



---

## soluzione Script

Linux Trainingsunterlagen  
Einführung in LDAP (Auszug)

Leseprobe

---

**soluzione Script GmbH**  
Karl-Theodor-Straße 25  
80803 München

Telefon: 089 38 99 70-50  
Telefax: 089 38 99 70-77  
script@soluzione.de  
www.soluzione.de

© Copyright 2004 Alle Rechte vorbehalten. Kein Teil dieser Unterlage darf in irgendeiner Form (Druck, Fotokopie, Mikrofilm oder anderes Reproduktionsverfahren) ohne schriftliche Genehmigung der soluzione Script GmbH reproduziert oder unter Anwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.

Diese Unterlage wurde mit großer Sorgfalt erstellt und geprüft. Trotzdem können Fehler nicht vollkommen ausgeschlossen werden. Die Autoren, und soluzione Script können für fehlerhafte Angaben und deren Folgen weder eine juristische Verantwortung noch irgendeine Haftung übernehmen.

Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in dieser Unterlage berechtigt auch ohne besondere Kennzeichnung nicht zu der Annahme, daß solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürften.

SOLUZIONE

# Inhaltsverzeichnis

<b>1</b>	<b>LDAP Directory Server</b>	<b>3</b>
1.1	Einführung . . . . .	3
1.1.1	Directory-Clients und -Server . . . . .	4
1.1.2	Allgemeine Begriffe . . . . .	4
1.1.3	LDAP und X.500 . . . . .	5
1.2	Überblick LDAP-Modelle . . . . .	6
1.3	Das Informationsmodell - Objekte und Attribute . . . . .	7
1.4	Das Namensmodell . . . . .	15
1.4.1	Der Namensraum des Directory Information Tree (DIT) . . . . .	15
1.4.2	Verteiltes Directory—Partitionierung des Verzeichnisbaumes . . . . .	17
1.4.3	LDAP URLs . . . . .	22
1.5	Das Funktionsmodell . . . . .	23
1.5.1	Suchfilter . . . . .	24
1.5.2	Directory-Operationen . . . . .	25
1.6	Das Sicherheitsmodell . . . . .	26
1.6.1	TLS-Konfiguration . . . . .	27
1.6.2	SASL-Konfiguration . . . . .	27
1.6.3	Access Control Lists . . . . .	27
1.7	Das LDAP-Data-Interchange-Format (LDIF) . . . . .	29
1.8	Installation und Konfiguration des OpenLDAP-Servers . . . . .	30
1.8.1	Konfigurieren des LDAP-Servers . . . . .	31
1.9	LDAP-Clients und Hilfsprogramme . . . . .	36
1.10	LDAP-Replikation mit <b>slapd</b> und <b>slurpd</b> . . . . .	40
1.10.1	Replikation konfigurieren . . . . .	42
1.10.2	Umgang mit Replikationsfehlern . . . . .	44
1.11	Wissensfragen . . . . .	46
1.12	Übungen . . . . .	49
1.13	Lösungen . . . . .	50
1.14	Querverweise . . . . .	52

## INHALTSVERZEICHNIS

---

<b>2</b>	<b>Object Identifier, OID</b>	<b>53</b>
2.1	Eigene OID . . . . .	53
2.2	Schemata . . . . .	54
2.3	Übungen . . . . .	57
2.4	Lösungen . . . . .	58
<b>3</b>	<b>Erweiterte Serverkonfiguration</b>	<b>61</b>
3.1	Indizes . . . . .	61
3.2	Logging . . . . .	62
3.3	Limit . . . . .	63
3.4	Backend-Konfiguration . . . . .	64
3.5	Übungen . . . . .	67
3.6	Lösungen . . . . .	68
<b>4</b>	<b>Grafische Frontends</b>	<b>69</b>
4.1	GQ . . . . .	69
4.2	LdapBrowser . . . . .	72
4.3	Weitere GUIs . . . . .	76
4.4	Übungen . . . . .	77
<b>5</b>	<b>Stichwortverzeichnis</b>	<b>79</b>

---

# 1 LDAP Directory Server

In diesem Kapitel lernen Sie:

- Die Grundbegriffe von LDAP und deren Nutzen kennen,
- Die Konfiguration des LDAP-Servers OpenLDAP
- Die LDAP-Client-Konfiguration
- Das LDIF-Dateiformat, das zum Austausch von LDAP-Daten dient
- LDAP-Authentifizierungsserver und weitere Anwendungen einzurichten

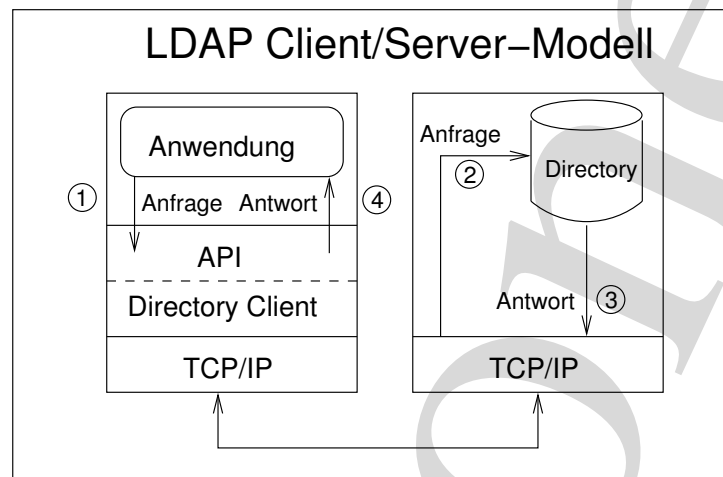
## 1.1 Einführung

Das Informationsmanagement spielt bei der steigenden Komplexität der Netzwerke eine immer größere Rolle. Beschreibungen von Usern, Anwendungen, Dateien, Druckern und anderen Ressourcen im Netzwerk werden oft in speziellen Datenbanken-Directories zusammengefaßt. In der Praxis wird immer mehr das Lightweight Directory Access Protocol (LDAP) für den einheitlichen Zugriff und für die Änderung der Directory-Information benutzt. LDAP ist ein Industriestandard, der in wachsender Anzahl von Anwendungen implementiert und immer mehr im Internet und in Firmen-Intranets als einheitliches Directory-Werkzeug akzeptiert wird.

**Der Begriff Directory** Das Directory (das Verzeichnis) ist eine, nach bestimmten Kriterien geordnete Liste von Informationen über verschiedene Objekte. Ein Beispiel ist das Telefonbuch: Es ist eine Liste von Namen, die alphabetisch geordnet sind, und jeder Eintrag beinhaltet Informationen, wie Telefonnummer und Adresse. Im Directory kann die Information nach verschiedenen Kriterien gesucht werden. Man kann direkt nach Namen, oder nach Information über eine Person (z.B. Adresse) suchen. Ein Directory, das von vielen Anwendern im Netz genutzt wird, kann eine einheitliche Sicht auf User, Ressourcen und andere Objekte im Netz anbieten. Damit können die Anwender auf ein Objekt mit Namen oder Funktion zugreifen, ohne die einzelnen Teile der Objektinformation (IP-Adressen, E-Mail-Adressen, File-Server) zu kennen.

**Unterschied zwischen Directory und Datenbank** Die Directories werden oft mit Datenbanken verglichen, sind aber spezialisierte Datenbanken. Sie sind für Lese- und Such-, aber nicht für Schreibanforderungen optimiert. Im Telefonbuch wird oft nach Namen gesucht, es wird aber nicht so oft geändert. Eine allgemeine Datenbank dagegen (z.B. Hotelreservierungen) muß auch oft geändert werden. Deswegen sind Directories zum Speichern von statischen Informationen gut geeignet, schlechter für Einträge, die sich oft ändern.

### 1.1.1 Directory-Clients und -Server



Die Directory-Server werden meistens sehr einfach implementiert. Es wird dabei auf Transaktionen oder Konsistenz-Prüfungen verzichtet. Deswegen sind Directories für das Speichern von empfindlichen Informationen nicht geeignet. Ein anderer Unterschied zwischen Datenbanken und Directories ist das Zugriffsverfahren. Die Datenbanken nutzen meistens eine sehr komplexe SQL-Sprache (Structured Query Language). LDAP dagegen nutzt ein vereinfachtes Zugriffsprotokoll. Das erlaubt auf LDAP-Basis, sehr einfache und kleine Client-Server-Anwendungen zu schreiben und schnell auf Informationen in verteilten Umgebungen zuzugreifen.

**Directory-Clients und -Server** Directories nutzen sehr oft das Client-Server-Modell. Die Anwendung (Directory-Client), die Informationen im Directory lesen oder schreiben möchte, schickt eine Anfrage zum Directory-Server-Prozeß. Er greift auf die Information im Directory zu und liefert das Ergebnis zurück. In einem verteilten Directory kann der Server seinerseits als Client andere Server abfragen.

### 1.1.2 Allgemeine Begriffe

**Verteilte Directories** Bei den verteilten Directories wird die Information von mehreren Servern bereitgestellt. Dabei kann die Information partitioniert oder repliziert sein. Wenn sie partitioniert ist, speichert jeder Server einen Teil der Information. Dabei wird jeder Directory-Eintrag nur in einem Server gespeichert. Beim Replizieren werden Directory-Entries in mehr als einem Server gespeichert.

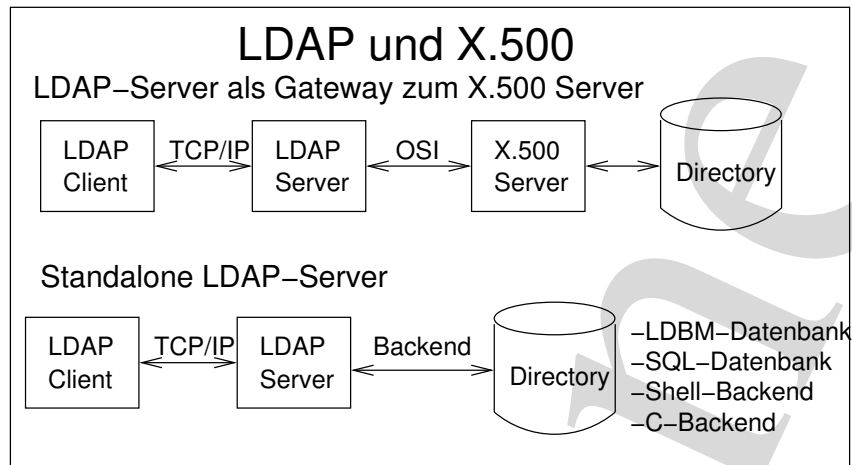
**Zugriffsrechte** Es werden hauptsächlich zwei Security-Mechanismen in die Directories eingebaut - Authentizität und Autorisierung. Die Authentizität überprüft die Identität des Users anhand des Benutzernamens und des Paßworts. Die Autorisierung wird mit den Access Control Listen (ACL) vergeben. Die ACL ist eine Liste von User- oder Gruppen-Zugriffsrechten, die an Objekte und Attribute geknüpft werden kann.

**Geschichte und Standards von LDAP** Im Rahmen der OSI-Protokolle wurde der X.500-Standard (Data Communications Network Directory Recommendations) für Directory Services entwickelt. Der X.500-Standard ist sehr komplex und definiert eine hierarchische Namensstruktur und leistungsfähige Suchmöglichkeiten. Außerdem spezifiziert er das Protokoll für die Kommunikation zwischen Directory-Client und -Server - das Directory Access Protocol (DAP). DAP liegt in der OSI-Anwendungsschicht und braucht für seine Funktionen alle darunterliegende OSI-Protokolle, was die Einführung von DAP erschwert hat. Als einfachere Alternative zu DAP wurde Lightweight-DAP (LDAP) entwickelt. LDAP setzt auf dem TCP/IP- statt auf dem OSI-Protokoll auf, vereinfacht einige X.500-Funktionen oder implementiert andere X.500-Funktionen gar nicht.

**LDAP: Protokoll oder Directory?** LDAP definiert das Protokoll- und Message-Format, mit denen der Client auf Daten im X.500-Directory zugreifen kann. Die Realisierung dieses Protokolls wird oft LDAP-Directory genannt. Eine mögliche Realisierung ist der LDAP-Gateway zwischen LDAP-Clients und X.500-Server.

### 1.1.3 LDAP und X.500

Der LDAP-Client greift auf X.500-Daten mit einer LDAP-Message zu. Der X.500-Server kann keine LDAP-Nachrichten verstehen. Dazu nutzen Client und Server unterschiedliche Kommunikationsprotokolle - TCP/IP und OSI. Deswegen kommuniziert der LDAP-Client mit einem LDAP-Server, der als Gateway zwischen LDAP und X.500 agiert. Der LDAP-Server selbst ist ein Client des X.500-Servers. Diese Architektur macht Sinn, wenn ein X.500-Server schon existiert. Es ist aber völlig überflüssig, für einen Stand-Alone-LDAP-Server auch einen X.500-Server zu installieren. Deswegen hat man LDAP-Server implementiert, die direkt mit dem Directory operieren. Sie sind komplizierter als die LDAP-Gateways zu X.500, aber immer noch viel einfacher als der X.500-Server selbst.



Von der Sicht des LDAP-Client sind beide Arbeitsweisen gleich - er kommuniziert letztendlich mit einem LDAP-Server.

**LDAP-Kommunikation** Die Kommunikation zwischen LDAP-Client und -Server verläuft in folgenden Schritten:

- Der Client eröffnet eine TCP-Session (bekannt als *binding*) mit dem LDAP-Server. Dem Client muß die IP-Adresse/der Host-Name und TCP-Port (meistens 389) des LDAP-Servers bekannt sein. Der Client kann sich vor dem Server mit User-Name und Paßwort oder als Anonymous-User ausweisen und bekommt die entsprechenden Berechtigungen. Server und Client können auch ein Verschlüsselungsalgorithmus für den Datenaustausch vereinbaren.
- Der LDAP-Client führt eine Operation wie Lesen, Ändern oder Suchen von Directory-Daten durch.
- Wenn der Client mit der Operation fertig ist, schließt er die Session (bekannt als *unbinding*) mit dem Server.

## 1.2 Überblick LDAP-Modelle

In der Geographie versteht man ein sehr komplexes Gebilde wie unsere Erde dadurch, daß man Karten erstellt, die jeweils einen Teil der Erde genau beschreiben. Diese Karten nennt man in der Informatik ein *Modell*, denn sie beschreiben eine bestimmte Sichtweise. Dann nimmt man eine Anzahl Karten und faßt sie in einen Atlas zusammen, so daß sie eine Beschreibung der ganzen Erde liefern. Genauso gehen wir mit LDAP vor.

## 1.3 Das Informationsmodell - Objekte und Attribute

LDAP kann am besten mit folgenden vier Modellen (Sichtweisen) beschrieben werden.

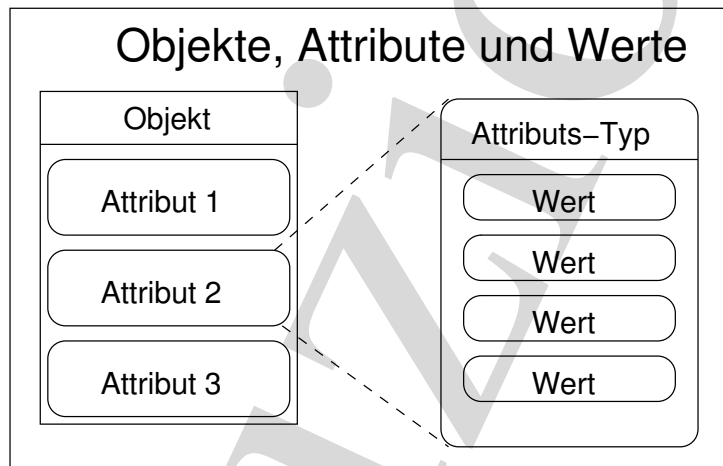
**Informationsmodell** beschreibt die Struktur der Information im LDAP-Directory.

**Namensmodell** beschreibt die Organisation der Information.

**Funktionsmodell** beschreibt, welche Operationen mit den Directory-Informationen möglich sind.

**Sicherheits-Modell** beschreibt, wie die LDAP-Information geschützt werden kann.

### 1.3 Das Informationsmodell - Objekte und Attribute



Das Basiselement der Directory-Information ist der Eintrag (engl. „entry“). Die Einträge repräsentieren die Objekte im Directory, wie Menschen, Organisationen, Länder usw. Die Objekte bestehen aus Attributen, die Objektinformationen beinhalten. Jedes Attribut hat einen Typ und einen oder mehrere Werte. Jeder Attributtyp ist mit einer Syntax verbunden. Sie beschreibt die Art von Werten, die in diesem Typ gespeichert werden können.



*Beispiel:* Im LDAP-Directory wird als Objekt die Firma Example gespeichert. Dieses Objekt hat Attribute, wie Adresse, Telefonnummer, Geschäftsstellen usw. Das Attribut Adresse kann z.B. vom Typ Directory String sein und den Wert „Karl-Theodor-Straße 25, 80803 München, Deutschland“ haben.

OpenLDAP Versionen vor 2.0 verwenden die simple und leicht lesbare „University of Michigan“-Schreibweise. Seit OpenLDAP 2.0 verwendet man die viel komplexere,

aber auch mächtigere *ASN.1*-Notation, was für *Abstract Syntax Notation* steht. Diese Notation wird übrigens auch für SNMP verwendet.

Unter OpenLDAP 2.0 findet man die Attributdefinitionen in den Dateien im Verzeichnis `/etc/openldap/schema/*`.

Attribute werden in ASN.1 wie folgt definiert (Hier sind der Übersichtlichkeit halber nur die wichtigsten Teile abgedruckt):

### Syntax:

```
attributetype ( OID NAME ( 'Name' 'Aliasname' )
  [ DESC 'Beschreibung' ]
  [ SUP Name oder OID des Mutterattributes ]
  [ EQUALITY Prüfverfahren für Gleichheit ]
  [ ORDERING Prüfverfahren für Sortieren ]
  [ SUBSTR Prüfverfahren für Teilvergleiche ]
  [ SYNTAX Syntax-OID{Datenlänge in Bytes} ]
  [ SINGLE-VALUE ]
  [ NO-USER-MODIFICATION ]
  [ USAGE Verwendungszweck ]
)
```

Die obigen Elemente der Attributdefinition haben folgende Bedeutung:

**OID** ist der *Object Identifier*. Dieser bewirkt eine eindeutige Bezeichnung einer ASN.1-Objektes wie in unserem Fall Attribute und Objektklassen. Die OIDs werden hierarchisch vergeben, die Hierarchieebenen werden durch Punkte getrennt.

