

# Einführung in LDAP

Handreichung zur Präsentation

Michael Dienert

5. April 2022

## Inhaltsverzeichnis

<b>1</b>	<b>X.500 und LDAP</b>	<b>1</b>
1.1	X.500, DAP, DIT . . . . .	1
1.2	Entwicklung von LDAP . . . . .	1
1.3	Aufbau des Directory Information Trees . . . . .	1
1.4	Aufbau der Verzeichnis-Einträge . . . . .	1
1.4.1	DistinguishedNames . . . . .	2
1.4.2	ObjectClasses . . . . .	2
1.4.3	Definition von ObjectClasses . . . . .	3
1.5	Vererbungshierarchie von ObjectClasses . . . . .	3
1.6	Attribute . . . . .	3
1.6.1	Beispiele . . . . .	4
<b>2</b>	<b>Beispiel für einen Directory Information Tree</b>	<b>4</b>
<b>3</b>	<b>Erzeugen des Baums und Einfügen von Daten</b>	<b>5</b>
3.1	LDAP Data Interchange Files . . . . .	5
<b>4</b>	<b>PAM-login auf RaspberryPI mit slapd</b>	<b>6</b>
4.1	OpenLDAP . . . . .	6
4.2	Optionen von ldapsearch / ldapadd . . . . .	7
<b>5</b>	<b>LDAP-Filter</b>	<b>7</b>
<b>6</b>	<b>Grundkonfiguration des Servers anzeigen lassen</b>	<b>7</b>
<b>7</b>	<b>Änderungen an einem Verzeichnis-Eintrag</b>	<b>8</b>

# 1 X.500 und LDAP

## 1.1 X.500, DAP, DIT

- LDAP basiert auf der CCITT-Spezifikation **X.500**
- X.500 ist ein Dienst, der ursprünglich von der CCITT (heute ITU) zur Namensauflösung und Zustellung elektronischer Post entwickelt wurde (X.400 email-Standard).
- globale Zustellung von email ⇒ erfordert globalen Verzeichnisdienst.
- X.500 besteht aus 2 Teilen:
  - dem X.500 Protokoll: **Directory Access Protocol (DAP)**. Problem: DAP benötigt den kompletten OSI-Stack ⇒ X.500 wurde nie vollständig implementiert!
  - dem X.500 Datenmodell: das Datenmodell besteht aus einem einzelnen **Directory Information Tree (DIT)**. Der DIT ist ähnlich wie der DNS-Baum als *verteilte, baumförmige* Datenbank aufgebaut.

## 1.2 Entwicklung von LDAP

- Das **Lightweight Directory Access Protocol (LDAP)** benötigt lediglich TCP/IP ⇒ weite Verbreitung
- LDAP implementiert nur einen Teil der X.500-Protokollspezifikation ⇒ Fehlendes wird emuliert.
- LDAP wurde 1993 in der RFC 1487 beschrieben und geht auf Arbeiten an der *Universität von Michigan* zurück.

## 1.3 Aufbau des Directory Information Trees

- Beide, LDAP und X.500, speichern ihre Daten in *baumförmigen* Strukturen, den bereits erwähnten **Directory Information Trees (DIT)**.
- Das Datenmodell im DIT ist *objektorientiert*. Es gibt *Objekte*, *Klassen* und -ganz wichtig- *Vererbung*.
- Die *Knoten* im Baum nennt man *Verzeichnis-Einträge* (*Directory Entries* oder *LDAP Entries*).

## 1.4 Aufbau der Verzeichnis-Einträge

Jeder LDAP-Eintrag besteht aus drei Hauptbestandteilen:

**DistinguishedName:** Der DN ist ein *eindeutiger* Schlüssel für den Eintrag.

**Liste mit objectClasses:** Die *objectClasses* bestimmen die Attribute und den Typ des Eintrags.

**Attribute:** Die Attribute enthalten die eigentlichen Daten des Eintrags.

```
# alfred, people, linux-rocks.org
dn: uid=alfred,ou=people,dc=linux-rocks,dc=org
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: shadowAccount
uid: alfred
sn: e.neumann
givenName: alfred
cn: alfred
uidNumber: 4000
gidNumber: 4000
userPassword: SHAGzzM2if4x6ZgXigFPDAGb3PNVxc=
loginShell: /bin/bash
homeDirectory: /home/alfred
```

