens möchte man eigene Objektklassen definieren, wenn man spezialle Anforderungen hat. So möchte man vielleicht keine "gewöhnliche" inetOrgPerson-Klasse verwenden, sondern zusätzliche Attribute erlauben. Eigene Definitionen schreibt man natürlich am Besten in eine eigene Datei, die man zusätzlich mit "include" einbindet (so muß man bei Software-Updates nur ein include hinzufügen).

Als Beispiel fügen wir optional (MAY) das Attribut "myPhoto" zu einer Ableitung (SUP) der Klasse inetOrgPerson hinzu:

/usr/local/etc/openldap/schema/local.schema

```
objectclass ( 1.1.2.2.1 NAME 'myPerson'
        DESC 'my person'
        SUP inetOrgPerson
        MUST ( myUniqueName $ givenName )
        MAY myPhoto )
```

Damit man nicht immer die langen Nummern schreiben muß, kann man sich Macros definieren:

/usr/local/etc/openldap/schema/local.schema

```
objectIdentifier myOID                  1.1
objectIdentifier mySNMP                 myOID:1
objectIdentifier myLDAP                 myOID:2
objectIdentifier myAttributeType        myLDAP:1
objectIdentifier myObjectClass          myLDAP:2

objectclass ( myObjectClass:1 NAME 'myPerson'
        DESC 'my person'
        SUP inetOrgPerson
        MUST ( myUniqueName $ givenName )
        MAY myPhoto )
```

Danach käme dann myObjectClass:2, was sicherlich viel klarer ist als 1.1.2.2.2. 1.1.1 taucht als "mySNMP" in dem Beispiel auch wieder auf, so sieht man gleich, warum 1.1.1 hier nicht verwendet wird - es "gehört" ja im Beispiel den Netzwerkern für SNMP.

6 Erstellen eines Beispielverzeichnisses

6.1 Erstellen der LDIF Dateien

Nach der Installation und Konfiguration des LDAP-Servers muss dieser mit Daten gefüttert werden. Das folgende Beispiel erklärt einen "Directory Information Tree" anhand einer Firma mit mehreren Abteilungen. Die einzelnen Felder müssen den lokalen Gegebenheiten nur angepaßt werden, um eine simple Konfiguration aufzusetzen.
Für das folgende Beispiel wird im Verzeichnis `/usr/local/etc/openldap` das Unterverzeichnis `ldif/` angelegt. In diesem Verzeichnis kann mit jedem x-beliebigen Editor, der ASCII unterstützt, eine Datei mit dem Namen struktur.ldif erstellt werden. Der Name und das Verzeichnis für die Beispiel-LDIF-Dateien sind beliebig. Es müssen für den Fall, dass andere Namen oder Pfade verwendet werden, diese nur an die lokalen Gegebenheiten angepaßt werden.

Kommen wir aber nun zu der LDIF-Datei. Wir definieren einen neuen "Namensraum" `selflinux.de`. In LDAP-"Sprache" heißt das `dc=selflinux,dc=de`. Alles, was definiert wird, spielt sich unter diesem Punkt ab, damit gibt es keine Konflikte mit anderen Verzeichnissen.

Das Verzeichnis besteht zunächst aus einer Firma (bzw. Organisation) SelfLinux. Neben dieser Firma wird noch ein separater Eintrag eingerichtet, in dem die Administratoren (Manager) referenziert werden. Verzeichnis-Manager gehören also zu keiner Organisation oder Abteilung. Logisch gesehen ist das ja so auch korrekt.

Die Firma Selflinux bekommt drei Abteilungen ("Unterorganisationen") spendiert: Development, Sales und Support. Die Mitarbeiter werden dann mit den Abteilungen referenziert. Daraus ergibt sich folgende Struktur:

Struktur der Firma Selflinux

```
selflinux.de:
|
+-- Manager
|
+-- Selflinux
|
|
+-- Development
|   |
|   +-- Mitarbeiter 1 (Thomas Bendler)
|   |
|   +-- Mitarbeiter 2 (Steffen Dettmer)
|
+-- Sales
|   |
|   +-- Mitarbeiter 4 (Sonja Essler)
|
+-- Support
|
+-- Mitarbeiter 3 (Thomas Lippert)
```

Hat man keinen `rootdn`-Eintrag in der `slapd.conf`, sollte unbedingt ein Manager-Account in dieser Struktur mit angegeben werden, da ansonsten kein Schreib-Zugriff auf das Verzeichnis möglich wäre.