OID-Stamm	Bedeutung
1.1	OIDs von Firmen
1.1.1	SNMP-Elemente
1.1.2	LDAP-Elemente
1.1.2.1	AttributTypes
1.1.2.2	ObjectClasses

Man sollte *niemals* Phantasie-OIDs verwenden. Dafür gibt es sie bei der IANA kostenlos. Einfach das Web-Formular (Link siehe Querverweise) ausfüllen und in einer Woche bekommen Sie Ihren eigenen OID-Stamm zugewiesen, unterhalb diesem Sie dann schalten und walten können, wie Sie wollen.

**NAME** ist der Name des Attributes, dabei kann ein zweiter Name (Aliasname) angegeben werden.

**DESC** liefert eine menschenlesbare Beschreibung des Attributs.

---

## 1.3 Das Informationsmodell - Objekte und Attribute

**EQUALITY** gibt das Prüfverfahren an, nach dem auf Gleichheit des Attributs geprüft wird. Es sind folgende Prüfverfahren möglich:

Verfahrensname	Funktionsweise
booleanMatch	Booleschen Wert vergleichen („Null oder Eins“)
objectIdentifierMatch	OID vergleichen
distinguishedNameMatch	DN vergleichen
uniqueMemberMatch	DN mit optionaler UID
numericStringMatch	Numerischer Vergleich
caseIgnoreMatch	Zeichenkettenvergleich: Unempfindlich gegenüber Groß/Kleinschreibung und Leerzeichen
caseExactMatch	Zeichenkettenvergleich: Empfindlich gegenüber Groß/Kleinschreibung, unempfindlich gegenüber Leerzeichen



*Hinweis:* Es existieren für ASCII-Zeichenketten die absolut gleichen Verfahren `numericStringIA5Match`<sup>1</sup>, `caseIgnoreIA5Match` und `caseExactIA5Match`

**ORDERING** gibt das verwendete Prüfverfahren für Sortiervorgänge an:

Verfahrensname	Funktionsweise
numericStringOrderingMatch	Numerischer Vergleich
caseIgnoreOrderingMatch	Zeichenkettenvergleich: Unempfindlich gegenüber Groß/Kleinschreibung und Leerzeichen
caseExactOrderingMatch	Zeichenkettenvergleich: Empfindlich gegenüber Groß/Kleinschreibung, unempfindlich gegenüber Leerzeichen




*Hinweis:* Es existieren für ASCII-Zeichenketten die absolut gleichen Verfahren `numericStringOrderingIA5Match`, `caseIgnoreOrderingIA5Match` und `caseExactOrderingIA5Match`

**SUBSTR** gibt das verwendete Prüfverfahren für Teilzeichenkettenvergleiche an:

---

<sup>1</sup>IA5=International ASCII 5

Verfahrensname	Funktionsweise
numericStringSubstringsMatch	Numerischer Vergleich
caseIgnoreSubstringsMatch	Zeichenkettenvergleich: Unempfindlich gegenüber Groß/Kleinschreibung und Leerzeichen
caseExactSubstringsMatch	Zeichenkettenvergleich: Empfindlich gegenüber Groß/Kleinschreibung, unempfindlich gegenüber Leerzeichen

*Hinweis:* Es existieren für ASCII-Zeichenketten die absolut gleichen Verfahren `numericStringSubstringsIA5Match`, `caseIgnoreSubstringsIA5Match` und `caseExactSubstringsIA5Match` 

**SYNTAX** gibt die OID der verwendeten Syntax und die Datenlänge des Attributes in Bytes an. Folgende Syntaxes sind möglich:

Name	OID	Bedeutung
binary	1.3.6.1.4.1.1466.115.121.1.5	Binäre Daten
boolean	1.3.6.1.4.1.1466.115.121.1.7	Boolescher Wert
distinguishedName	1.3.6.1.4.1.1466.115.121.1.15	DN
directoryString	1.3.6.1.4.1.1466.115.121.1.15	UTF-8 String
IA5String	1.3.6.1.4.1.1466.115.121.1.26	ASCII String
Integer	1.3.6.1.4.1.1466.115.121.1.27	Ganzzahl
Name and Optional UID	1.3.6.1.4.1.1466.115.121.1.34	DN plus UID
Numeric String	1.3.6.1.4.1.1466.115.121.1.36	Numerischer String
OID	1.3.6.1.4.1.1466.115.121.1.38	Object Identifier
Octet String	1.3.6.1.4.1.1466.115.121.1.40	Beliebige Bytes
Printable String	1.3.6.1.4.1.1466.115.121.1.44	Druckbarer String

**SINGLE-VALUE** läßt nur *einen* Wert pro Attribut zu. Standardmäßig sind mehrere Werte möglich.

**NO-USER-MODIFICATION** erlaubt keine Modifikation des Attributs durch den Benutzer.

**USAGE** Definiert die Verwendung des Attributs. USAGE kann folgende Werte annehmen:

**userApplications** Das Attribut dient für eine normale Anwendung.

**directoryOperation** Das Attribut dient für den Betrieb des Directories. Darunter fallen zum Beispiel Timestamps, in denen die Daten der letzten Änderung festgehalten werden.

**distributedOperation** Das Attribut dient für verteilte Directories. Eine Referenz (Referral) auf einen Directory Server hat diese Verwendung. (Attribut ref)

**dSAOperation** Das Attribut dient zum Serverbetrieb. Hierzu zählen Paßwortattribute und Zugriffskontrolllisten (ACLs)



*Beispiele:*

- Ein ganz grundlegendes Attribut ist name, von dem viele andere Attribute, wie zum Beispiel cn, sn, o, ou und c abgeleitet sind:

```
attributetype ( 2.5.4.41 NAME 'name'  
                EQUALITY caseIgnoreMatch  
                SUBSTR caseIgnoreSubstringsMatch  
                SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{32768} )
```

Das Attribut name hat den OID 2.5.4.41, Groß/Kleinschreibung spielt beim Test auf Gleichheit (EQUALITY) und beim Vergleich von Teilzeichenketten (SUBSTR) keine Rolle. Das Attribut ist 32768 Bytes lang und entspricht der SYNTAX für UTF-8 Zeichenketten oder DN-Angaben.

- Davon abgeleitet ist cn:

```
attributetype ( 2.5.4.3 NAME ( 'cn' 'commonName' ) SUP name )
```

cn hat die OID 2.5.4.3, den Aliasnamen commonName und erbt die übrigen Eigenschaften vom Attribut name. (SUP name)

- Eine Domänen-Komponente dc ist wie folgt definiert:

```
attributetype ( 0.9.2342.19200300.100.1.25  
                NAME ( 'dc' 'domainComponent' )  
                DESC 'RFC1274/2247: domain component'  
                EQUALITY caseIgnoreIA5Match  
                SUBSTR caseIgnoreIA5SubstringsMatch  
                SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE )
```

Sie hat den OID 0.9.2342.19200300.100.1.25, den langen Namen `domainComponent`, Groß/Kleinschreibung wird bei Gleichheits- und Teilzeichenkettentests nicht unterschieden und die SYNTAX ist `IA5String`<sup>2</sup>. Damit sind nur ASCII-Zeichen zugelassen.

- Eine Referenz (englisch: Referral) auf einen LDAP-Server, der einen Teilbaum enthält, bewirkt das Attribut `ref`; Sie läßt sich mit der URL-Angabe in einem Hyperlink vergleichen.

```
attributetype ( 2.16.840.1.113730.3.1.34 NAME 'ref'  
              DESC 'Named referral'  
              EQUALITY caseExactIA5Match  
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.26  
              USAGE distributedOperation )
```

Bei `ref` wird der ASCII-Zeichensatz verwendet und das Attribut wird für verteilte Directories verwendet (USAGE `distributedOperation`).

**Objekte und Schemata** Nun, wenn man eine Ansammlung von Attributen definiert hat, dann kann man aus diesen Attributen *Objekte* gestalten. Die Zusammensetzung von Objekten beschreibt man in einer *Objektklasse*.

Eine Sammlung zusammengehöriger Objektklassen und Attribute nennt man ein *Schema*. Ein Schema, also fest definierte Objektklassen und Attribute, haben den Sinn, daß auf dem LDAP-Server standardmäßig Schema-Überprüfung stattfindet. Dies zwingt die Benutzer, Ihre Daten genau nach den Attribut- und Objektdefinitionen ins Directory einzugeben, sonst wird ein Fehler zurückgeliefert. Damit wird der Wildwuchs unstrukturierter Daten vermieden, denn für die maschinelle Auswertung und Verwertung von Daten sind eben klar definierte Schemata notwendig.

Eine Objektklasse besteht aus folgenden Angaben:

- Objektklassenname
- Attribute, die im Objekt enthalten sein *müssen*
- Attribute, die im Objekt enthalten sein *können*
- Eine Objekt-Identifikationsnummer (OID). Damit werden ASN.1-Objekte weltweit eindeutig identifiziert.

---

<sup>2</sup>IA5=International ASCII 5

- Eine Angabe über die *Art* der Objektklasse:
  - *Abstrakte Objektklassen (ABSTRACT)* wie zum Beispiel die Objektklasse `top` haben nur den Sinn, daß andere Objektklassen von ihnen abgeleitet werden.
  - *Hilfsobjektklassen (AUXILIARY)* definieren eine Sammlung von Attributen und stellen zusätzliche Eigenschaften dar, die man zu einem anderen Objekt hinzufügen kann.  
Ein Beispiel ist die Objektklasse `strongAuthenticationUser`, die einen Benutzer um die Fähigkeit zu starker Authentifizierung erweitert.
  - *Strukturelle Objektklassen (STRUCTURAL)* sind normale Objektklassen für den Einsatz in der Praxis. Indem man eigene Objektklassen von diesen ableitet, kann man sie seinen Bedürfnissen anpassen.

Objekte können Attribute von anderen Objekten erben. Dabei kann ein Objekt nur von *einem* anderen Objekt abgeleitet werden. Mehrfachvererbung ist also nicht möglich. Dies ist besonders nützlich, wenn eben schon ein Standardobjekt den Bedürfnissen fast schon entspricht, denn dann kann man das Standardobjekt einfach um die zusätzlichen Attribute erweitern.

Die Definitionen aller dem LDAP-Server bekannten Objekte findet in der Datei `/etc/openldap/slapd.oc.conf` statt. Dabei wollen wir im Folgenden einige wichtige Beispiele daraus erläutern.



*Beispiele:*

- Das Objekt `top` bildet ein abstraktes Mutterobjekt, von dem viele andere Objekte abgeleitet sind. Es enthält nichts außer einem Attribut namens `objectClass`. Die Werte dieses Attributs geben an, von welchen Objekten ein Objekt abgeleitet ist. Es ist unter OpenLDAP 2.0 (ASN.1) wie folgt definiert:

```
objectclass ( 2.5.6.0 NAME 'top' ABSTRACT
              MUST objectClass )
```

Die Objektklasse `top` hat den OID `2.5.6.0`, ist eine abstrakte Klasse, die lediglich das Attribut `objectClass` enthalten *muß*.

- Die Objektklasse `person` beschreibt eine Person:

```
objectclass ( 2.5.6.6 NAME 'person' SUP top STRUCTURAL
              MUST ( sn $ cn )
              MAY ( userPassword $ telephoneNumber $ seeAlso $ description ) )
```

Hier sind die Attribute `objectClass`, `sn` und `cn` zwingend notwendig, `description`, `seeAlso`, `telephoneNumber` und `userPassword` sind optional. Letzteres wird dann verwendet, wenn der Zugang zum Directory mit Paßwörtern geschützt ist.

Die ASN.1-Definition ist hier recht kurz, denn `person` wird einfach von `top` abgeleitet. Sie hat den OID 2.5.6.6, ist eine Strukturelle Klasse (also eine Klasse für die normale Anwendung). Die Attribute sind natürlicherweise diesselben wie im alten Format. Als Trennzeichen dient hier ein `$`-Zeichen.

Allgemein notiert man Objektklassen nach ASN.1 wie folgt (nur wichtigste Elemente angeben):

### Syntax:

```
objectclass (  
    OID  
    [ NAME 'Name' ]  
    [ DESC 'Beschreibung' ]  
    [ SUP Namen oder OIDs der Mutterklassen ]  
    [ ( ABSTRACT oder STRUCTURAL oder AUXILIARY ) ]  
    [ MUST ( Attribut $ Attribut ... ) ]  
    [ MAY ( Attribut $ Attribut ... ) ]  
)
```

Die Felder haben folgende Bedeutung:

**NAME** Der Name der Objektklasse

**DESC** Die Beschreibung

**SUP** Der Name oder die OID der Mutterklassen, von denen die Objektklasse abgeleitet wurde und deren Attribute sie erbt

**ABSTRACT/STRUCTURAL/AUXILIARY** Die Angabe, ob es sich um eine abstrakte Klasse, eine strukturelle Klasse oder um eine Hilfsklasse handelt. Ist keiner dieser Begriffe angegeben, so gilt die Objektklasse als strukturelle Klasse.

**MUST** gibt die Attribute an, die das Objekt enthalten *muß*. Mehrere Attribute werden durch ein `$`-Zeichen getrennt.

**MAY** gibt die Attribute an, die das Objekt enthalten *kann*.

### 1.4 Das Namensmodell

Der LDAP-Namensmodell beschreibt, wie die Einträge organisiert sind und wie sie identifiziert werden. Die Einträge sind in einer baumähnlichen Struktur, bekannt als *Directory Information Tree (DIT)*, organisiert. Die Einträge werden im DIT auf Basis der sog. *Distinguished Name (dn)* plaziert.

**DN und RDN** Der Distinguished Name (DN) identifiziert eindeutig einen Eintrag und ist ein primärer Schlüssel in einem Directory, was bedeutet, daß er einen Datensatz eindeutig identifiziert. Der DN besteht aus einer Sequenz von *Relative Distinguished Names (RDNs)*. Jeder RDN entspricht einem Zweig im DIT. Man kann den DN mit einem vollen Dateipfad und RDN mit den einzelnen Verzeichnisse in diesem Pfad vergleichen.

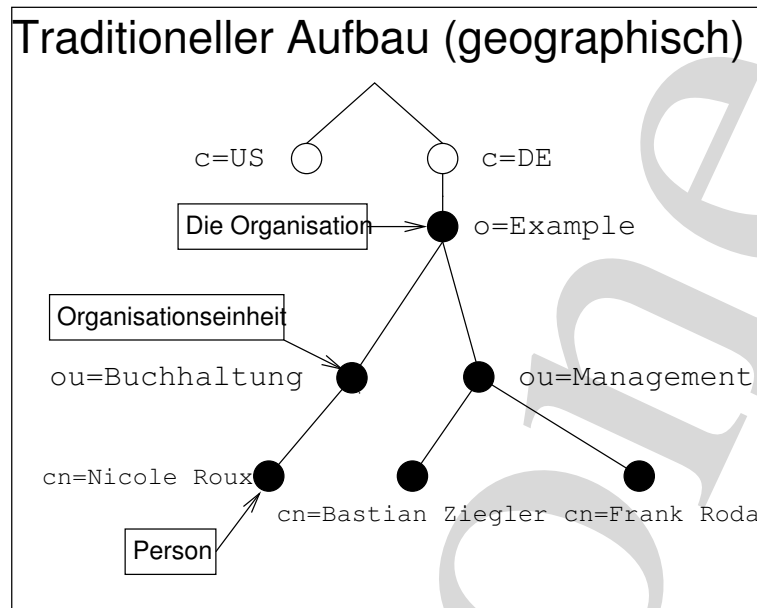
#### 1.4.1 Der Namensraum des Directory Information Tree (DIT)

<u>Attributtyp</u>	<u>Name im DIT</u>
CommonName	cn
LocalityName (Lokationsname)	l
OrganizationName	o
OrganizationalUnitName	ou
CountryName	c
StateName	st
StreetAddress	street
domainComponent	dc
userid	uid

Wie man seinen Verzeichnisbaum aus diesen Komponenten aufbaut, ist letztendlich jedem selbst überlassen. Jedoch haben sich für Firmen zwei beliebte Strukturen für den Verzeichnisbaum herauskristallisiert<sup>3</sup>. Zum einen kennt man den traditionellen oder geographischen Aufbau:

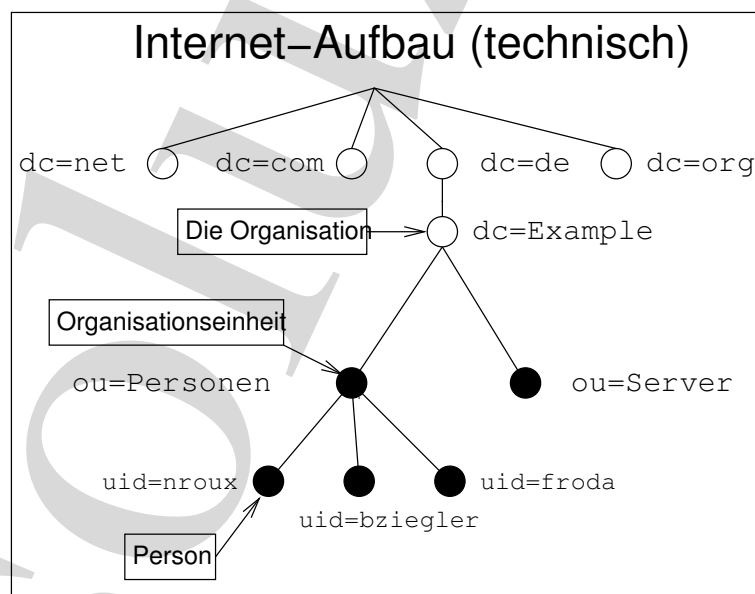
---

<sup>3</sup>Die Verzeichnisstruktur nennt man auch *Namensraum*



Beim Traditionellen Aufbau erfolgt die Gliederung mit Attributen, die die Benutzer-sicht widerspiegeln. Der DIT ist nach `c` (Land), `o` (Organisation), `ou` (Organisationseinheit: Abteilungen) und `cn` (Namen) gegliedert. Dieses Setup ist sinnvoll, wenn die Daten im Directory überwiegend aus organisatorischen Daten wie etwa Mitarbeiterdaten wie Namen, Telefonnummern und Adressen bestehen.