#### 1.4.1 DistinguishedNames

- Der DN ist der *full qualified name* des Eintrags.
- Er setzt sich zusammen aus dem RDN (relativer DN) des *aktuellen* Knotens und dem DN des *Vorgängerknotens*.
- hier also RDN: uid=alfred

```
# alfred, people, linux-rocks.org
dn: uid=alfred,ou=people,dc=linux-rocks,dc=org
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: shadowAccount
uid: alfred
sn: e.neumann
givenName: alfred
cn: alfred
uidNumber:
...
```

#### 1.4.2 ObjectClasses

- ObjectClasses legen fest, welche Attribute ein Eintrag enthalten muss (MUST) oder enthalten kann (MAY). Ein Eintrag kann mehrere objectClass-Definitionen haben.
- Es muss **exakt eine** ObjectClass geben, die vom Typ STRUCTURAL ist. Diese und ihre Superklassen-Vererbungskette (bis hinauf zur abstrakten Klasse TOP) legen den Typ des Eintrags fest.
- Die weiteren ObjectClasses erweitern den Eintrag (AUXILIARY).

```
# alfred, people, linux-rocks.org
dn: uid=alfred,ou=people,dc=linux-rocks,dc=org
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: shadowAccount
uid: alfred
sn: e.neumann
givenName: alfred
cn: alfred
uidNumber: 4000
gidNumber: 4000
userPassword: SHAGzzM2if4x6ZgXigFPDAGb3PNVxc=
loginShell: /bin/bash
```

```
homeDirectory: /home/alfred
```

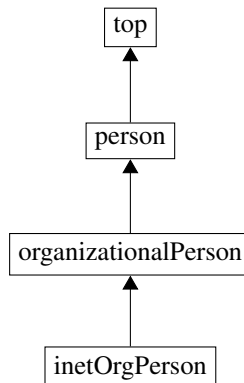
### 1.4.3 Definition von ObjectClasses

```
objectclass ( 2.16.840.1.113730.3.2.2
  NAME 'inetOrgPerson'
  SUP organizationalPerson
  STRUCTURAL
  MAY (
    audio $ businessCategory $ carLicense $ departmentNumber $
    displayName $ employeeNumber $ employeeType $ givenName $
    homePhone $ homePostalAddress $ initials $ jpegPhoto $
    labeledURI $ mail $ manager $ mobile $ o $ pager $
    photo $ roomNumber $ secretary $ uid $ userCertificate $
    x500uniqueIdentifier $ preferredLanguage $
    userSMIMECertificate $ userPKCS12
  )
)
```

```
objectclass ( 1.3.6.1.1.1.2.0
  NAME 'posixAccount'
  SUP top
  AUXILIARY
  DESC 'Abstraction of an account with POSIX attributes'
  MUST ( cn $ uid $ uidNumber $ gidNumber $ homeDirectory )
  MAY ( userPassword $ loginShell $ gecos $ description )
)
```

### 1.5 Vererbungshierarchie von ObjectClasses

Alle STRUCTURAL-Classes müssen direkt oder abgeleitet die abstrakte Klasse TOP erweitern:



### 1.6 Attribute

- Attribute sind Name-Werte-Paare und legen die eigentlichen Daten eines Eintrags fest.
- Die Attribute (z.B. uid, sn, uidNumber, ...) werden mit AttributeTypes definiert.
- Auch bei Attributen existiert eine Vererbungshierarchie (SUP)

```
# alfred, people, linux-rocks.org
dn: uid=alfred,ou=people,dc=linux-rocks,dc=org
objectClass: inetOrgPerson
objectClass: posixAccount
```

```
objectClass: shadowAccount
uid: alfred
sn: e.neumann
givenName: alfred
cn: alfred
uidNumber: 4000
gidNumber: 4000
loginShell: /bin/bash
homeDirectory: /home/alfred
userPassword:: e1NIQX1HenpNM ... UE5WeGM9
```