6.2 Beispiel LDIF struktur.ldif

Beispiel LDIF: struktur.ldif

```
dn: cn=Manager, dc=selflinux, dc=de
cn: Manager
description: Directory Manager
description: Verzeichnis-Manager
objectClass: organizationalRole
objectclass: top
roleOccupant: cn=Thomas Bendler,ou=Development,dc=selflinux,dc=de
roleOccupant: cn=Steffen Dettmer,ou=Development,dc=selflinux,dc=de

dn: o=Selflinux,dc=selflinux,dc=de
objectclass: top
objectclass: domain
objectclass: organization
o: Selflinux
l: Hamburg
postalcode: 21033
streetadress: Billwiese 22

dn: ou=Development,o=Selflinux,dc=selflinux,dc=de
objectclass: top
objectclass: organizationalunit
ou: Development

dn: ou=Sales,o=Selflinux,dc=selflinux,dc=de
objectclass: top
objectclass: organizationalunit
ou: Sales

dn: ou=Support,o=Selflinux,dc=selflinux,dc=de
objectclass: top
objectclass: organizationalunit
ou: Linux

dn: cn=Thomas Bendler,ou=Development,o=selflinux,dc=selflinux,dc=de
objectclass: top
objectclass: person
objectclass: organizationalperson
objectclass: inetorgperson
cn: Thomas Bendler
sn: Bendler
ou: Development
mail:project@selflinux.de
l: Hamburg
postalcode: 21033
streetadress: billwiese 22
telephonenumber: 040-7654321
facsmiletelephonenumber: 040-7654321
userpassword: {CRYPT}saHW9GdxihkGQ

dn: cn=Steffen Dettmer,ou=Development,o=Selflinux,dc=selflinux,dc=de
cn: Steffen Dettmer
sn: Dettmer
ou: Development
mail: steffen@dett.de
telephoneNumber: +49 (30) 1234567
objectClass: person

dn: cn=Sonja Essler,ou=Sales,o=Selflinux,dc=selflinux,dc=de
cn: Sonja Essler
sn: Essler
ou: Sales
mail: sonja@bendler-net.de
telephoneNumber: +49 (30) 1234568
objectClass: person

dn: cn=Thomas Lippert,ou=Support,o=Selflinux,dc=selflinux,dc=de
cn: Thomas Lippert
sn: Lippert
ou: Support
mail: tom@bendler-net.de
telephoneNumber: +49 (30) 1234569
objectClass: person
```

Das in diesem Beispiel eingesetzte Passwort `{CRYPT}saHW9GdxihkGQ` wurde mit Hilfe von Perl erzeugt. Dazu muss in der Shell folgender Befehl eingegeben werden. Das Ergebnis wird auf dem Bildschirm angezeigt und muss 1:1 in die LDIF Datei übertragen werden. Dieser Aufruf ist exemplarisch und nur zum Testen geeignet, da "salt" konstant und nicht zufällig ist! `perl -e 'print("{CRYPT}".crypt("secret","salt")."\n");'`

Damit wird das Passwort in ein für den Server verständliches Format gebracht. Möchten sie ein anderes Passwort verwenden, müssen sie stattdessen das gewünschte einsetzen. Sie können es nachträglich mit dem Befehl `ldappasswd` ändern.

Es ist in der Praxis einfacher, sofort das Passwort mit ldappasswd zu setzen (es beim Erstellen der Ursprungsdatei erstmal wegzulassen, wie im Beispiel bei den anderen Personen).

Falls man nicht über Rollen sondern Gruppen arbeiten möchte, kann man beispielsweise auch folgenden Manager-Eintrag benutzen:

Auszug LDIF struktur.ldif

```
dn: cn=Manager,dc=selflinux,dc=de
cn=Manager
objectclass: top
objectclass: groupofnames
member: cn=Thomas Bendler,ou=People,dc=selflinux,dc=de
```

Das Rollen-Modell ist jedoch üblicher und natürlicher, denn es ist hier klar, dass die Mitglieder bestimmte Rollen oder Aufgaben erfüllen. Der Begriff "Gruppe" hingegen suggeriert jedoch eher eine Personenaufteilung, technisch gesehen können natürlich Personen in mehreren Gruppen sein. Wenn die Gruppe jedoch eine bestimmte abstrakte Aufgabe hat, sollte man hier jedoch eher eine Rolle definieren. Dies wird oben im Beispiel so verwendet. Die Rolle "Manager" hat hier zwei "Mitglieder".