Alternativ dazu kann man auch den Internet-Aufbau oder technischen Aufbau wählen:



Hier lehnt sich die Bezeichnungsweise an die technische Struktur der Firma an. Die Gliederung erfolgt zunächst nach den Domänen-Komponenten `dc=de` und `dc=example`, dann in die Organisationseinheiten `ou=Server` und `ou=Personen`; in letzterer sind wiederum die Personen mit dem RDN `uid=Login-Name` abgelegt. Man verwendet diese Bezeichnungsweise, wenn das Directory überwiegend technische Daten enthält. Meistens laufen die Anwendungen, die auf das Directory zugreifen, hinter den Kulissen der Benutzeroberfläche ab.

**Notation von RDN und DN** Der RDN wird in dieser Form notiert: „*Name vom Attribut = Wert*“. Bei den Attributen wird kein Unterschied zwischen Groß- und Kleinbuchstaben gemacht. Der DN besteht aus der Liste der RDNs, getrennt mit Komma. Spezielle Zeichen ( , = + << >> # ; ) oder Leerzeichen am Anfang und Ende vom RDN müssen mit dem \-Zeichen maskiert werden.

**User Friendly Name (UFN)** Einige LDAP-Produkte erlauben, die DN-Notation zu vereinfachen. Wenn man in der DN-Notation die Attributnamen und das =-Zeichen weglässt, entsteht der sog. *User Friendly Name (UFN)*. Solange diese Notation eindeutig ist, kann sie verwendet werden. Immer eindeutig ist aber nur die DN-Notation.



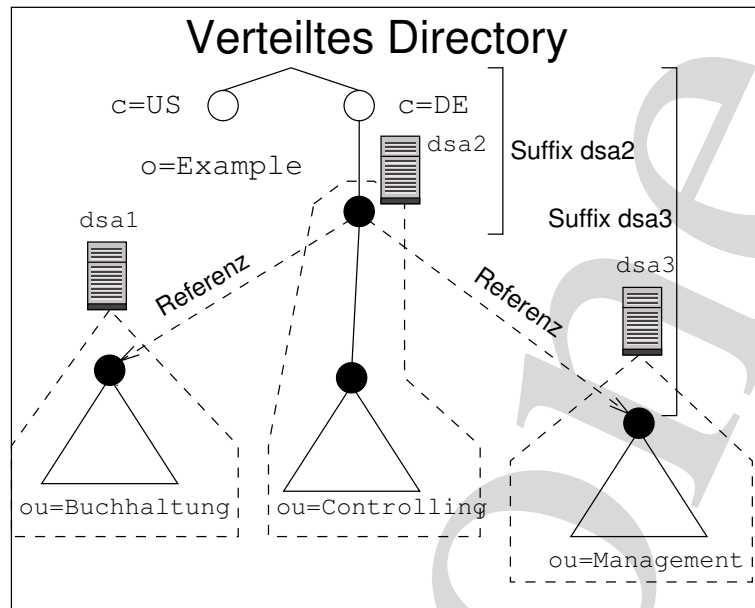
*Beispiel für UFN:* Der DN `cn=Meier,ou=finanzen,o=Example,c=DE` wird in der UFN-Form als `Meier,finanzen,Example,de` notiert. Die Einträge im DIT werden meistens nach geographischen oder organisatorischen Gesichtspunkt angelegt. Objekte, die ganze Länder repräsentieren, stehen an der Spitze des DIT. Unter den Ländern können Bundesländer, nationale Organisationen usw. stehen. Darunter können Abteilungen oder Suborganisationen stehen. Ganz unten werden Endobjekte, wie Menschen, Computer usw. eingeordnet. Die Namen der Einträge entsprechen ihren Positionen im DIT. Der Eintrag ganz unten rechts im Beispiel heißt `cn=Jorg Meier,o=Example,c=DE`. Der DN wird immer in der Richtung vom Endknoten zum Root gelesen (die Pfadangabe in einer Datei läuft umgekehrt - vom Root zum Dateiname). Der DN besteht aus dem RDN des Eintrags und den DN des Vorgängereintrags.



*Beispiel für DN und RDN:* Der DN `cn=Jorg Meier,o=Example,c=DE` besteht aus dem RDN `cn=Jorg Meier` und den Suffix `o=Example,c=DE`.


### 1.4.2 Verteiltes Directory—Partitionierung des Verzeichnisbaumes

Da LDAP ähnlich wie DNS ein verteilter Verzeichnisdienst ist, besteht die Möglichkeit, genau wie bei DNS, Unternehmensräume auf eigene Server zu verteilen. Dieses Vorgehen nennt man *Partitionierung* des Directory-Baumes:

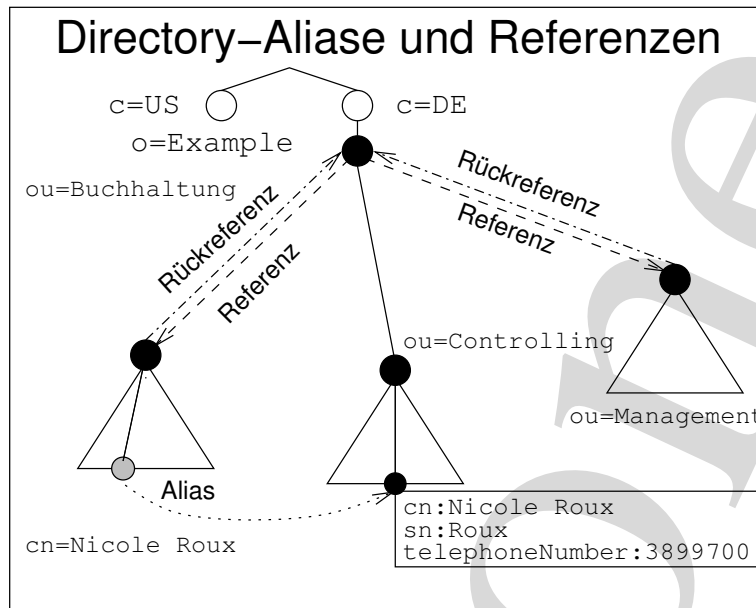


Im obigen Beispiel haben wir den DIT der Firma example in drei Partitionen unterteilt, die jeweils auf den Servern `dsa1.example.de`, `dsa2.example.de` und `dsa3.example.de` liegen. Die Unternehmenswurzel `o=Example, c=DE` wird auf dem Server `dsa2` gehalten. Unterhalb dieses Mutterknotens befindet sich die Organisationseinheit `ou=Controlling`. Die Organisationseinheiten `ou=Buchhaltung` und `ou=Management` sind nicht auf diesem Server gespeichert.

**Suffixe** Ein LDAP-Server muß nicht unbedingt den ganzen DIT speichern. Er muß z.B. die Einträge einer Abteilung, aber nicht alle übergeordneten Knoten, noch unbedingt alle Unterbäume verwalten. Als *Suffix* bezeichnet man den höchsten Eintrag in einem Server. Wie bei DNS kann jedoch ein Server mehrere Directories mit mehreren Suffixes beherbergen.

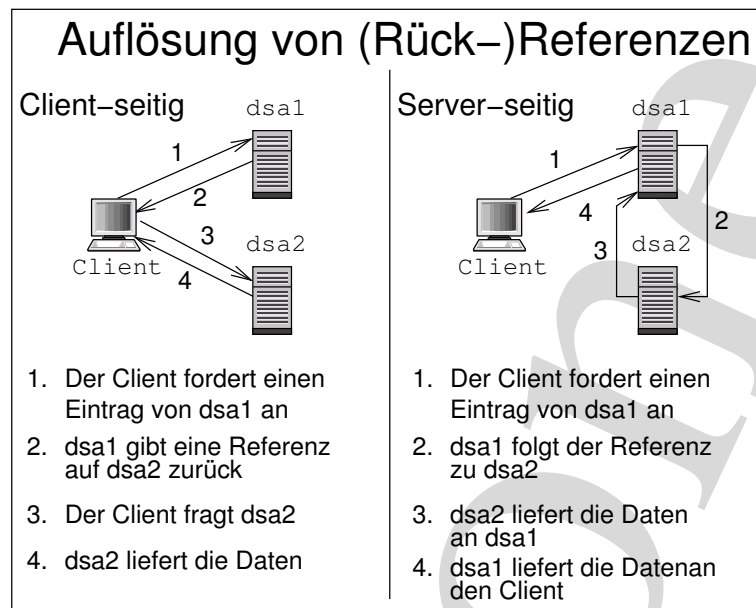
*Beispiel:* In der obigen Abbildung hat der Server `dsa1` den Suffix `ou=Buchhaltung, o=Example, c=DE`, der Server `dsa2` den Suffix `o=Example, c=DE` und der Server `dsa3` den Suffix `ou=Management, o=Example, c=DE`. 

**Aliase und Referenzen** Durch die Definition eines *Alias* kann die DIT-Struktur von der Baumstruktur abweichen. Aliase werden benutzt, wenn Einträge zu mehreren Teilverzeichnissen gehören oder zu komplex sind. In der Abbildung übernimmt die Controllerin Nicole Roux auch noch zusätzliche Aufgaben in der Buchhaltungsabteilung. Deswegen existiert in der Buchhaltung ein Alias, das auf den Eintrag im Controlling-Directory verweist. So wird vermieden, daß man dieselben Daten mehrfach im Baum ablegen muß. Damit vermeidet man Dateninkonsistenzen.



Um ein verteiltes Directory mit dem ganzen DIT-Baum zu bilden, müssen Server, die nur Teile vom DIT speichern, miteinander verbunden sein. Das wird über *Referenzen* erreicht, die eine Art Pointer zum nächsten Server sind.


Im obigen Beispiel verwaltet der Server `dsa2` die Organisationseinheiten `ou=Buchhaltung` und `ou=Management` nicht, sondern enthält lediglich Verweise auf die Server `dsa1` bzw. `dsa3`. Einen Nachteil hat diese oben dargestellte Konfiguration jedoch: Alle Clients stellen die Anfragen an den Server `dsa2`, der damit am stärksten belastet ist. Sinnvoller ist dagegen, daß der Abteilungsserver die Anfragen seiner Clients direkt bedient, wodurch man eine Lastverteilung erreicht. Die Abteilungsserver `dsa1` und `dsa3` sind so konfiguriert, daß sie alle unbekanntes DNS automatisch an den Server `dsa2` weitergeben.



Wenn ein Client eine Anforderung an einen LDAP-Server sendet, kann der Server mit einer Referenz zu einem anderen LDAP-Server antworten. Der Client kann die Anfragen an den neuen LDAP-Server richten. Die Referenzen werden nicht vom LDAP-Server aufgelöst.

**Untergeordnete Informationsbäume - *subordinated*** Untergeordnete Informationen bedeuten, daß man einen Zweig des Verzeichnisses aus seinem Baum an einen anderen Server delegiert. Dieser Zweig wird dann dort verwaltet. Dazu wird ein spezielles Referenzobjekt erzeugt, daß dann auf den anderen Server verweist. Dieses Objekt ist nichts anderes als ein Zeiger.

Das Zeigerobjekt besteht aus einer strukturellen Objektklasse, `referral`, und dem „Distinguished Name“ der den Namen des ausgelagerten Zweiges trägt. Dazu hat das Objekt zwei Objektklassen: `referral`, eine Objektklasse die besagt, daß dies nur ein Zeiger ist, und `extensibleObject`, welche die Angabe eines relativen Distinguished Name ermöglicht.

*Beispiel:* Ein Server `ldap.linux.de` verwaltet folgenden Baum: `dc=linux,dc=de`. Sie möchten nun ein Teilbaum mit Accounts von einer kommerziellen LDAP-Lösung einbinden, um sie nutzen zu können. 

Der Rechner mit der kommerziellen Anwendung heißt `ldap2.linux.de`. Der Zweig, den Sie bei Sich einbinden wollen, ist dort unter `ou=Accounts,dc=linux,dc=de` zu finden. Dann müßten Sie auf dem Server `ldap.linux.de` folgendes Objekt anlegen:

```
dn: dc=Accounts,dc=linux,dc=de
```

```
objectClass: referral
objectClass: extensibleObject
dc: Accounts
ref: ldap://ldap2.linux.de/ou=Accounts,dc=linux,dc=de
```

Der Server nutzt diese Information, folgt dem Zeiger und sucht auch im untergeordneten externem Zweig bei einer Anfrage. Der Server übernimmt also die Auflösung der Suche.

**Übergeordnete Informationsbäume - Superior** Ein übergeordnetes Verzeichnis wird über die `referral`-Direktive in der Konfigurationsdatei `slapd.conf` konfiguriert. Diese Direktive ist eine Liste mit übergeordneten Verzeichnisbäumen.

Wenn ein Server z.B. ausschließlich den Teilbaum `ou=Accounts,dc=linux,dc=de` betreut, dann wäre seine übergeordneter Verzeichnisbaum `dc=linux,dc=de`. Damit könnte der Server `ldap2.linux.de` mit einer `referral`-Direktive auf den übergeordneten Baum verweisen.

```
referral ldap://ldap.linux.de/
```

Diese Referenz wird dem Client vom Server mitgeteilt. Das heißt der Client sucht nun auf dem anderem Server weiter. Dazu muß der Client das Verfolgen von Referrals unterstützen und es aktivieren.



*Beispiel:* Wie `ldapsearch` mit Referrals umgehen soll, kann man mit der Option `-a` festlegen:

```
ldapsearch -x -a always ...
```

Für den Server legt man dies in `ldap.conf` mit `DEREF` fest:

```
DEREF always
```

Die angegebenen Werte haben hierbei folgende Bedeutung:

**never:** Referenzen werden nicht aufgelöst.

**searching:** Nur im entfernten Baum nach der Referenz suchen.

**finding:** Nur im lokalen Baum nach der Referenz suchen.

**always:** Referenzen im eigenen und anderen Bäumen auflösen.



*Hinweis:* Um einem Referral zu folgen muß entweder `always` oder `searching` aktiv sein. Lokales Auflösen von Referenzen ist nur bei Aliassen von Bedeutung.

**Server-Information** Der LDAP-Server speichert im DIT auch Informationen über sich selbst. Die Attribute eines speziellen Eintrags mit leeren DN (Länge Null) beschreiben folgende Server-Eigenschaften:

- Die Suffixe, die der Server speichert.
- Die LDAP-Version
- Liste der unterstützten Sicherheitsmechanismen
- Liste der alternativen LDAP-Server.

### 1.4.3 LDAP URLs

Die Uniform Resource Locators (URL) ist eine Standardadressierung für Ressourcen im Internet.

*Beispiele:*

```
http://www.Example.de  
ftp://ftp.novell.de/pub
```

Auch für LDAP-Ressourcen wurde ein LDAP-URL mit folgender Schreibweise definiert:

```
ldap://Host/DN?Attribute?Suchtiefe?Filter
```

Es ist möglich, einige der Felder — `Attribute`, `Suchtiefe` und `Filter` leer zu lassen — das `?`-Zeichen muß aber bleiben. Die Felder haben folgende Bedeutung:

**Host[:Port]** ist der Host-Name und die TCP-Port-Nummer vom LDAP-Server. Der Default Port 389 kann weggelassen werden.

**DN** der Basis-DN für die Suche.

**Attribute** welche Attribute ausgegeben werden sollen. Wenn das Feld leer ist, werden alle Attribute ausgegeben.

**Suchtiefe (Scope)** die Suchtiefe hat die Werte `sub`, `one`, `base`.

**Filter** gibt den Suchfilter an. Ein leerer Filter wird mit `objectclass=*` ersetzt und liefert alle Einträge als Ergebnis.

*Beispiele:*



**ldap://10.2.1.200/cn=jorg meier,ou=finanzen,o=Example,c=de**  
gibt alle Attribute von Jorg Meier aus.

**ldap://10.2.1.200/o=Example,c=de??sub** gibt alle Einträge im DIT unter o=Example, c=de aus.

**ldap://10.2.1.200/o=Example,c=de??sub?cn=\*jorg\*** sucht im DIT unter o=Example, c=de nach Common-Name \*jorg\*.

**ldap://10.2.1.200/o=Example,c=de?telephonenumber?sub?cn=\*jorg\***  
Es wird nur die Telefonnummer von Jorg ausgegeben.

**ldap://10.2.1.200/cn=config** zeigt die LDAP-Server-Konfiguration (z.B. LDAP-Suffixe) an.

**ldap://10.2.1.200/cn=monitor** zeigt Laufzeitstatistiken des LDAP-Servers an.

## 1.5 Das Funktionsmodell

**Suchen im Directory und Suchparameter** Im LDAP-Funktionsmodell werden die Operationen zum Suchen und Ändern der Directory-Einträge beschrieben. Man kann diese Operationen in den folgenden drei Kategorien betrachten:

**Suchen** diese Such- oder Vergleichsoperationen holen Informationen aus dem Directory.

**Update** erlauben das Einfügen, Löschen und Modifizieren der gespeicherten Directory-Einträge.

**Authentication** das sind Operationen wie *bind*, *unbind*, Setzen von Benutzer-Zugriffsrechten usw.

**Suchen** In den Suchfunktionen können der Suchstartpunkt, die Suchtiefe, die Attribute, die bei der Suche verglichen werden und die Attribute, die für gefundene Einträge zurückgeliefert werden, angegeben werden. Beispiele: Suche nach der Postadresse vom *cn=Jorg Meier, o=Example, c=DE*

**Suchparameter** Bei der Suche werden folgende Parameter angegeben (oder mit Default-Werten belegt):

**Basis (engl. *base*)** Das ist ein DN, der bei der Suche in DIT als Ausgangspunkt benutzt wird.

**Bereich (engl. *scope*)** spezifiziert die Suchtiefe im DIT ausgehend von der Basis. Dieser Parameter kann drei Werte annehmen:

**base** nur das Basisobjekt wird durchsucht.

**one** nur der direkte Nachfolger der „Basis“ wird durchsucht, die Basis selbst aber nicht.

**sub** die Basis und alle nachfolgenden Objekte werden durchsucht.

**Suchfilter (engl. *Search Filter*)** spezifiziert die Suchkriterien und besteht aus logischen Verknüpfungen von Attributvergleichen.