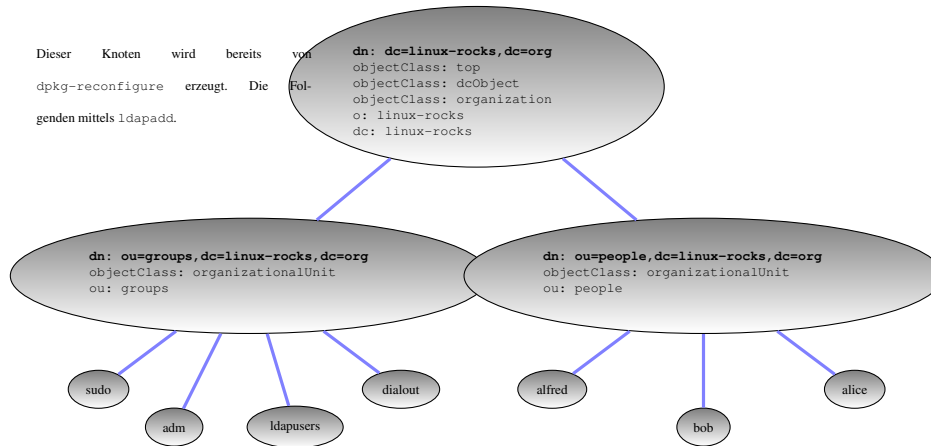
### 1.6.1 Beispiele

```
attributeType: ( 0.9.2342.19200300.100.1.1
NAME ( 'uid' 'userid' )
DESC 'RFC4519: user identifier'
EQUALITY caseIgnoreMatch
SUBSTR caseIgnoreSubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15256
)
```

```
attributeType: ( 1.3.6.1.1.1.1.0
NAME 'uidNumber'
DESC 'RFC2307: An integer uniquely identifying a user in an
administrative domain'
EQUALITY integerMatch
ORDERING integerOrderingMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
SINGLE-VALUE
)
```

```
attributeType ( 2.5.4.11
NAME ( 'ou' 'organizationalUnitName' )
SUP name
)
```

## 2 Beispiel für einen Directory Information Tree



```
dn: cn=dialout,ou=groups,dc=linux-rocks,dc=org
objectClass: posixGroup
cn: dialout
gidNumber: 20
memberUid: alfred
memberUid: alice
memberUid: bob
```

```
dn: cn=ldapusers,ou=groups,dc=linux-rocks,dc=org
objectClass: posixGroup
cn: ldapusers
gidNumber: 4000
```

```
dn: uid=alfred,ou=people,dc=linux-rocks,dc=org
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: shadowAccount
uid: alfred
sn: e.neumann
givenName: alfred
cn: alfred
uidNumber: 4000
gidNumber: 4000
userPassword:: e1NIQXlHenpNm1mNHg2WmdYaWdGUERBR2IzUE5WeGM9
loginShell: /bin/bash
homeDirectory: /home/alfred
```

Gruppen adm und sudo entsprechend.  
bob und alice entsprechend.

### 3 Erzeugen des Baums und Einfügen von Daten

Um den LDAP-Baum aufzubauen, gibt es mehrere Möglichkeiten:

1. Einfügen von Einträgen direkt in die Backend-Datenbank: slapadd, slapcat, ...
2. Einfügen von Einträgen mit dem LDAP-Protokoll selbst: ldapsearch, ldapadd, ldappaswd, ldapdelete, ...
3. Mit grafischen Werkzeugen, sog. LDAP-Browsern: Apache Direct Studio, jexplore, luma, ...
4. Durch Einspielen eines *LDAP Data Interchange Files*

#### 3.1 LDAP Data Interchange Files

- reine Textdateien, bilden DIT ab
- DIT-Einträge durch Leerzeilen trennen
- Line-Continuation: Leerzeichen am Zeilenanfang der umgebrochenen Zeile. Escape-Character sind unnötig

```
# alfred, people, linux-rocks.org
dn: uid=alfred,ou=people,dc=lin
ux-rocks,dc=org
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: shadowAccount
```

```
uid: alfred
sn: e.neumann
givenName: alfred
cn: alfred
uidNumber: 4000
gidNumber: 4000
userPassword: {SHA}GzzM2if4x6ZgXigFPDAGb3PNVxc=
loginShell: /bin/bash
homeDirectory: /home/alfred
```

## 4 PAM-login auf RaspberryPI mit slapd

- LDAP-Praxis: Benutzer auf einem RaspberryPi (oder Notebook) in einem openLDAP-Verzeichnis anlegen
- raspbian mit dd auf micro-sd kopieren, einstecken, PI booten, mit dhcp-Server verbinden
- auf raspies per ssh anmelden. Passwort der raspies: ljwml!

```
ping ff02::1%np3s0
ssh pi@fe80::8196:18e9:c122:fbbb%np3s0 #durchsichtiges gehaeuse
ssh pi@fe80::f584:7d29:356f:ac78%np3s0 #pi400
ssh pi@fe80::e576:b770:e9cd:e3ee%np3s0 #joy-it gehaeuse
ssh pi@fe80::c46a:62d1:f743:58dc%np3s0 #ohne gehaeuse PI 3
#ohne gehaeuse PI 2
```