6.3 Umwandeln der LDIF-Datei in das LDBM-Format

Als nächstes muss die LDIF-Datei ins LDBM-Format konvertiert werden. Dazu dient der Befehl `ldif2ldbm`. Dieser ist unter "/usr/local/sbin/" zu finden.
Der Aufruf lautet:
`ldif2ldbm -i /usr/local/etc/openldap/ldif/struktur.ldif -f /usr/local/etc/openldap/slapd.conf`

Sollten sich irgendwelche Dateien nicht in den Standardpfaden befinden, so kann man so nach den Dateien suchen lassen:
`locate "Dateiname"`

oder, falls das nichts hilft, über das jedoch sehr langsame Kommando:
`find / -name "Dateiname"`

Ist die LDIF-Datei konvertiert, muss der LDAP-Server gestartet werden. Die meisten Distributionen stellen dafür ein Skript zur Verfügung welches sich unter `/etc/init.d/`, `/etc/rc.d/init.d/` oder unter `/sbin/init.d` befindet. Für gewöhnlich schimpft sich dieses `ldap` und kann mit diversen Parametern aufgerufen werden. Unter SuSE z.B. würde man mit folgendem init-Script den Server starten:
`/sbin/init.d/ldap start`

Ist kein Startskript vorhanden, wird der LDAP-Server mit folgendem Kommando gestartet:

```
/usr/local/libexec/slapd -f /usr/local/etc/openldap/slapd.conf
```

Damit der slapd beim Hochfahren automatisch gestartet wird, muss man ein Skript im `initd/` Verzeichnis anlegen. Das befindet sich je nach Distribution an einer anderen Stelle, so dass dem Leser nichts übrig bleibt als ein bisschen zu suchen. Normalerweise bieten die Distributionen ein Skeleton an, das an die jeweiligen Bedürfnisse angepasst werden muss. Ausführliche Informationen entnehmen Sie bitte der Dokumentation über das "Sys V System" Ihrer Distribution. Wahlweise hilft meistens auch ein `man init`, um etwas mehr über selbiges zu erfahren.

6.4 Testen des LDAP-Servers

Um den LDAP-Server zu testen, kann man jetzt eine Anfrage an selbigen schicken. Dies geschieht mit folgendem Befehl:
```
ldapsearch \
-D "cn=Thomas Bendler,ou=Develpoment,o=Support,dc=selflinux,dc=de" \
-W objectclass=\*
```

Der Server sollte nun eine Struktur, wie in der Datei beschrieben, als Antwort übergeben.

6.5 Hinzufügen von Datensätzen

Nun geht es an das Hinzufügen von Datensätzen. Dazu werden die bereits erstellten LDIF-Dateien benutzt. Das Hinzufügen geschieht mit Hilfe des Befehls `ldapadd` entsprechend der Syntax für die zwei erstellten LDIF-Dateien:
```
ldapadd -v \
-D "cn=Thomas Bendler,ou=Develpoment,o=Support,dc=selflinux,dc=de" \
-W \
-f /usr/local/etc/openldap/ldif/people.ldif
ldapadd -v \
-D "cn=Thomas Bendler,ou=Develpoment,o=Support,dc=selflinux,dc=de" \
-W \
-f /usr/local/etc/openldap/ldif/zuordnung.ldif
```

Auf diese Weise können auch weitere Einträge hinzugefügt werden.. Neben der reinen Kommandozeile gibt es auch Tools, die eine ``komfortablere" Eingabe zulassen (auch wenn's meiner Meinung nach nichts komfortableres als die Kommandozeile gibt). Diese werden in der zweiten LDAP-Serie beschrieben.

6.6 Bilder im Verzeichnis

Man kann Bilder im JPEG-Format ebenfalls in das Verzeichnis aufnehmen. Das folgende Beispiel zeigt, wie man zu einem Personeneintrag ein "Paßfoto" hinzufügt.

Angenommen, das Bild heißt "/tmp/pic.jpg". Nun muss der DN der betreffenden Person natürlich bekannt sein und man benötigt selbstverständlich Schreibzugriff auf das Verzeichnis.