**Attribute zurück (engl. *Attributes to Return*)** spezifiziert die Attribute der gefundenen Einträge, die als Ergebnis der Suche zurückgeliefert werden.

**Alias-Referenz auflösen (engl. *Alias Dereferencing*)** gibt an, ob der Alias oder der Eintrag, auf den er zeigt, benutzt werden. Ein aufgelöster Alias ist ein echter Ersatz für das ursprüngliche Objekt. Ohne Auflösung wird der Alias selbst untersucht.

**Einschränkungen (engl. *Limits*)** können die Anzahl der Ergebnisse oder die Suchzeit einschränken.

### 1.5.1 Suchfilter

Operator	Beschreibung	Beispiel
<i>Attr=Wert</i>	Liefert die Einträge, deren Attribut gleich dem Wert ist.	<code>cn=Jorg Mueller</code> findet den Eintrag mit <i>Common Name</i> Jorg Mueller
<i>Attr&gt;=Wert</i>	Sucht alle Einträge, deren Attribut größer oder gleich dem Wert ist.	<code>sn&gt;=meier</code> findet alle Einträge von <code>meier</code> bis <code>z*</code>
<i>Attr&lt;=Wert</i>	Sucht alle Einträge, deren Attribut kleiner oder gleich dem Wert ist.	<code>sn&lt;=meier</code> findet alle Einträge von <code>a*</code> bis <code>meier</code>
<i>Attr=*</i>	Liefert alle Einträge, die in dem Attribut einen Wert besitzen.	
<i>Attr~Wert</i>	Liefert alle Einträge, deren Attribut ähnlich dem Wert ist.	<code>sn =meier</code> findet <code>meier,maier,mayer</code> usw.

**Syntax der Suchfilter** Der Suchfilter definiert Kriterien für die Suche nach Directoryinträgen. Der Filter ist eine logische Verknüpfung von Attributvergleichen. Ein Attributvergleich hat folgende Form: *Attribut Operator Wert* z.B. im Filter `cn=Jorg Meier` (CommonName gleich Jorg Meier) ist `cn` der Attribut, `=` der Operator, `Jorg Meier` der Wert.

Das `*`-Zeichen kann jeden Substring ersetzen und wird für eine ungenaue Suche verwendet. Mit dem Filter `cn=J*M*` wird Jorg Meier, aber auch Jens Müller gefunden. Die Suchfilter können mit logischen Operatoren verknüpft werden, und damit kann eine komplexere Suche durchgeführt werden. Diese Verknüpfung sieht folgendermaßen aus:

- `( & (filter1) (filter2) ... )` UND-Verknüpfung der Filter.
- `( | (filter1) (filter2) ... )` ODER-Verknüpfung der Filter.
- `( ! (filter) )` diese Filter-Negation, kann sich nur auf einem Filter beziehen.



*Beispiel:*

```
( | (sn=Meier) (sn=Müller) )
```

liefert alle Personen mit Nachnamen Meier oder Müller. Die Filter können verschachtelt werden:

```
( | (cn=Meier) (& (ou=Finanzen) (sn=Müller)) )
```

Damit wird eine Person mit Nachname Meier oder Müller aus der Finanzabteilung gesucht.

### 1.5.2 Directory-Operationen

**Vergleichen** Mit der Operation „Vergleichen“ (engl. Compare) wird nur geprüft, ob die Suchkriterien erfüllt sind oder nicht. Das Ergebnis ist WAHR oder FALSCH.

**Update-Operationen** Die Update-Operationen ändern den Inhalt vom Directory. Die wichtigsten Operationen sind:

**add** fügt neue Einträge ein.

**delete** löscht Einträge. Es können nur Einträge gelöscht werden, Aliase werden dabei nicht aufgelöst.

**modify** ändert die Werte der Attribute von vorhandenen Einträgen. Es können neue Attribute hinzugefügt und existierende gelöscht werden.

**Authentizität** Die Authentizitätsoperationen werden bei dem Auf- und Abbau einer Session zwischen dem LDAP-Client und -Server benutzt. Die wichtigsten Operationen sind:

**Bind** baut eine Session auf. Es können User-Name und Paßwort überprüft werden. Wird auf das Directory ohne vorherigen bind-Aufruf zugegriffen, so gilt man beim Zugriff als Benutzer `anonymous`.

**Unbind** beendet eine Session.

Zusätzliche Sicherheit kann erreicht werden, wenn die Session verschlüsselt wird (z.B. mit SSL). SSL-Verschlüsselung wird aber erst von OpenLDAP Version 2.0 unterstützt.

### 1.6 Das Sicherheitsmodell

OpenLDAP implementiert Sicherheit auf mehreren Ebenen:

**Kerberos** ist ein am MIT entwickeltes System, das sowohl zur starken Authentifizierung als auch zur starken Verschlüsselung verwendet wird. Sicherheit wird hier durch die Generierung von Einmal-Paßwörtern erreicht. Durch Kerberos ist eine Single-Sign-On-Lösung für viele Dienste realisierbar. Kerberos ist jedoch sehr komplex und benötigt für seinen Betrieb wiederum einige Serverdienste, deswegen wird es in diesem Rahmen nicht ausführlicher behandelt.

**TLS (SSL)** Transport Layer Security (*TLS*) ist die Möglichkeit, die gesamte Kommunikation zwischen Client und Server zu verschlüsseln. Die Technik stammt von Netscape, hieß ehemals *SSL* (Secure Socket Layer) und arbeitet mit digitalen Zertifikaten nach X.509. Für den internen Gebrauch genügt es, ein selbstsigniertes Zertifikat zu erstellen. Auf keinen Fall sollen irgendwelche mitgelieferten Zertifikate verwendet werden, denn die privaten Schlüssel von diesen hat jeder und damit kann jeder die Kommunikation wieder mitlesen.

**SASL** ist die Simple Authentication and Security Layer des Cyrus-Projektes. Sie bietet starke Authentifizierung über einen einfachen Mechanismus. Die Administration ist denkbar einfach: es existiert eine Datenbank `/etc/sasl/db`, die mit dem Befehl `sasldblistusers` angezeigt werden kann. Eine SASL-Adresse

sieht so aus: *userRealm*. Realm ist englisch für Reich, ist die SASL-Domäne und wird im Falle eines zentralen SASL-Servers gleich dem Domänen-Namen sein. Ein SASL-Paßwort kann man mit dem Programm **saslpasswd** setzen. SASL hat sich mittlerweile wegen seiner Einfachheit recht gut durchgesetzt.

**Access Control Lists (ACLs)** sind die Möglichkeit, auf verschiedene Verzeichniseinträge Berechtigungen an verschiedene Benutzer des Verzeichnisses zu verteilen.

Alle Sicherheitskonfigurationen finden in der Hauptkonfigurationsdatei `slapd.conf` des OpenLDAP-Servers statt.

### 1.6.1 TLS-Konfiguration

Hier die wichtigsten Konfigurationsdirektiven für die TLS-Konfiguration:

**TLSCertificateFile** gibt an, in welcher Datei das Server-Zertifikat liegt. Es enthält unter anderem den öffentlichen Schlüssel des Servers.

**TLSCertificateKeyFile** gibt an, in welcher Datei der private Schlüssel des Servers liegt. Diese Datei darf kein Unbefugter lesen können!

### 1.6.2 SASL-Konfiguration

Von OpenLDAP 2.0 wird standardmäßig SASL verwendet. Mit der Konfigurationsdirektive `require SASL` wird SASL-Authentifizierung erzwungen. Weiterführende Direktiven finden sich in der Manpage von `slapd.conf`.

### 1.6.3 Access Control Lists

sind für einen gut konfigurierten Server absolut unabkömmlich. Im Folgenden wird das stark vereinfachte Prinzip erläutert.

#### Syntax:

```
access to Was
  by Wer Zugriff
  [ by Wer Zugriff ]
  ...
```


Auf *was* man Zugriff bekommt, kann man auf folgende Weise angeben:

**Angabe des DN** durch `dn=Regulärer Ausdruck`. Der Reguläre Ausdruck gilt über die *normalisierte Form* des DN, also, nachdem alle Leerzeichen entfernt wurden und Kommas nur zum Trennen der DN-Komponenten dienen.

**Angabe eines LDAP-Filters** mit der Syntax `filter=LDAP-Filter` bezeichnet man alle Einträge, auf die der Filter paßt.

**Angabe einer Attribut-Liste** regelt den Zugriff auf bestimmte Attribute wie folgt:

`attrs=Attribut1,Attribut2,...`

*Hinweis:* Zugriff auf das Attribut genügt hier nicht. Um Zugriff auf ein Attribut zu haben, braucht man auch Zugriff auf den Eintrag. Jedoch wird diese Methode verwendet, um bestimmte kritische Attribute wie etwa `userPassword` vor den Augen der Allgemeinheit zu verbergen. Mit `attrs=entry` kann man den Eintrag selbst definieren. 

\* steht für jeden Eintrag.

**Wer Zugriff bekommt** läßt sich wie folgt darstellen:

\* bezeichnet alle Benutzer, einschließlich der anonymen und angemeldeten Benutzer

**anonymous** bezeichnet den *nicht angemeldeten* Benutzer

**users** bezeichnet den *angemeldeten* Benutzer

**self** bezeichnet den mit dem Zieleintrag assoziierten Benutzer

**dn=Regulärer Ausdruck** bezeichnet alle Benutzer, deren DN auf den regulären Ausdruck paßt. Will man ein DN genau bezeichnen, verwende man `dn.exact=DN`.

**addr=Regulärer Ausdruck** bezeichnet die IP-Adresse des Clients

**domain=Regulärer Ausdruck** bezeichnet den FQDN des Clients

**dnattr=Attribut** bezeichnet den Benutzer, dessen DN im angegebenen Attribut steht.

*Beispiel:* Der Standardfall für die Angabe eines DN-wertigen Attributs ist das Attribut `owner`, das den DN des Eigentümers enthält. 

## 1.7 Das LDAP-Data-Interchange-Format (LDIF)

---

Welchen *Zugriff* jemand bekommt oder die Art des Zugriffs definiert man auf fünf Ebenen:

Zugriffsebene	Rechte	Beschreibung
none		Kein Zugriff
auth	=x	Anmeldung am Directory
compare	=cx	Vergleich
search	=scx	Anwendung von Suchfiltern
read	=rscx	Such-Resultate lesen
write	=wrscx	Einträge ändern/umbenennen

**Reihenfolge der Auswertung** Wenn **slapd** auswerten soll, ob der Anfragende Zugriff auf einen bestimmten Eintrag und/oder ein bestimmtes Attribut haben soll, so vergleicht **slapd** den Eintrag und/oder das Attribut mit den *Was*-Selektoren in der Konfigurationsdatei.

Die zur betreffenden Datenbank lokalen ACLs werden dabei zuerst überprüft, dann, falls keine lokale ACL zutrifft, gelten die globale ACLs. Unter Berücksichtigung dieser Priorität werden die ACLs an den entsprechenden Stellen nacheinander von oben nach unten überprüft. Beim ersten passenden *Was*-Selektor bleibt **slapd** stehen.

Dann geht **slapd** alle zu diesem *Was*-Selektor gehörigen *Wer*-Einträge von oben nach unten durch und bleibt wiederum beim ersten passenden stehen.

Schließlich vergleicht **slapd** den vom Client angeforderten Zugriff mit dem in *Zugriff* gegebenen Zugriff. Ist letzterer größer oder gleich dem angefragten Zugriff, so wird der Zugriff gewährt, ansonsten verweigert.

**Beispiele zu den Zugriffsverfahren** finden sich weiter unten bei der Erläuterung der Konfigurationsdatei des **slapd**.

## 1.7 Das LDAP-Data-Interchange-Format (LDIF)

Einzelne Operationen wie *add*, *delete* und *update* können mit Befehlen und Browser ausgeführt werden. Wenn aber viele Daten geändert, ein neues Directory erstellt oder auf einen anderen Server übertragen werden sollen, können die Einträge im *LDAP-Data-Interchange-Format (LDIF)* in einer Datei gespeichert werden. Die Einträge können dann mit einem Texteditor, mit Shell-Skripten oder anderen Tools modifiziert werden. Dateien im LDIF-Format können auch aus anderen Informations- (z.B. NIS) oder System-dateien (z.B. */etc/passwd*) erstellt werden. Danach ist es möglich, die Directory-Einträge oder einen ganzen DIT aus der LDIF-Datei automatisch zu erstellen.

**LDIF-Format** Die LDIF-Datei besteht aus einer Reihe von Records, getrennt von Leerzeilen und beschreibt entweder den Inhalt der Directory-Einträge oder deren Änderung. Die Zeilen eines Records beschreiben einen Directory-Eintrag oder Änderungen eines Eintrags. Generell sieht ein Record folgendermaßen aus:

**Syntax:**

```
[id]
dn: distinguished name
objectClass: object class
objectClass: object class
...
Typ des Attributs: Wert des Attributs
Typ des Attributs: Wert des Attributs
...
```

Obligatorisch ist nur die Zeile mit dem `dn` und mindestens eine Zeile mit `objectClass`. Die Zeilen haben folgende Bedeutung:

**id** eine optionale positive Dezimalzahl repräsentiert die Eintrags-ID.

**dn:** spezifiziert den DN vom Eintrag.

**objectClass:** spezifiziert die Schemata oder Typen von Attributen, die für diesen Eintrag erforderlich oder erlaubt sind. Der LDAP-Server kann so konfiguriert werden, daß er diese Angaben strikt überprüft.

## 1.8 Installation und Konfiguration des OpenLDAP-Servers

Die Installation vom LDAP-Server wird mit Hilfe des `rpm`-Programms (Red Hat Package Manager) durchgeführt und verläuft in folgenden Schritten:

- Zuers soll überprüft werden, ob das LDAP-Paket schon installiert ist:

```
# rpm -q openldap
```

- Wenn LDAP nicht installiert ist, kann es vom Paket `openldap-version.rpm` (RedHat) bzw. `openldap.rpm` installiert werden. Dort sind die binären Versionen vom LDAP-Server, von Hilfsprogrammen und Konfigurationen gepackt:

```
# rpm -ivvh openldap-version.rpm
```

### 1.8.1 Konfigurieren des LDAP-Servers

Der OpenLDAP-Server ist aus dem LDAP-Server **slapd** und dem LDAP-Replikationsdaemon **slurpd** der University of Michigan heraus entstanden.

Bevor der LDAP-Server gestartet werden kann, muß er konfiguriert werden. Alle Konfigurationsdateien der OpenLDAP-Server finden sich im Verzeichnis `/etc/openldap/`, bzw. `/usr/local/etc/openldap/`, falls Sie den OpenLDAP-Server Version 2.0 von Hand kompiliert und installiert haben. Im Text verwenden wir aber stets den ersten Pfad.

Die Hauptkonfigurationsdatei für **slapd** und **slurpd** ist `/etc/slapd.conf`.

Die Hauptkonfigurationsdatei hat folgende Struktur:

```
# Dies ist ein Kommentar
Globale Konfigurationsparameter

database Typ:Backend 1
Konfigurationsparameter für Backend 1

database Typ:Backend 2
Konfigurationsparameter für Backend 2

...
```

Kommentarzeichen ist, wie so oft unter UNIX/Linux, das Doppelkreuz #.

Die wichtigsten Parameter der Hauptkonfigurationsdatei sind:

**suffix** setzt das LDAP-Server-Suffix pro Datenbank fest.

**directory** das Verzeichnis, in dem die Directory-Datenbank-Dateien gespeichert werden. Sinnvollerweise nimmt man entsprechend dem Filesystem Hierarchy Standard (FHS) das Verzeichnis `/var/lib/ldap/`.

**rootdn, rootpw** das sind DN und Paßwort, die uneingeschränkten Zugriff zum Directory haben. Der Administrator kann ein „bind“ mit diesem Eintrag machen und dann weitere Einträge einfügen, ändern und löschen oder Zugriffsrechte verwalten.

**database** gibt den Typ des Backends an. LDAP ist ja nur ein Protokoll. Die Daten, die das Frontend LDAP liefert, können aus recht beliebigen Quellen, den Backends stammen. Das Standard-Backend `ldbm` ist die LDAP-native *LDBM*-Datenbank (*LDAP Database Manager*). Diese ist in den OpenLDAP-Server eingebaut und ist für die Verwaltung von LDAP-Daten optimiert. Alternativ lassen sich mit

## LDAP Directory Server

---

OpenLDAP 2.0 auch SQL-Datenbanken anbinden, oder mit dem C-Backends und dem Shell-Backend lassen sich per C-Programmen oder Shell-Skripten beliebige Datenquellen anbinden. Dafür muß man die wesentlichen Funktionen aus dem Funktionsmodell schreiben. Beispiele hierfür finden sich im „**slapd** and **slurpd** Administrators Guide“ der University of Michigan.

**referral** Gibt die LDAP-URL einer Rückreferenz an. Das bedeutet, wenn ein Client einen Eintrag auf diesem Server nicht findet, so gibt der Server diese Rückreferenz zurück.