### 4.1 OpenLDAP

#### 1. warum OpenLDAP?

- frei, quelloffen
- Standard unter Linux, BSD und OSX-Server
- Direkter Ableger der ersten LDAP-Implementierung der Univ. of Michigan.
- unzählige Programmierschnittstellen (JAVA, Python, Perl, ...)
- Anbindung an viele Dienste möglich (out-of-the-box)

#### 2. Installieren: slapd = Standalone LDAP Demon, ldap-Hilfprogramme:

```
aptitude install slapd ldap-utils
```

#### 3. Alles Weitere: ReadTheFine-HowTo: <https://dt.wara.de/fobiLdap/ldapAuthPI-howto.txt>

- DIT ausgeben lassen:

```
ldapsearch -D "cn=admin,dc=linux-rocks,\
dc=org" -b "dc=linux-rocks,dc=org" -w ljwml!
```

- ldif-Datei einspielen:

```
ldapadd -x -D "cn=admin,dc=linux-rocks,\
dc=org" -w ljwml! -f irgendwas.ldif
```

- Mit folgendem Kommando kann man wieder komplett von vorne starten:

```
dpkg-reconfigure slapd
```

Details: RTFHowTo

## 4.2 Optionen von `ldapsearch` / `ldapadd`

Optionen `-b`, `-D`, `-f`, `-W`, `-x`

- b** Nur bei `ldapsearch`; gibt die *search-base* an, also den Knoten, ab dem der DIT ausgegeben wird.
- D** Distinguished Name (=abs. Pfad) desjenigen, der sich mit dem Verzeichnis verbindet (*bind*). Muss ggfs. autorisiert sein. Aka: *bindUser*
- f** File.
- W** Passwort über Eingabeaufforderung einlesen. Alternative: mit Option `-w` direkt mitgeben.
- x** einfache Autorisierung anstelle von SASL verwenden

## 5 LDAP-Filter

- LDAP-Filter bestehen aus Boole'schen Ausdrücken der Form

```
Attribut Operator Wert  
mit den Operatoren: =  ~=  <  <=  >  >=
```

- Grössenvergleich nur bei Attributen mit ORDERING-Matching-Rule möglich.
- Substring-Matching mit `*`.
- Verkettung der Filter mit UND (&) ODER (|) NICHT (!) in *Polnischer Notation* (& (cn=a)(sn=b))
- Viele runde Klammern setzen!

```
ldapsearch -D "cn=admin,dc=linux-rocks,dc=org" \  
-b "dc=linux-rocks,dc=org" -w ljwml! \  
'(|(sn=imwunderland) (&(sn=bau*)(uid=bob)))'
```

## 6 Grundkonfiguration des Servers anzeigen lassen

Mit folgender Kommandozeile kann man sich die komplette *slapd* Grundkonfiguration einschliesslich aller geladener Schemata ansehen:

```
ldapsearch -Y EXTERNAL -H ldap:// -b "cn=config" |less
```

- Das Kommando setzt root-Rechte voraus (`uid=0`), sonst bekommt man nicht viel zu sehen. Der Benutzer mit der `uid=0` (root) hat alle Rechte.
- verantwortlich dafür ist das Attribut `olcAccess`



## 7 Änderungen an einem Verzeichnis-Eintrag

Mit folgender LDIF-Datei können einige Einträge von Alfred geändert werden:

```
version: 1

##änderungen bei alfred

dn: uid=alfred,ou=people,dc=linux-rocks,dc=org
changetype: modify
add: employeeType
employeeType: vollzeit
-
replace: uid
uid: aeneumann
```

Einspielen lässt sich diese LDIF-Datei mit:

```
ldapadd -x -D "cn=admin,dc=linux-rocks,dc=org" -w ljwml! -f aenderung.ldif
```

```
version: 1

##änderungen bei alfred

dn: cn=Alfred E. Neumann,
  ou=lehrer,dc=wara,dc=de
changetype: modify
add: employeeType
employeeType: vollzeit
-
replace: uid
uid: aeneumann
```

- `changetype: modify` wird verwendet, wenn man den Eintrag ändern möchte.
- mit `changetype: delete` kann man ihn komplett löschen.
- `add: employeeType` benötigen wir, wenn wir ein Attribut hinzufügen möchten. Das Attribut mit seinem Wert muss unmittelbar folgen.
- `replace: uid` ersetzt das Attribut durch das direkt folgende Attribut-Wert Paar.
- dann gibt es noch: `delete: mail`. Damit würde das entsprechende Attribut gelöscht.