Man erzeugt eine Datei, in der der DN und der Bild-Eintrag stehen. Diese "pic.datei" könnte wie folgt aussehen:

pic.datei

```
dn: cn=Thomas Bendler,ou=People,dc=selflinux,dc=de
jpegphoto: /tmp/pic.jpg
```

Nun kann man mit dem Kommando `ldapmodify` den Eintrag dieses DNs ändern (indem man das Attribut jpegphoto hinzufügt). Damit nicht der Bildpfad, sondern dessen Dateiinhalt in das Verzeichnis aufgenommen wird, muss dem ldapmodify der Parameter "-b" mitgegeben werden. Diese Option veranlaßt die ldap*-Werkzeuge, absolute Pfade als Binärdaten zu betrachten. Der Beispielaufruf:

```
ldapmodify \
-D "cn=Thomas Bendler,ou=Develpoment,o=Support,dc=selflinux,dc=de" \
-W -b < pic.datei
```

Noch ein Hinweis zur Arbeitserleichterung. Da gerade beim ersten Aufsetzen in der Regel nur ein Manager benötigt wird, empfielt es sich, diesen Account über die Parameter "rootdn" und "rootpw" in der slapd.conf einzustellen. Man bindet in diesem Fall über:

```
...
-D "cn=Manager,dc=selflinux,dc=de"
...
```

und spart etliches an Tipparbeit. In der Produktion ist dies dann jedoch unzulänglich, da ein Manager ja auch mal Urlaub macht, und auch in dieser Zeit jemand beispielsweise vergessene Paßwörter ändern können muss. Man benötigt hier also immer mehrere Manager. Die Parameter "rootdn" und "rootpw" sollte man dann auskommentieren und den slapd neustarten.

6.7 Ändern von Indizes

Werden in der slapd.conf neue Indizes konfiguriert, so gelten die Einstellungen natürlich sofort für neue Einträge. Die bereits vorhandenen Einträge sind in diesen neuen Indizes aber nicht enthalten. Suchanfragen führen in diesem Fall zu merkwürdig aussehenden Resultaten: Alle neuen Datensätze werden gefunden, jedoch nie alte. In solchen Fällen müssen die Indizes neu erzeugt werden. Dazu stoppt man den slapd, erzeugt sich eine LDIF-Kopie der Datenbank und indiziert diese (der Indexer kann nur LDIFs indizieren). Letztlich startet man den slapd neu.

Hier die Kommandos (ohne start/stop):

```
user@linux / # ldbmcat /var/lib/ldap/id2entry.gdbm > id2entry.dump
user@linux / # ldif2ldbm -i id2entry.dump
```

Dieser Vorgang kann einige Zeit in Anspruch nehmen, wenn man große Datenbestände hat. Technisch gesehen exportiert man die gesammte Datenbank und importiert sie anschließend.

6.8 Datensicherung

Es gibt mehrere Möglichkeiten, Daten zu sichern. Die eleganteste Lösung ist, mehrere sich untereinander replizierende LDAP-Server zu verwenden. Für kleine Installationen ist dies jedoch zu aufwendig. Leichter ist es, sich einfach mit ldapsearch alle Datensätze ausgeben zu lassen, und in einer Datei zu speichern. Ein besserer Weg ist, das Werkzeug `ldbmcat` zu verwenden, um sicherzugehen, wirklich alle Einträge zu erhalten. Um Inkonsistenzen zu vermeiden, sollte der slapd unbedingt gestoppt werden. Um die Ausfallzeit gering zu halten, kopiert man einfach die Datenbankdatei. Unter SuSE-Distributionen könnte man folgende Kommandofolge verwenden:

```
rcldap stop \
&& cp /var/lib/ldap/id2entry.gdbm id2entry-snap.gdbm ; \
rcldap start \
id2entry-snap.gdbm > id2entry-snap
```

Man erhält id2entry-snap.gdbm im Datenbankformat und eine LDIF-Datei id2entry-snap. Es empfielt sich letztere zu sichern, da über diese Datei notfalls auch in andere Verzeichnisserver zurückgesichert werden kann. Das GDBM-Format hingegen ist versions- und plattformabhänig.

Der Backup-Vorgang muss vorher unbedingt durchgetestet werden, um vor Überraschungen sicher zu sein. Dieser Test muss auch eine Rücksicherung einschließen, denn nur so kann man sicher sein, dass alles funktioniert. In der Praxis kann es sonst zu bösen Überraschungen kommen, denn das Verzeichnis wird schnell zu einem wichtigen Dienst in der

Organisation werden!

7 Tuning des LDAP-Servers

Es gibt unterschiedliche Möglichkeiten, den LDAP-Server zu tunen. Diese Möglichkeiten beziehen sich in erster Linie auf die LDBM-Datenbank. Deutliche Performancegewinne lassen sich aber erst in Verbindung mit großen Datenbeständen erzielen. Der "The SLAPD and SLURPD Administration Guide" bietet einen Überblick der Möglichkeiten zum Tunen des LDAP-Servers. Weitere Informationen finden sich im FAQ-O-MATIC auf der Homepage des OpenLDAP Projekt.