**include** Bindet noch weitere Dateien in die Hauptkonfigurationsdatei mit ein. Die Standardverwendung dafür ist die Einbindung der Schemata, die man verwenden will. Dies sind die Dateien im Verzeichnis `/etc/openldap/schema/`. Mit `include` lassen sich bei der neuen Version die gewünschten Schemata einbinden. Folgende Schemata sind verfügbar:

Dateiname	Bedeutung
<code>core.schema</code>	OpenLDAP-Kern (notwendig)
<code>cosine.schema</code>	Modelliert einige Bedürfnisse einer Firma (Adressen/Dokumentenverwaltung) sowie DNS (nützlich)
<code>inetorgperson.schema</code>	Internet-Benutzer in einer Firma, braucht das <code>cosine.schema</code> (nützlich)
<code>misc.schema</code>	Experimentelles Mail-Routing
<code>nadf.schema</code>	Das Schema des North American Directory Forum (NADF) (nicht nützlich)
<code>nis.schema</code>	NIS durch LDAP ersetzen. Modelliert die wichtigsten UNIX-Konfigurationsdaten, wie Benutzeraccounts, Gruppen, Hosts, etc. (nützlich)
<code>openldap.schema</code>	OpenLDAP-Demo-Schema (experimentell)
<code>java.schema</code>	Zum Ablegen von Java-Objekten in LDAP (für Java-Programmierer nützlich)
<code>corba.schema</code>	Für die Ablage von CORBA-Objektreferenzen in LDAP. CORBA ist eine Programmierschnittstelle für verteilte Objekte. (Für Programmierer interessant)
<code>krb5-kdc.schema</code>	Für Kerberos-5 Key Distribution Center-Informationen (Für Kerberos-Administratoren interessant)
<code>microsoft*.schema</code>	Schemas für Microsofts Active Directory

**schemacheck** Kann die Werte `on` oder `off` annehmen. Ist die Schema-Überprüfung eingeschaltet, so nimmt überprüft der Server strikt die Objektdefinitionen und die Syntax der Attribute. Damit läßt sich struktureller Wildwuchs im Directory unterbinden, womit man die saubere Gliederung der Daten und damit die maschinelle Verwertbarkeit sicher stellt.

## 1.8 Installation und Konfiguration des OpenLDAP-Servers

---

Ist dagegen die Schema-Überprüfung abgeschaltet, so können beliebige Objekte und Attribute ins Directory eingefügt werden.

Eine vollständige Beschreibung der Konfigurationsparameter findet sich mit **man slapd.conf**



*Beispiel:* Hier ein (recht vollständiges) Beispiel einer Konfigurationsdatei für OpenLDAP 2.0:

```
1 # example config file - global configuration section
2 include /usr/local/etc/openldap/schema/core.schema
3 schemacheck on
4
5 referral ldap://dsa2.example.de
6 access to * by * read
7
8 # -----
9 # ldbm definition for the example.de
10 database ldbm
11 suffix "dc=example, dc=de"
12 directory /var/lib/openldap/example.de/
13 rootdn "cn=Manager, dc=example, dc=de"
14 rootpw desecret
15
16 # ldbm access control definitions
17 access to attr=userPassword
18     by self write
19     by anonymous auth
20     by dn="cn=Admin,dc=example,dc=de" write
21     by * none
22 access to *
23     by self write
24     by anonymous auth
25     by dn="cn=Admin,dc=example,dc=de" write
26     by * read
27 mode 0600
28 # -----
29 # ldbm definition for example.net
30 database ldbm
31 suffix "dc=example, dc=net"
32 directory /var/lib/openldap/example.net/
33 rootdn "cn=Manager, dc=example, dc=com"
34 rootpw netsecret
```

## LDAP Directory Server

---

35 access to \* by users read  
36 mode 0600

SOLUZIONE

**Zeile 1** 1 ist ein Beispiel-Kommentar

**Zeile 2** bindet das Core-Schema ein

**Zeile 3** aktiviert die Schema-Überprüfung. Alle Versuche, Daten ins Directory einzufügen, schlagen damit fehl.

**Zeile 5** definiert eine Rückreferenz. Alle Anfragen, die nicht auf diesem Server zu finden sind, werden mit einer Referenz auf den angegebenen Server beantwortet

**Zeile 6** ist die globale Zugriffsberechtigung. Sie greift *nur*, falls bei den einzelnen Datenbanken keine individuellen Zugriffsberechtigungen angegeben sind, sonst haben diese Vorrang.

**Zeile 10** läutet die erste Datenbank vom Typ LDBM ein.

**Zeile 11** legt den Suffix der in dieser Datenbank gespeicherten Objekte fest

**Zeile 12** legt fest, in welchem Verzeichnis die Datenbankdateien gespeichert werden. Sinnvollerweise wählt man bei mehreren Datenbanken ein Unterverzeichnis pro Datenbank

**Zeile 13** legt fest, welcher Eintrag der Datenbank sich als LDAP-Administrator anmelden darf. Dieser hat uneingeschränkte Rechte auf die Datenbank.

**Zeile 14** das Paßwort des Administrators. Auf Zugriffsrechte (0600) der `slapd.conf` achten!

**Zeile 17** definiert eine ACL (Access Control List) für den Zugriff auf das Attribut `userPassword`.

**Zeile 18** Wenn sich ein Benutzer am Directory anmeldet, hat er auf seinen eigenes Paßwort Schreib/Lesezugriff

**Zeile 19** Der anonyme Benutzer darf das Attribut nur zur Authentifizierung verwenden.

**Zeile 20** Einer der Administratoren mit dem DN `cn=Admin,dc=example,dc=de` darf daß Paßwort ebenfalls verändern.

**Zeile 21** Alle anderen haben keinen Zugriff auf das Paßwort.

**Zeile 22** Wir definieren eine ACL für alle Knoten und Attribute:

**Zeile 23** Der angemeldete Benutzer darf seinen eigenen DN uneingeschränkt verändern.

**Zeile 24** Ein anonymer Benutzer darf Attribute nur zur Anmeldung am Verzeichnis benutzen.

**Zeile 25** Einer der Administratoren mit dem DN `cn=Admin,dc=example,dc=de` darf alle Einträge verändern.

**Zeile 26** Alle anderen dürfen die Attribute lesen.

**Zeile 27** Die Datenbankdateien werden mit den UNIX-Zugriffsrechten `rw-----` versehen.

**Zeilen 30-36** definieren noch eine zweite Datenbank, die den Baum mit dem Suffix `dc=example,dc=net` darstellt.

**Starten und Stoppen vom slapd** Wenn der **slapd** ohne Parameter aufgerufen wird, startet er automatisch im Hintergrund, und wartet am Default-TCP-Port 389 auf LDAP-Anfragen. Der **slapd**-Dämon wird mit „kill -1“ gestoppt:

```
# kill -1 `cat /var/run/slapd.pid`
```

Der LDAP-Server kann auch im Debug-Modus aufgerufen werden:

```
# /usr/local/libexec/slapd -d 10
```

Damit arbeitet der **slapd**-Dämon nicht im Hintergrund, sondern zeigt Debug-Informationen an. Er kann mit `Strg-C` beendet werden. Die Ausgabe setzt nicht unbedingt Kenntnisse der Programmquelle vom **slapd** voraus. Aus manchen Ausgaben lassen sich nützliche Informationen über die Arbeitsweise gewinnen. Eine Liste der möglichen Debug-Level findet man in `man slapd.conf`.

## 1.9 LDAP-Clients und Hilfsprogramme

**LDAP Client-Befehle** Mit dem LDAP-Paket werden einige LDAP-Client-Programme geliefert, mit denen im LDAP-Server-Einträge gesucht, erstellt, gelöscht und geändert werden können.

**ldapsearch** Der **ldapsearch**-Befehl realisiert die LDAP-Suchfunktion mit all ihren Parametern.

### Syntax:

```
ldapsearch [Optionen] Filter [Attribute]
```

Die am meisten verwendeten Optionen sind:

---

- h gibt den Host-Namen vom LDAP-Server an.
- b Der Basisname im LDAP-DIT. Es wird ab diesem Punkt nach unten gesucht. Da der Default-Basisname schon beim Kompilieren in den LDAP-Programmen eingetragen ist, sollte -b fast immer angegeben werden.
- D **binddn** führt die bind-Operation mit dem binddn-DN aus.
- s bestimmt den Such-Scope (die Suchtiefe). Die drei Werte sind `base`, `one`, `sub`.
- x gibt an, daß die normale Authentifizierung (ohne SASL) verwendet wird. Wenn SASL nicht eingerichtet ist, sollten Sie diese Option unbedingt verwenden.
- v zeigt einige Diagnostikmeldungen an.

Seltener verwendete, aber sehr nützliche Optionen sind:

- L Die Ausgabe wird im LDIF-Format ausgegeben und kann z.B. in eine Datei umgeleitet werden. Auf diese Weise kann die LDAP-Datenbank in Textform gespeichert, modifiziert und wieder geladen werden.
- d Debug Level setzen. Man kann z.B. mit -d 2 anzeigen, wie die Anfrage genau abgesetzt wird. Die Ausgabe ist aber mehr für Leute, die sich in dem Source der LDAP-Programme auskennen, gedacht.

Die Filterangabe hat die Form *Attribute=Wert*.



*Beispiel:*

```
$ ldapsearch -x -b "o=Example,c=de" "cn=*jorg*"
```

Wenn der Filter `objectclass=*` benutzt wird, werden alle Einträge angezeigt, weil jeder Eintrag mindestens ein Attribut vom Typ `objectclass` besitzt. Es gibt zwei spezielle Such-Basis-Angaben, die die Konfigurations- oder Laufzeitinformationen des LDAP-Servers anzeigen:

- `ldapsearch -x -s base -b "cn=config" "objectclass=*"`  
zeigt die Konfiguration vom LDAP-Server an, unter anderem welche Suffixe der Server unterstützt.
- `ldapsearch -x -s base -b "cn=monitor" "objectclass=*"`  
zeigt einige Laufzeitstatistiken an.

Diese Befehle können als Test des LDAP-Servers benutzt werden.

**ldapmodify** und **ldapadd** Mit den **ldapmodify**- und **ldapadd**-Befehlen können die Werte der Eintragsattribute geändert werden.

### Syntax:

```
ldapmodify [Optionen] [-f Datei]
```

```
ldapadd [Optionen] [-f Datei]
```

Die Information, was hinzugefügt oder geändert werden soll, wird aus einer LDIF-Datei oder von der Standardeingabe eingelesen. Folgende Optionen werden oft verwendet:

**-r** sagt explizit, daß die Werte ersetzt und nicht hinzugefügt werden sollen.

**-a** sagt explizit, daß die Werte hinzugefügt und nicht ersetzt werden sollen.

**-D binddn** führt die bind-Operation mit dem binddn-DN aus.

**-w *Paßwort*** verwendet das Paßwort für die Anmeldung am LDAP-Server.

**-x** gibt an, daß die normale Authentifizierung (ohne SASL) verwendet wird. Wenn SASL nicht eingerichtet ist, sollten Sie diese Option unbedingt verwenden.

Die Optionen **-D** und **-w** entsprechen den Optionen `rootdn, rootpw` in der `slapd.conf`-Datei.

### Beispiel:

```
# ldapmodify -x -D "cn=root,o=Example,c=DE" -w secret -f change.ldif
```

die Datei `change.ldif` kann dabei etwa so aussehen:

```
dn: cn=root, o=Example, c=DE
  changetype: modify
  replace: mail
  mail: modme@terminator.rs.itd.umich.edu
  -
  add: title
  title: Grand Poobah
  -
  delete: description
  -
```

Mit `changetype` wird die Art der Änderung angegeben (hier: Modifikation). Dann kommt eine Serie von Änderungen. Zuerst die Art der Änderung, dann das Attribut.

- `replace` ersetzt ein Attribut
- `add` fügt ein neues Attribut hinzu
- `delete` löscht ein Attribut

Schließlich wird jede Änderung mit einem Bindestrich am Zeilenanfang terminiert.

**Der `ldapdelete`-Befehl** Mit dem `ldapdelete`-Befehle können ganze Einträge geändert werden:

**Syntax:**

```
ldapdelete [Optionen] DN1 DN2 ...
```

Die Optionen sind ähnlich wie bei dem `ldapmodify`-Befehl.

**Der `ldapmodrdn`-Befehl** Der `ldapmodrdn`-Befehl ändert den RDN eines Eintrags und hat folgende Syntax:

**Syntax:**

```
ldapmodrdn [Optionen] DN Neuer-RDN
```

oder

**Syntax:**

```
ldapmodrdn [Optionen] -f file
```

In der ersten Schreibweise wird der RDN-Name des DN-Eintrags auf dem Wert `Neuer-RDN` geändert.


In der zweiten Schreibweise wird erwartet, daß in der Datei in zwei Zeilen der DN und der neue RDN getrennt mit Leerzeile von den anderen Beschreibungen stehen. Die Optionen sind ähnlich wie bei dem `ldapmodify`-Befehl. Die `-r`-Optionen hat eine andere Bedeutung: Der Eintrag mit dem alten RDN soll gelöscht werden. Sonst bleibt er weiter bestehen.

**LDAP-Dienstprogramme** Für den Umgang mit dem LDAP-Server existieren einige Dienstprogramme, die einem die Administration erleichtern. Ein sehr nützliches ist das Programm **slapadd**. Damit können LDIF-Einträge in die LDAP-Datenbank übernommen werden:

```
# slapadd -v -l beispiel.ldif
```

Die Option `-v` erzeugt ausführliche (verbose) Rückmeldungen des Programmes **slapadd**. Mit `-l datei` wird **slapadd** angewiesen den LDIF-Eintrag nicht von der Standardeingabe einzulesen, sondern aus der angegebenen Datei.

Die Datenbank wird im konfigurierten Verzeichnis (Parameter `directory` in `slapd.conf`) in mehrere Dateien geschrieben. Um Dateninkonsistenzen vorzubeugen, sollten Sie **slapadd** nur dann ausführen, wenn der **slapd**-Dämon nicht geladen ist.

*Achtung:* Importiert man Daten mit **slapadd**, muß man selbst dafür sorgen, daß diese stimmen. Überprüfungen wie z.B. Schema-Check, werden nicht durchgeführt. Will man einer bereits bestehenden Datenbank neu Einträge hinzufügen, verwendet man besser **ldapadd**. 

**Der slapcat-Befehl** Mit dem **slapcat**-Befehl kann die Datenbank in LDIF-Form konvertiert und ausgegeben werden. Mit der Option `b` kann durch ein Suffix angegeben werden, welche Datenbank ausgegeben wird. Im folgenden Beispiel wird alles ausgegeben, statt auf die Standardausgabe aber in die Datei `backup.ldif` geschrieben:

```
# slapcat -l backup.ldif
```

So können alle Daten des LDAP-Servers in eine Datei gesichert werden.

### 1.10 LDAP-Replikation mit slapd und slurpd

Unter Replikation versteht man das Kopieren der Directory-Daten von einem Master-Server auf einen oder mehrere Slave-Server.

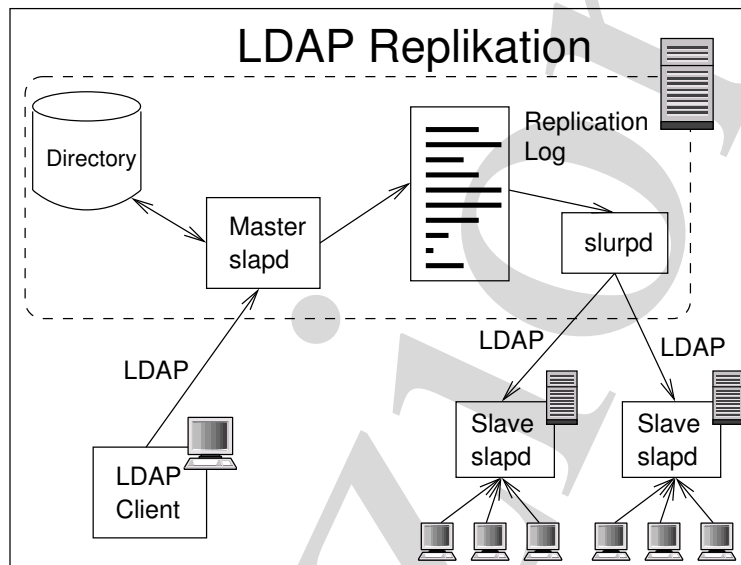
Die Vorteile von Replikation liegen auf der Hand:

- Die Last der Anfragen wird auf mehrere Server verteilt
- Falls einer der Server ausfällt, arbeiten die anderen weiter
- Falls das Replikat auf der anderen Seite einer langsamen Wahlverbindung liegt, spart man kostbare Bandbreite und erhöht die Antwortgeschwindigkeit drastisch

## 1.10 LDAP-Replikation mit slapd und slurpd

Der Nachteil, daß Änderungen weiterverteilt werden müssen, fällt dagegen nicht so sehr ins Gewicht, denn LDAP ist als Verzeichnisdienst sowieso hauptsächlich auf Lesezugriffe spezialisiert. Ferner riskiert man vom Zeitpunkt der Änderung bis zur Übertragung auf die Slave-Server, daß die Clients, die auf die Slave-Server zugreifen in diesem Zeitraum mit den alten Daten arbeiten.

In folgendem Diagramm ist die Funktionsweise der Replikation mit OpenLDAP dargestellt:



Ein Client ändert einen Eintrag im Directory. Der Master-**slapd** ändert den Eintrag im Directory und schreibt gleichzeitig einen Eintrag in die Replikations-Logdatei. Diese wiederum wird von **slurpd** ausgelesen, der die Änderungen an die Slave-**slapds** per LDAP weiterverteilt.



*Beispiel:* Eine Replikations-Logdatei kann wie folgt aussehen:

```
replica: slave.example.com:389
time: 809618633
dn: uid=bjensen, dc=example, dc=com
changetype: modify
replace: multiLineDescription
description: A dreamer...
-
replace: modifiersName
modifiersName: uid=bjensen, dc=example, dc=com
-
```

## LDAP Directory Server

---

```
replace: modifyTimestamp
modifyTimestamp: 20000805073308Z
-
```

Man erkennt den Slave-Rechner und dessen Port, den Zeitstempel der Änderung und schließlich eine Serie von Änderungen im LDIF-Format.

Das dargestellte Setup hat den Vorteil, daß die Last verteilt wird, fällt jedoch einer der Server aus, ist eine ganze Reihe von Clients nicht mehr fähig, auf ihren LDAP-Server zuzugreifen.

### 1.10.1 Replikation konfigurieren

Wir betrachten anhand eines Beispiels die Replikations-Konfiguration. Der Rechner `master.example.de` enthält das Directory, das auf den Rechner `slave.example.de` repliziert werden soll.

#### **slapd.conf auf dem Master-slapd**

---

```
...
database          ldbm
cachesize         0
suffix            "o=Example, c=DE"
directory
rootdn            "cn=Manager, o=Example, c=DE"
rootpw            secret
lastmod          on

relogfile        /var/lib/openldap/slapd.relog
replica          host=slave.example.de:389
                 binddn="cn=Replicator, o=Example, c=DE"
                 bindmethod=simple
                 credentials=secret
```

---

Die Besonderheit in dieser Konfigurationsdatei sind die `replica-` und `relogfile-` Direktiven. Sie werden vom **slurpd** gelesen. `relogfile` gibt an, in welcher Datei das Replikations-Logbuch vom **slapd** geschrieben und vom **slurpd** gelesen wird. Die `replica`-Direktive sagt dem **slurpd**, auf welchem Host er replizieren soll, und mit welchem DN er sich an diesen binden soll.

#### **slapd.conf auf dem Slave-slapd**

---

```
...
referral          "ldap://master.example.de:389/" # optional

database           ldbm
cachesize         0
suffix             "o=Example, c=DE"
directory        /var/lib/openldap/example_replica/
rootdn           "cn=Replicator, o=Example, c=DE"
rootpw           secret
updatedn         "cn=Replicator, o=Example, c=DE"
updateref        "ldap://master.example.de:389/o=Example, c=DE"
lastmod            on
```

---

Die Referenz auf den Master ist optional, und nur dann notwendig, wenn der Master mehr Daten hält als der Slave, bzw. wenn zu befürchten ist, daß die Synchronisation ausfällt. Als Root-DN wird der DN angegeben, der auf dem Master in der `replica`-Direktive zu finden ist. `updateref` ist die Referenz, die der Client erhält, wenn er versucht, einen Datensatz zu ändern. Die Cache-Größe wurde auf Null reduziert, weil der Slave gar nicht nachsieht, ob sich der Datensatz geändert hat, wenn er im Cache ist.

Nun wollen wir im Folgenden die Vorgehensweise bei der Konfiguration von Master und Slave eingehen:



### **Master und Slave für Replikation konfigurieren**

1. Zuerst wird der Master entsprechend dem obigen Beispiel konfiguriert. Die `repllogfile`- und `replica`-Direktiven müssen hinzugefügt werden. Der `binddn`-Parameter muß mit dem `updatedn` des Slave übereinstimmen.
2. Dann wird der Slave konfiguriert. Dabei sollen keine `replica`/`repllogfile`-Direktiven enthalten. Es ist zwar prinzipiell möglich, Replikationsketten zu erzeugen, aber meistens ist es unpassend. Man ergänzt die `updatedn`-Zeile, die dem Slave sagt, wer berechtigt ist, Updates machen. Diese muß mit dem `binddn` des Master übereinstimmen. Es muß sichergestellt sein, daß dieser DN auf diesem Server Schreibrecht hat. Mit der `updateref`-Direktive wird schließlich noch der Master-Server angegeben, auf dem ja die tatsächlichen Änderungen stattfinden sollen.
3. Beenden Sie den Master-**slapd**

4. Kopieren Sie alle Dateien aus der Master-Datenbank in das Datenbank-Verzeichnis des Slave-**slapd**.
5. Starten Sie den Master-**slapd**
6. Starten Sie den Slave-**slapd**. Testen Sie gegebenenfalls, ob die Erzeugung der Replikations-Logdatei funktioniert hat, indem Sie einen Eintrag ändern.
7. Starten Sie **slurpd** auf dem Master mit

```
# slurpd -f /etc/openldap/slapd.conf
```

### 1.10.2 Umgang mit Replikationsfehlern

Wenn **slurpd** eine Änderung abschickt, die einen Fehler beim Slave-**slapd** auslöst, so wird dieser Datensatz in eine *Reject-Datei* geschrieben. Wenn beispielsweise auf dem Rechner `slave.example.de` (Port 389) der Fehler auftritt, heißt die Reject-Datei `repllog.slave.example.com:389` und steht im in der `directory`-Direktive angegebenen Verzeichnis des Masters, falls Sie existiert.

Ihr Inhalt kann zum Beispiel so aussehen:

```
ERROR: No such attribute
replica: slave.example.de:389
time: 109618733
dn: uid=nroux, dc=example, dc=de
changetype: modify
replace: description
description: Unsere Beste Kraft...
-
replace: modifiersName
modifiersName: uid=nroux, dc=example, dc=de
-
replace: modifyTimestamp
modifyTimestamp: 20001105073308Z
-
```

Das ist genau das Format des Original-Replikations-Log-Eintrages, nur mit der Fehlermeldung vorangestellt.

**One-Shot-Modus des slurpd** Normalerweise läuft **slurpd** ständig, und propagiert alle Änderungen, sobald sie eingetreten sind. Jedoch ist auch ein zweites Vorgehen möglich: **slurpd** läuft nicht, und man arbeitet die Replikations-Logdatei von Zeit zu Zeit in einem Aufwasch ab, was aber normalerweise eher unpassend ist.

## 1.10 LDAP-Replikation mit `slapd` und `slurpd`

---

Am besten eignet sich dieser Modus zur Abarbeitung von Reject-Dateien:

```
# slurpd -o -r /var/ldap/repllog/repllog.slave.example.de:389
```

Die ERROR-Meldungen werden von **slurpd** ignoriert, man muß sie also nicht erst entfernen.

**1.11 Wissensfragen**

1. In welcher Beziehung stehen Objekte, Attribute und Werte in einem Directory?

---

---

---

---

---

---

---

2. Was ist die Syntax eines Attributs?

---

---

---

---

3. Was sind Objekt-Schemata?

---

---

---

---

---

---

---

4. Was ist der Directory Information Tree (DIT)?

---

---

---

---

5. Was ist DN?

---

---

---

---

6. Welcher Unterschied besteht zwischen DN und RDN?

---

---

---

---

7. Wo werden die RDNs ou=Marketing, c=DE, cn=Michael Wagner, o=BMW im DIT eingeordnet und welchen DN bilden sie?

---

---

---

---

---

8. Was ist das Directory-Suffix?

---

---

---

9. Was ist eine Referenz im DIT?

---

---

---

10. Wer kann einer Referenz folgen und wie wird das festgelegt?

---

---

---

11. Was bedeuten der Scope- und der Base-Suchparameter?

---

---

---

12. Wie können alle Einträge im DIT angezeigt werden?

---

---

---

13. Wie werden Filter negiert?

---

---

## LDAP Directory Server

---

14. Welche Rolle spielt der LDIF-Format in den Directories?

---

---

---

15. In welcher Datei wird der LDAP-Dämon **slapd** konfiguriert?

---

16. Mit welchem Programm unter UNIX werden LDAP-Einträge gesucht?

---

17. Wie wird aus einer LDIF-Datei ein LDAP-Directory erstellt?

---

---

---

---

18. Wie kann aus dem LDAP-Directory eine LDIF-Datei erstellt werden?

---

---

---

---

19. Wie sieht ein LDAP-URL aus, mit der alle Einträge im DIT angezeigt werden?

---

20. Was ist ein UFN?

---

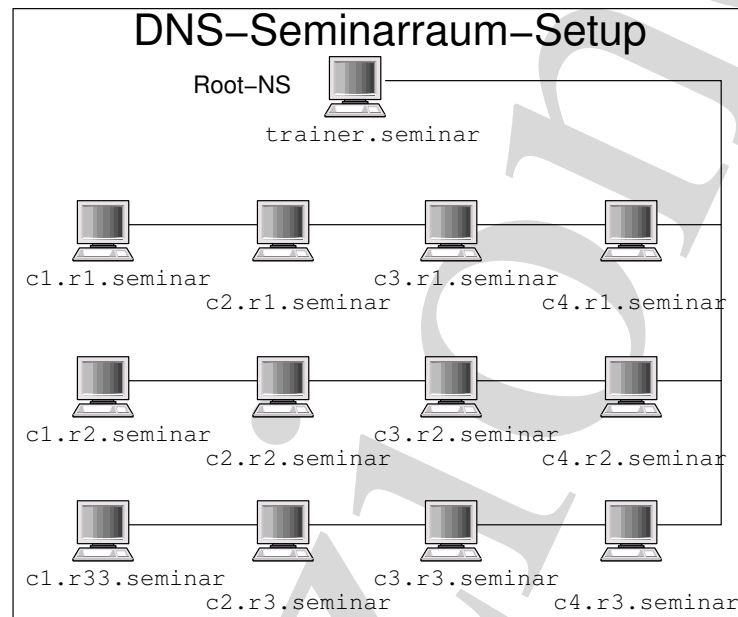
---

21. Welche Operation muß ein Programm durchführen, bevor es mit dem Directory arbeiten kann?

---

## 1.12 Übungen

1. Konfigurieren Sie Ihre Rechner so, daß sie den Trainer-Rechner als Nameserver verwenden. Gehen Sie sicher, daß auf dem Trainer-Rechner der DNS-Server folgendes Klassenraumsetup verwendet:



Prüfen Sie auch, ob die übrige Netzwerkkonfiguration (Hostname, IP-Adressen, /etc/hosts,...) vollständig funktioniert. Das ist eine wesentliche Voraussetzung für die kommenden Aufgaben.

2. Konfigurieren Sie den LDAP-Server. Die LDAP-Server-Struktur soll der DNS-Struktur entsprechen, daß heißt, auf dem Trainer-Rechner wird ein LDAP-Server für den Root-DN `dc=seminar` konfiguriert, entsprechend für die einzelnen Reihen LDAP-Server mit den Root-DNs `dc=rN,dc=seminar`, wobei  $n$  die Reihennummer ist. Jeder Server in einer Teilreihe soll eine Rückreferenz auf den Trainer-LDAP-Server aufweisen. Schema-Überprüfung soll aktiviert sein.
3. Starten Sie Ihren LDAP-Server und überprüfen Sie mit einem LDAP-Client dessen Funktionsfähigkeit!
4. *Optional:* Konfigurieren Sie auf Port 1389 einen Slave-LDAP-Server, der das Datenbankverzeichnis `/var/lib/ldap/slave/` hat und per **slurpd** die Daten des Hauptdirectories bekommt. Dazu brauchen Sie eine modifizierte `slapd.conf`, die wir `slave.slapd.conf` nennen, deren Name Sie dem Slave-**slapd** mit dem Argument `-f` übergeben.

### 1.13 Lösungen

```
1. # nslookup trainer.seminar
   IP.Addr.Trainer.Rechner
# vi /etc/hosts
```

---

```
...
nameserver IP.Addr.Trainer.Rechner
...
```

---

Ansonsten `/etc/hosts` auf Konsistenz überprüfen, die Ausgabe von `ifconfig` und `route -n` überprüfen.

#### 2. Hauptkonfigurationsdatei `/etc/openldap/slapd.conf`:

```
include          /etc/openldap/schema/core.schema
include          /etc/openldap/schema/cosine.schema
include          /etc/openldap/schema/inetorgperson.schema
include          /etc/openldap/schema/nis.schema
include          /etc/openldap/schema/kerberosobject.schema

schemacheck     on

pidfile          /var/run/slapd.pid
argsfile         /var/run/slapd.args

referral ldap://trainer.seminar

# Unsere Datenbank
database         ldbm
suffix           "dc=rN, dc=seminar"
rootdn           "uid=root,ou=People, dc=rN, dc=seminar"
directory        /var/lib/ldap
# Indizes
index objectClass eq
index uid         eq
# Zugriffskontrolle
access to attr=userPassword
                by self write
                by anonymous auth
                by dn.exact="ou=People,dc=rN,dc=seminar" write
                by * none
```

---

```
access to *
    by self write
    by anonymous auth
    by dn.exact="ou=People,dc=rN,dc=seminar" write
    by * read
```

3.

```
# /etc/init.d/ldap start
# ldapsearch -x -W -D "ou=People,dc=rN,dc=seminar" \
> -b "dc=rN,dc=seminar" '(uid=root)'
```

4. Optional, siehe Beschreibung im Kapitel.

### 1.14 Querverweise

1. Mehr Informationen zu OIDs, siehe:  
<http://www.alvestrand.no/harald/objectid/>
2. OID-Stamm-Anforderung bei der IANA kostenlos unter:  
<http://www.iana.org/cgi-bin/enterprise.pl>
3. Die aktuellsten Quellen von Cyrus-SASL finden sich als Tarball unter  
<ftp://ftp.andrew.cmu.edu/pub/cyrus-mail/>
4. Die Heimatseite von OpenLDAP, wo es immer die neueste Version gibt:  
[www.openldap.org](http://www.openldap.org)
5. Die LDAP-Migrationstools, `pam_ldap` und `nss_ldap` in das NIS-Schema finden sich unter <http://www.padl.com/software.html>
6. Ein exzellenter LDAP-Browser, komplett in JAVA geschrieben und so unter Linux wie Windows lauffähig: <http://www-unix.mcs.anl.gov/~gawor/ldap/>
7. Die Original Netscape-Anleitung zum Aufsetzen von HTTP- und LDAP-Roaming-Servern:  
[http://help.netscape.com/products/client/communicator/manual\\_roaming2.html](http://help.netscape.com/products/client/communicator/manual_roaming2.html)
8. Bezugsquelle LDAP-HOWTO: <http://www.linuxdoc.org>
9. Eine exzellente Web-Seite über LDAP und Verzeichnisdienste:  
<http://www.verzeichnisdienst.de>

---

## 2 Object Identifier, OID

In diesem Kapitel lernen Sie:

- OIDs kennen
- eigene Schemata zu erstellen

Jede Eigenschaft und jede Objektklasse, die im LDAP-Baum verwendet wird, besitzt eine eindeutige Bezeichnung, die *OID Object Identifier*.

Sie kommt niemals doppelt vor. OID's sind sehr gebräuchlich im SNMP-Protokoll, sind jedoch auch fester Bestandteil im LDAP-Protokoll. Die formelle Definition einer OID kommt vom ITU als Empfehlung (ASN.1).

OID's sind hierarchisch organisiert und sind dazu geeignet, hierarchische Strukturen numerisch abzubilden. Die OID's werden z.B. bei der IANA ([www.iana.org](http://www.iana.org)) für das Internet kostenlos registriert. Einen Überblick über alle wichtigen OIDs erhält man unter:

<http://www.alvestrand.no/objectid/top.html>

Die Nummern sind natürlich Organisationen/Personen zugeordnet. So kann man unterhalb der eigenen OID, selber den Namensraum bestimmen. Die grundlegenden OID's sind bekannt.



*Beispiel:* Die OID 1.3.6.1 steht für: iso(1) org(3) dod(6) iana(1)

Dieses Beispiel hat übrigens auch Praxisbezug, da alle internet-registrierten OID mit 1.3.6.1 beginnen.



*Hinweis:* OID's sind organisationsspezifisch bzw. firmenspezifisch. OID's die im Internet gültig sind, werden von der IANA vergeben und beginnen alle mit 1.3.6.1. Viele wichtige Produkte, auch Open Source-Produkte, haben eigene OID's, die man sich mit entsprechenden Objektklassen und Eigenschaften als Schema herunterladen kann (z.B. Für Samba bei [www.samba.org](http://www.samba.org)).

### 2.1 Eigene OID

Will man eigene Attribute bzw. Objektklassen für LDAP definieren, so sollte man seine Firmen-OID bei der IANA registrieren lassen:

<http://www.iana.org/cgi-bin/enterprise.pl>.

So hat z.B. SuSE die OID 1.3.6.1.1057

## Object Identifier, OID

---

Wenn man nun eine eigene Firmen-OID besitzt, so man kann ähnlich einer DNS-Domain, alles darunter liegende frei bestimmen. Man geht dabei in der Regel wie folgt vor:

Zunächst legt man eine eigene Hierarchie an:

*Beispiel:* Angenommen Die eigene Organisation besitzt folgende OID:

1.3.6.1.333333.

Dann bildet man eine grundlegende Hierarchie:

Organization's OID	1.3.6.1.333333
SNMP Elements	1.3.6.1.333333.1
LDAP Elements	1.3.6.1.333333.2
LDAP-AttributeTypes	1.3.6.1.333333.2.1
LDAP-ObjectClasses	1.3.6.1.333333.2.2

Darunter können dann z.B. eigene Eigenschaften und Klassen angelegt werden:

LDAP-erstes-myAttribute	1.3.6.1.333333.2.1.1
LDAP-erste-myObjectClass	1.3.6.1.333333.2.2.1


## 2.2 Schemata

Objektklassen und Attribute für einen bestimmten Themenbereich werden in einem sogenannten Schema zusammengefasst. Man lädt dieses Schema nur dann, wenn man die entsprechenden Eigenschaften und Objektklassen benötigt.

Bei Objektklassen unterscheidet man zwischen „Distributed (veröffentlichten) Schemata“ und „Eigene Schemata“ (die man selbst erstellt hat).

Distributed Schemata werden mit dem Produkt mitgeliefert. Hier die wichtigsten Distributed Schema für OpenLDAP:

- `core.schema`: OpenLDAP-Hauptschema (sollte immer geladen werden).
- `cosine.schema`: Internet-X.500-Schema.
- `inetorgperson.schema`: InetOrgPerson (Adressbücher).
- `misc.schema`: diverser (Mailinginfos).
- `nis.schema`: Network Information Services (Unix-Accounts).
- `openldap.schema`: OpenLDAP Project (experimental).

*Hinweis:* Es gibt auch Schemata von anderen Projekten oder Produkten. Diese werden vom jeweiligen Anbieter gestellt, z.B. das `samba.schema` vom SAMBA-Projekt. Es liegt dem Samba-Paket bei und dient der Benutzerverwaltung von SAMBA, die auf Wunsch von LDAP übernommen wird. 

**Erweitern der Schemata** Will man eigene Schemata erstellen, so muß man wie folgt vorgehen:



### **Eigenes Schema erstellen**

1. Von der IANA eine eigenen OID erlangen.
2. Erarbeiten des OID-Prefix für Eigenschaften und Objektklassen.
3. Erstellen von eigenen Attributen.
4. Zusammenfassen der Attribute zu Klassen.
5. Ablegen von Attributen und Klassen in einer Schema-Datei.



*Beispiel:* Hier ein einfaches Beispiel für eine Eigenschaft und eine Klasse:

```
attributetype ( 1.3.6.1.1.2.1.1
    NAME 'Sozialversicherungsnummer'
    DESC 'Sozialversicherungsnummer'
    EQUALITY caseIgnoreMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
    SINGLE-VALUE )
objectclass ( 1.3.6.1.1.2.2.1
    NAME 'Verwaltung'
    DESC 'Wichtige Infos der internen Verwaltung'
    SUP organizationalPerson
    STRUCTURAL
    MAY Sozialversicherungsnummer
    )
```

SOLUZIONE

### 2.3 Übungen

1. Erstellen Sie für ihre Firma „pinguine“ mit Sitz in Deutschland und einer Zweigstelle in Hamburg eine Namensbaum (tradionell und internettechnisch).
2. Geben Sie einen ldap-URL für den `uid=user1,ou=people,dc=pinguine,dc=de` auf dem Rechner `ldap.tux.de` an.
3. Erstellen Sie einen URL, der alle `objectClass=posixAccount` unterhalb von `ou=people,dc=pinguine,dc=de` auf dem Rechner `ldap.tux.de` sucht.
4. Sie haben Sich bei der IANA registriert und haben die OID `1.3.6.1.1` bekommen. Legen Sie hierzu eine hierarchische Struktur an, um später die SNMP und LDAP -OID zu vergeben.
5. Legen Sie entsprechend ihrer obigen OID, zwei OID für Attribute und ein OID für die erste Objektklasse fest.
6. Erstellen Sie ein eigenes Schema, das folgende Vorgaben hat:
  - Attribute unter: `1.3.6.1.1.2.1`
  - Objektklassen unter: `1.3.6.1.1.2.2`
  - Zwei Eigenschaften:
    - `InternalTelephoneNumber`
    - `InternalEmailAddress`
  - Eine Objektklasse: `InternalData` (Mögliche Eigenschaften z.B.):
    - `InternalTelephoneNumber`
    - `InternalEmailAddress`

### 2.4 Lösungen

#### 1. Tradionell:

```
c=DE
o=pinguine
ou=Hamburg
```

#### Internetaufbau:

```
dc=de
dc=pinguine
ou=Hamburg
```

- ldap://ldap.tux.de/uid=user1,ou=people,dc=pinguine,dc=de
- ldap://ldap.tux.de/ou=people,dc=pinguine,dc=de?posixAccount?sub?

#### 4. OID's:

Organization's OID	1.3.6.1.1
SNMP Elements	1.3.6.1.1.1
LDAP Elements	1.3.6.1.1.2
LDAP-AttributeTypes	1.3.6.1.1.2.1
LDAP-ObjectClasses	1.3.6.1.1.2.2

#### 5. OID's:

LDAP-erstes-myAttribute	1.3.6.1.1.2.1.1
LDAP-zweites-myAttribute	1.3.6.1.1.2.1.2
LDAP-erste-myObjectClass	1.3.6.1.1.2.2.1

- attributetype ( 1.3.6.1.1.2.1.1  
DESC 'Kurze ISDN-Nummer intern'  
NAME 'InternalTelephoneNumber'  
EQUALITY telephoneNumberMatch  
SUBSTR telephoneNumberSubstringsMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.50 )  
attributetype ( 1.3.6.1.1.2.1.2  
NAME 'InternalEmailAddress'  
DESC 'Interne Emailadresse'  
EQUALITY caseIgnoreIA5Match  
SUBSTR caseIgnoreIA5SubstringsMatch  
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26256 )  
objectclass ( 1.3.6.1.1.2.1.2.1  
NAME 'InternalData'

```
DESC 'Informationen zur Internen Kommunikation'  
STRUCTURAL  
MAY ( InternalEmailAddress $ InternalTelephoneNumber ) )
```

SOLUZIONE

---

## 3 Erweiterte Serverkonfiguration

In diesem Kapitel lernen Sie:

- das Erstellen von Indexdaten zum schnelleren Auffinden von Datensätzen
- ein sauberes Logging um Fehler auffindig zu machen
- das Limitieren von Zugriffen
- die Optimierung des Backendes, (in der Regel der Datenbank)

Um einen Server im Produktivsystem einzusetzen, sollten einige Einstellungen aktiviert und optimiert werden, auf die wir im folgendem näher eingehen.

### 3.1 Indizes

Indexdateien sind Auszüge aus der originalen Datenbank (bdb oder lbdm). Man kann sie sich wie den Index in einem Buch vorstellen. Es sind nur bestimmte Werte eingetragen, und diese zeigen auf den originalen Datensatz.

Wenn man eine bestimmte Eigenschaft, z.B. UID sucht, wird man diese in einem Index mit allen UID's schneller finden, als in der originalen Datenbank und dann nur noch auf diesen Datensatz zugreifen.

Man sollte Indizes nur für wichtige Attribute oder Objektklassen anlegen. Zu viele Indizes verbrauchen mehr Arbeitsspeicher und verlangsamen dadurch das Aktualisieren der Datenbank.

**Die Indextdirektive:** Indizes werden mit der Direktive `index` festgelegt:

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

Der Index kann auf verschiedene Vergleichsarten optimiert werden. Standardmäßig wird auf `pres` und `eq` optimiert. Folgende Indexarten sind vorhanden:

**eq** : Equality, Gleichheit.

**pres** : Present, vorhanden.

**approx** : Approximate, ungefähr.

**sub** : Substring, Teilzeichenkette.

## Erweiterte Serverkonfiguration

---

**none** : Keine Vergleichsart vorgegeben.

Empfehlenswert ist es, mindestens einen Index über die ObjectClass zu haben. Man konfiguriert Indizes in der `slapd.conf`.

*Beispiel:*

```
index      objectClass  eq
index      cn,sn,mail    pres,eq,approx,sub
```

Der Index muß mit dem Programm **slapindex** angelegt werden. Dabei werden alle Indizes aus den bestehenden Datenbanken neu generiert. Das Programm **slapindex** liest die Konfigurationsdatei aus und erzeugt im dynamischen Verzeichnis unter `/var/.../ldap` aktualisierte Indexdateien.

Der Befehl wird einfach in einer Shell ausgeführt und kommt im einfachsten Fall ohne Optionen aus.

*Hinweis:* Soll ein neuer Index erstellt werden, muß dieser erst mit **slapindex** erstellt und initialisiert werden.

### 3.2 Logging

Überwachung ist eine der wichtigsten Aufgaben eines Produktivsystems. Dies geschieht unter anderem mit Hilfe der Meldungen im Logging. Der OpenLDAP-Server hat, wenn er mit `--enable-debug` kompiliert wurde, ein fein einstellbares Logging.

Er gibt seine Logging-Protokolle an den Syslogd weiter. Dieser speichert sie entsprechend seiner Konfiguration z. B. In `/var/log/messages`. Das Logging wird über die Direktive `loglevel` gesteuert:

```
loglevel <integer>
```

Es existiert eine Vielzahl von Logleveln, die sich additiv zusammensetzen lassen. Das Default-Loglevel ist 256.

### Loglevel:

- 1 enable all debugging
- 0 no debugging
- 1 trace function calls
- 2 debug packet handling
- 4 heavy trace debugging
- 8 connection management
- 16 print out packets sent and received
- 32 search filter processing
- 64 configuration file processing
- 128 access control list processing
- 256 stats log connections/operations/results
- 512 stats log entries sent
- 1024 print communication with shell backends
- 2048 print entry parsing debugging



*Beispiel:* Zugriffskontrolle (128) und Status-Logging (256) erhält man demzufolge als  $128+256 = 384$ . Für das Loglevel gibt man also folgende Direktive an:

```
loglevel 384
```

### 3.3 Limit

Es macht keinen Sinn, alle Benutzer beliebig viele Einträge, beliebig lange suchen zu lassen. Das würde den LDAP-Server unnötig unter Last setzen.

Mit dem Setzen von Limitierungen hat man die Möglichkeit, Zugriffe zu beschränken.

- Zum einen mit `sizelimit`; das ist die maximale Anzahl an zurückgegebenen Einträgen bei einer Suche,
- zum anderen mit `timelimit`; die maximale Zeit in Sekunden die für eine Suche genutzt wird.

#### Direktiven:

**sizelimit** <integer>: Default-Wert ist 500

**timelimit** <integer>: Default Wert ist 3600

**Benutzerspezifische Limits** Natürlich sind solche Default-Optionen immer schwierig, denn sie gelten für alle Benutzer. Das heißt, auch ein Administrator oder der Replikationsdienst können z.B. nur 500 Einträge lesen. Um dies zu vermeiden, gibt es noch die Möglichkeit benutzerspezifische Limits zu definieren.

## Erweiterte Serverkonfiguration

---

Die zugehörige Direktive hat folgende Form:

```
limits <who> <limit> [<limit> [...]]
```

Der <who>-Teil ist wie folgt aufgebaut:

```
anonymous | users | [dn[.<style>]]<pattern>
```

Wobei <style> einen der folgenden Werte annimmt:

- exact
- base
- one
- subtree
- children
- regex

*Beispiel:* Einige Beispiele:

- `limits dn.sub=ou=admins,dc=example,dc=de size=10000 time=36000`
- `limits anonymous size=100 time=600`
- `limits users size=500 time=3600`

### 3.4 Backend-Konfiguration

Es gibt eine Vielzahl von möglichen Backends: Datenbanken (BDB, LDBM oder SQL), LDAP (wenn dies ein Proxy ist), die Shell, Meta,- oder Passwortdateien. Das Backend wird durch die Direktive `backend` definiert:

```
backend <databasetype>
```

Folgende Backends existieren:

- bdb
- dnssrv
- ldap
- ldbm

- meta
- monitor
- null
- passwd
- perl
- shell
- sql
- tcl

Im Folgenden wird der Umgang mit den zwei wichtigsten Datenbankformaten BDB und LDBM beschrieben.

**BDB-Datenbanken** Die Berkely Datenbank ist sehr verbreitet. Sie ist das empfohlene Datenbankformat. Sie nutzt Indizes und Caching sehr ausgiebig. Damit wird der Zugriff auf Daten erheblich beschleunigt.

Wichtige BDB spezifische Parameter:

**cache size** <integer> Wie viele Einträge sollen über BDB-Instanzen im Cache gehalten werden? Der Default-Wert ist 1000 Einträge.

**dbnosync** Soll die Datenbank sofort synchronisiert werden? Sofortige Synchronisation erhöht die Sicherheit, dies aber leider auf Kosten der Geschwindigkeit.

**mode** <integer> Gibt den Modus (Rechte) von neuen Datenbanken an. Der Default-Wert ist 0600.

**directory** <directory> Gibt das Verzeichnis an, in dem die Datenbank gespeichert werden soll.

☞ *Hinweis:* Wenn Sie große LDAP-Datenbanken verwalten, sollte die Cachesize sehr groß gewählt werden. Dafür benötigt der LDAP-Server allerdings ausreichend Speicherplatz.

## Erweiterte Serverkonfiguration

---

**LDBM** Das LDBM-Backend hat die Möglichkeit auf unterschiedliche Datenbankformate zuzugreifen: BerkeleyDB, GNU DBM, MDBM oder NDBM. Es nutzt Indizes und Caching ebenfalls sehr stark. Natürlich verwendet es dieselben Direktiven wie BDB, verfügt jedoch über einige Extraeinstellungen, wie z.B. das Feintuning des Synchronisierens auf die Platte:

```
dbsync <frequency> <maxdelays> <delayinterval>  
database <database>
```

Verwaltet ein OpenLDAP-Server mehrere Datenbanken mit unterschiedlichen Suffixen (Bäumen) gleichzeitig, kann für jede Datenbank das Backend gesondert festgelegt werden. Dazu leitet jeder database-Eintrag einen neuen Bereich ein.

### 3.5 Übungen

1. Erstellen Sie einen Index über die Objektklassen mit equality und einen Index für die Attribute cn, sn, uid mit equality, present, substring und approximate.
2. Erweitern Sie ihr Logging zum Testen mit gesendeten/empfangen Paketen, Suchfiltern, Zugriffskontrolle und Statuslogging.
3. Setzen Sie das Zeit- und Größenlimit für einen Benutzer ihrer Wahl auf das Zehnfache vom Default-Wert.
4. Erhöhen Sie das Caching für ihre Datenbank auf zehntausend Einträge (nur falls genügend Arbeitsspeicher vorhanden ist!).

### 3.6 Lösungen

1. Tragen Sie in `slapd.conf` folgendes ein:

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

Initialisieren Sie die Änderungen dann mit dem Befehl **slapindex**.

2. Suchen Sie in der Tabelle nach den Zahlenwerten für die entsprechenden Logging-Informationen. Addieren Sie diese:

$16 + 32 + 128 + 256 = 432$

Tragen Sie das Ergebnis in `slapd.conf` ein:

```
loglevel 432
```

3. Definieren Sie das Limit in `slapd.conf` (eine Zeile):

```
limits dn.exact=uid=admin,ou=people,dc=example,dc=de
       size=10000 time=36000
```

4. Tragen Sie in `slapd.conf` die neue Cache-Größe ein:

```
cachesize 10000
```

---

## 4 Grafische Frontends

In diesem Kapitel lernen Sie:

- LDAP mit grafischen Frontends zu bedienen

Eigentlich macht ein Verzeichnisdienst erst so richtig Spaß, wenn man ein grafisches Frontend hat. Natürlich gibt es hier eine Vielzahl von freien und kommerziellen LDAP-Browsern, die sich zur Administration eignen.

Die meisten Internetbrowser ermöglichen auch den Zugriff auf einen LDAP-Baum (z.B. Netscape, Outlook Express). Wir werden zwei grafische Tools vorstellen, GQ und den LdapBrowser.

### 4.1 GQ

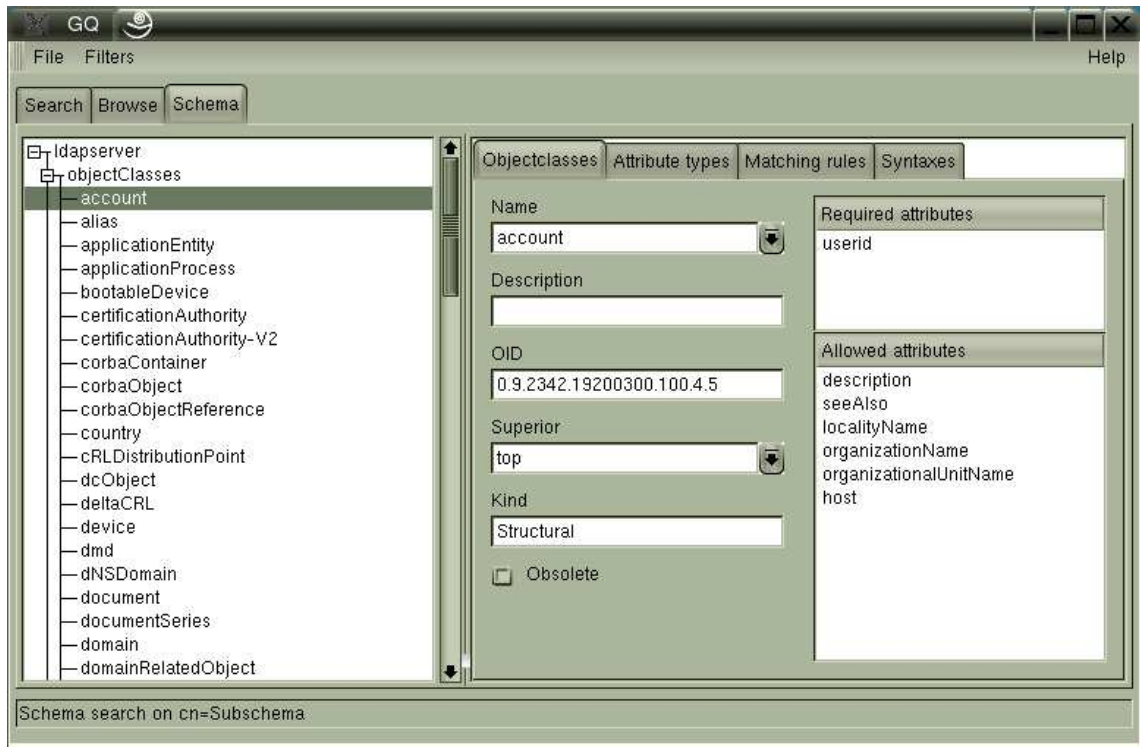
Der Gentleman LDAP Client, so bezeichnet er sich. Ein feiner Client, der mit GTK geschrieben wurde. Er enthält alle wichtigen Features und kann sogar in den Attributen und Objektklassen des Servers browsen.

Er verfügt über Suchfilter und Templates, um damit z.B. Benutzeraccounts anlegen zu können. Bei den meisten Distributionen ist er als rpm-Paket enthalten. Man kann ihn sonst unter

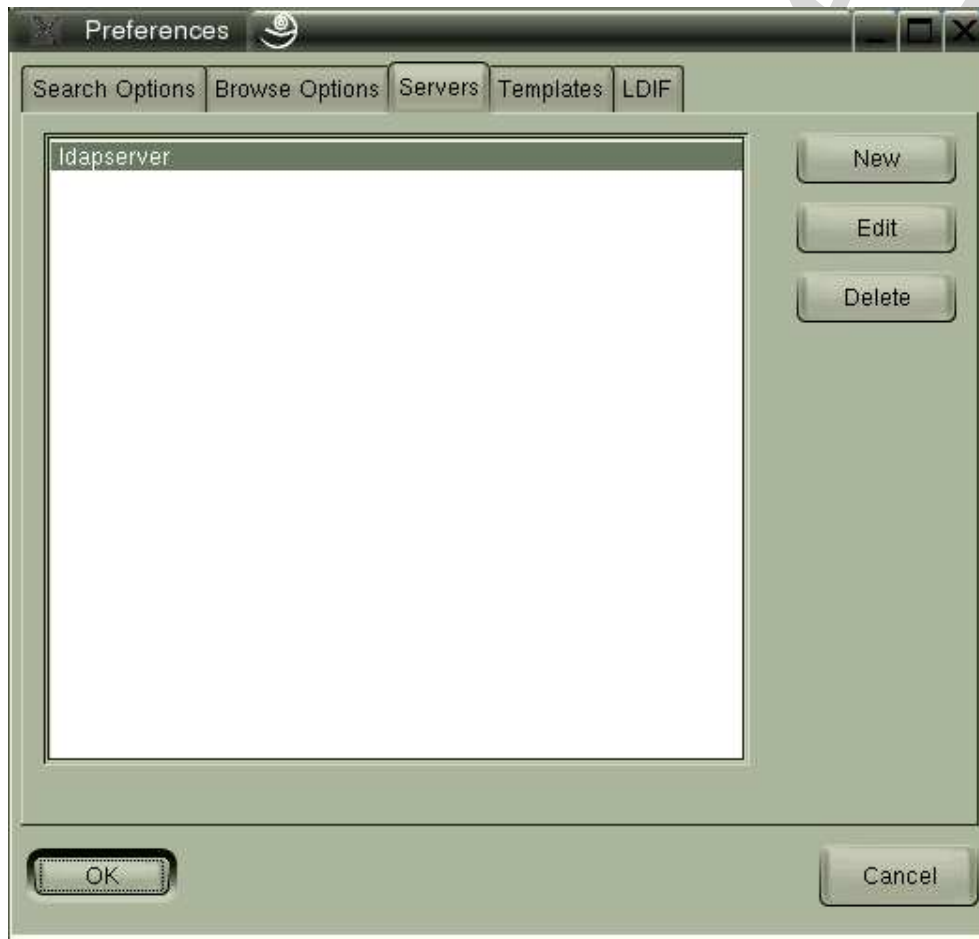
<http://sourceforge.net/projects/gqclient/>

herunterladen:

## Grafische Frontends



Zunächst muß man natürlich den LDAP-Server konfigurieren. Das geschieht unter Preferences. Dort kann dann unter **Servers** der abzufragende LDAP-Baum eingetragen werden:



In einem Dialogfenster können dann die Einstellungen vorgenommen werden:



## 4.2 LdapBrowser

Viele bezeichnen ihn als den König unter den LDAP-Browsern, besonders weil er eine gewisse Ähnlichkeit mit der Console1 von Novell hat. Er ist eines der ausgereifteren Tools und kann unter anderem Templates erzeugen, suchen, browsen und natürlich editieren.

Das Tool ist komplett in Java geschrieben und sollte auch unter Windows laufen. Man kann es von der Seite

<http://www-unix.mcs.anl.gov/gawor/ldap/download.html>

herunterladen und installieren.

Entpacken:

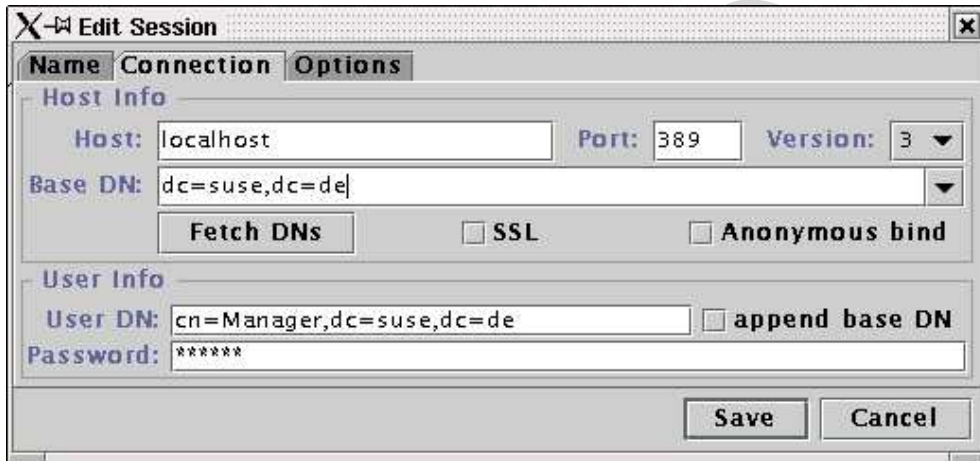
```
tar -xvzf Browser282.tar.gz -C /usr/local
```

Starten:

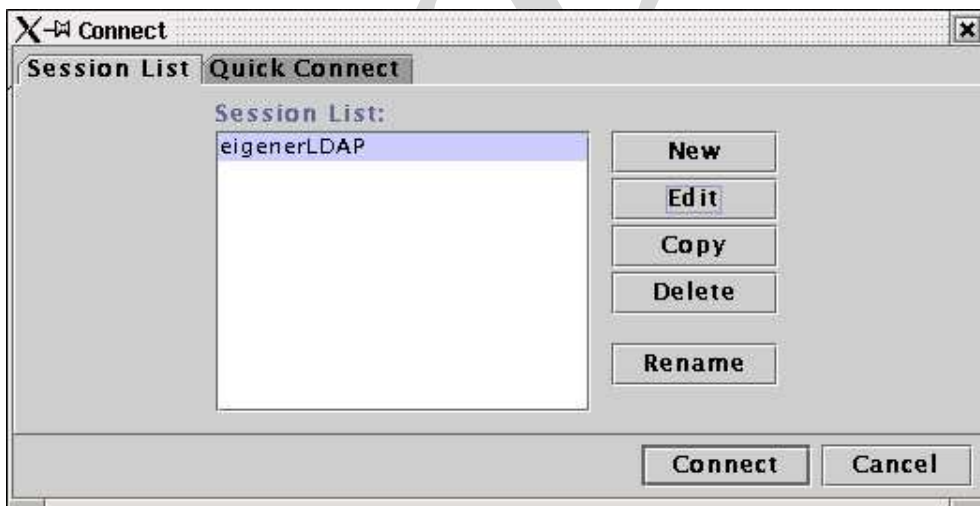
`/usr/local/ldapbrowser/lbe.sh`

Sehr schön ist bei diesem Client, daß man Templates hinterlegen kann, um z.B. Useraccounts anzulegen:

Einstellungen vornehmen:



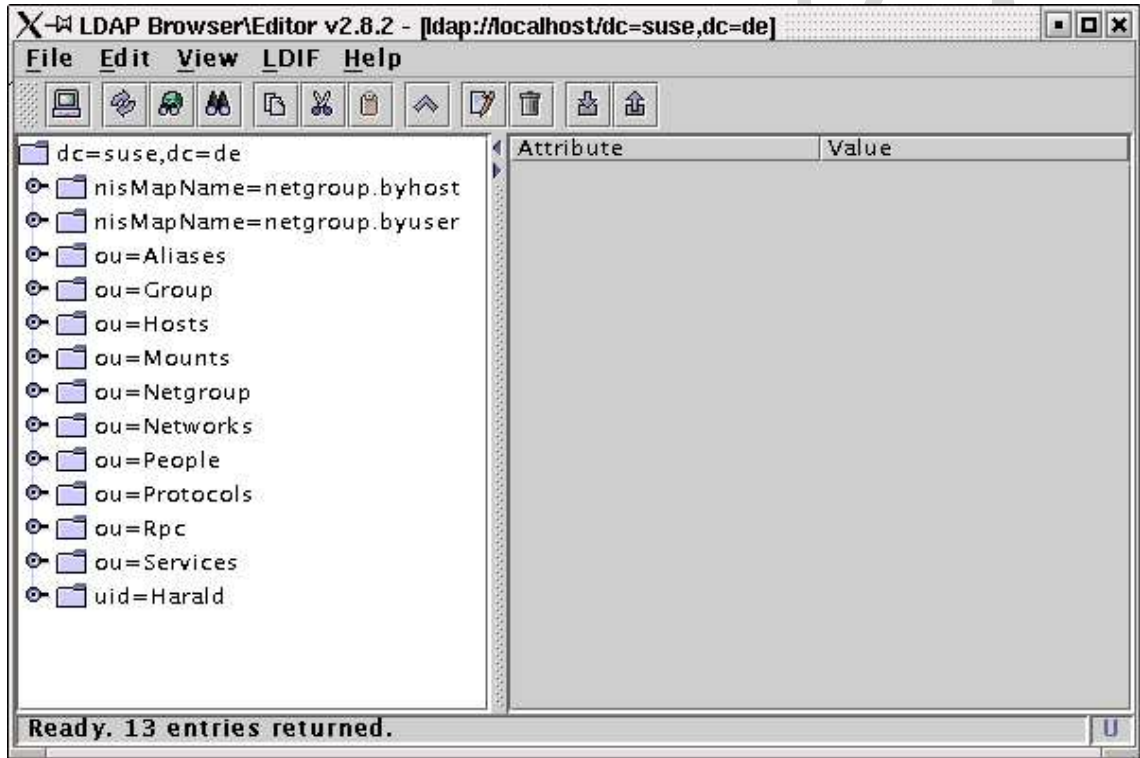
Verbinden:



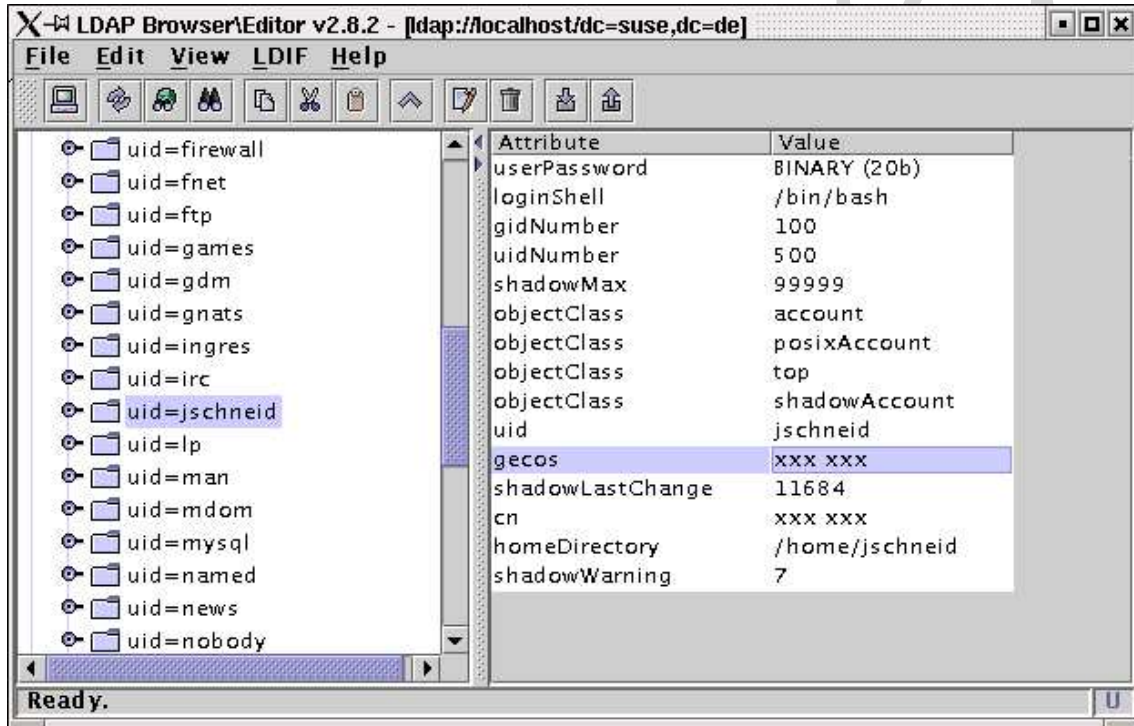
## Grafische Frontends

---

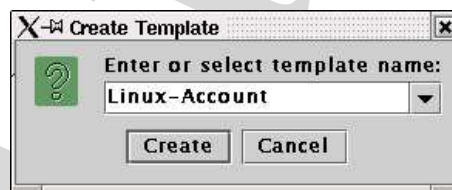
Browsen:

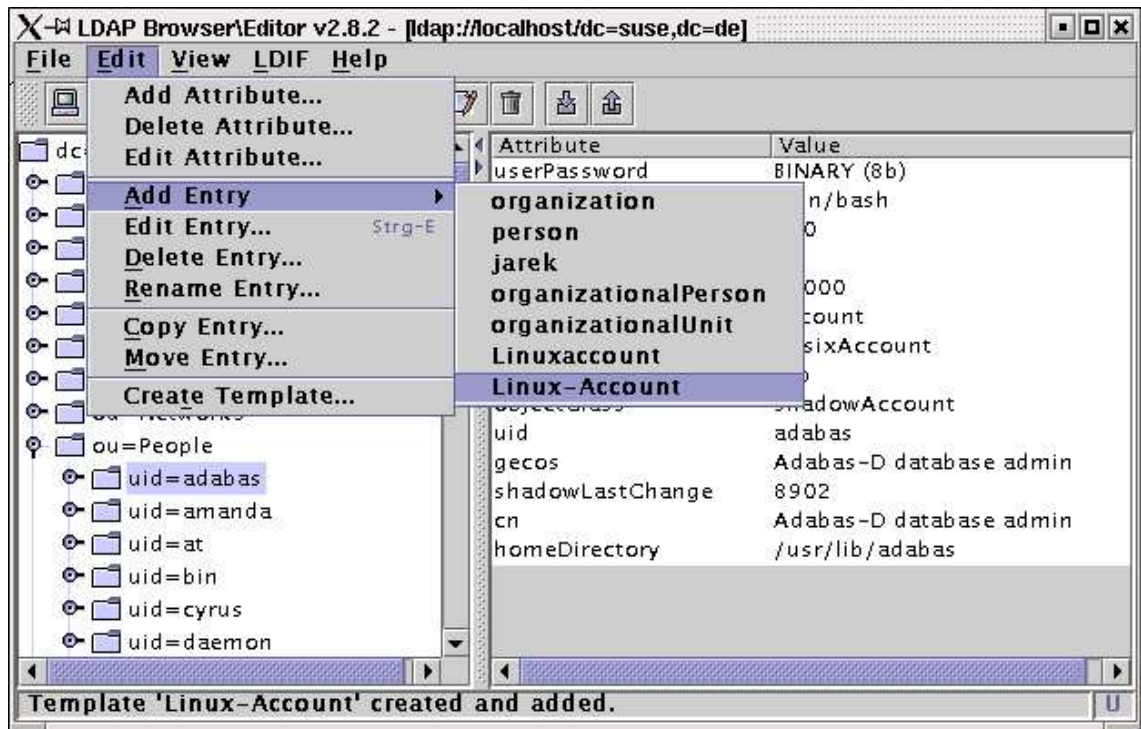


Bearbeiten:



Ein Template erzeugen (z.B. für Benutzeraccounts):





### 4.3 Weitere GUIs

Hier können nicht alle verfügbaren GUI-Frontends für LDAP vorgestellt werden. Interessierte können sich auch die folgenden Clients näher betrachten:

- Frood: <http://frood.sourceforge.net/download.html> (ähnlich wie `gq`, oder `ldapbrowser`).
- likken: <http://www.likken.org/> (sehr einfaches, benutzerfreundliches Frontend)
- web2ldap <http://www.web2ldap.de/> (Webbasierter Client).
- kldap: <http://www.mountpoint.ch/oliver/kldap> (Schöner KDE LDAP-Client, erleichtert ein bißchen die Administration).
- und weitere ...

Daneben gibt es auch eine Reihe ausgereifter, kommerzieller Tools.

## 4.4 Übungen

1. Installieren Sie einen LDAP-Browser ihrer Wahl und, binden Sie ihn an ihren LDAP-Server an. (Anonyme Verbindung).
2. Wechseln sie die Verbindung auf eine authentifizierte Connection und editieren Sie einen Eintrag.

SOLUZIONE

## 5 Stichwortverzeichnis

<b>Symbols</b>	<b>H</b>	Referenzen .....	19
/etc/openldap/ ..	Hilfsobjektklassen .....	Reject-Datei .....	44
13, 31	13	Relative Distinguished	
/etc/openldap/schema/		Names .....	15
.....	<b>L</b>	rootdn .....	31
8, 32	l (LDAP Lokationsname)	rootpw .....	31
/etc/sasldb .....	15		
26	LDAP .....		
/etc/slapd.conf ..	5		
31	LDAP Database Manager		
/var/lib/ldap/ ..	31	<b>S</b>	
31	LDAP-Data-Interchange-	saslpasswd .....	27
	Format .....	saslsblistusers ..	26
<b>A</b>	29	Schema .....	12
ABSTRACT .....	LDAP-URL .....	Scope .....	37
13	22	scope .....	24
Abstract Syntax Notation	ldapadd .....	Search Filter .....	24
8	38, 40	slapadd .....	40
Abstrakte Objektklassen	ldapdelete .....	slapcat .....	40
13	39	slapd .....	31, 36
ACL .....	ldapmodify .....	slapd.conf .....	40, 62
5, 27	38	slapd.oc.conf .....	13
add .....	ldapmodrdn .....	slapindex .....	62
29	39	slurpd .....	31, 41, 44
Alias .....	ldapsearch .....	SSL .....	26
18	36	STRUCTURAL .....	13
Alias Dereferencing ..	ldapsearch .....	Strukturelle Objektklas-	
24	21, 36	sen .....	13
ASN.1 .....	LDBM .....	subordinated .....	20
8	31	Suchfilter .....	24
Attributes to Return ..	LDIF .....	Suffix .....	18
24	29	Superior .....	21
AUXILIARY .....	Limit .....		
13	24	<b>T</b>	
	Limits .....	TLS .....	26
	24		
<b>B</b>	<b>M</b>	<b>U</b>	
base .....	Modell .....	UFN .....	17
24	6	uid (LDAP UID) .....	15
bind .....		unbind .....	23
23		unbinding .....	6
binding .....	<b>N</b>	update .....	29
6	Namensraum .....	User Friendly Name ..	17
	15		
<b>C</b>	normalisierte Form ..		
c (LDAP Land) .....	28	<b>X</b>	
15	<b>O</b>	X.500 .....	5
c (LDAP Staat) .....	o (LDAP Organisations-		
15	name) .....		
cn (LDAP Common Na-	15		
me) .....	Object Identifier .....		
15	8, 53		
Common Name .....	Objekte .....		
24	12		
<b>D</b>	Objektklasse .....		
DAP .....	12		
5	OID .....		
dc (LDAP Domänenkom-	8, 53		
ponente) .....	ou (LDAP Organisations-		
15	einheit) .....		
delete .....	15		
29	<b>P</b>		
Directory .....	Partitionierung .....		
3	17		
Directory Information	<b>R</b>		
Tree .....	RDN .....		
15	15, 17		
Distinguished Name ..	15		
15	RDNs .....		
DIT .....	15		
15			
DN .....			
15, 17			
dn .....			
15			