LDAP Grundlagen

als Einführung zum

Workshop

Inhaltsverzeichnis

Was ist ein Verzeichnisdienst?....3
Die aktuelle LDAP Version 3....3
Der Einsatz von LDAP im Netzwerk....3
Aufbau des LDAP Datenmodell....4
Objekte....5
Attribute....6
Verzeichniseinträge....6
Vererbung....7
Klassenzugehörigkeit und Polymorphie....7
Der Verzeichnisbaum....7
Aliase....8
Verwaltung der Namen im DIT mit dem Distinguished Name....8
Der "Common Name" eines Objektes....9
Zusammengesetzte RDN....9
Vom RDN zum DN....9
Kodierung von RDN....10
Schema....11
Aufbau eines Schemas....12
Referenzierung von Objekten durch Namen oder OID....12
Eigenes Schema entwerfen....13
Kommandos zur Verwaltung des LDAP....13
Das Kommando "ldapsearch"....14
Das Kommando ldapadd....14

Hier sollen nun ein paar Grundlagen zum Thema LDAP (Lightweight Directory Access Protocol) besprochen werden. Die Schwerpunkte liegen auf den folgenden Themen:
- Grundlagen eines Verzeichnisdienst
- Datenmodelle
- Namenskonzepte
- Kommandos für die Administration

Was ist ein Verzeichnisdienst?

In einem Verzeichnisdienst werden Informationen in einer hierarchischen Struktur abgelegt, die auch als baumartige Struktur bezeichnet wird.
Die Vorteile einer solchen Struktur sind folgende:
- Teile der Struktur können von verschieden Administratoren verwaltet werden.
- Unternehmensstrukturen können 1:1 abgebildet werden.
- Die Struktur kann auf mehrere Server verteilt werden.
- Die Suche nach bestimmten Inhalten kann auf bestimmte Bereiche eingeschränkt werden.
- Da LDAP auf X.500 basiert, werden hier objektorientierte Datenmodelle verwendet.

Neben LDAP gibt es ein zweites, gutes Beispiel für die Verwendung eines Verzeichnisdienstes, nämlich DNS, auch dort wird eine Baumstruktur verwendet.

Für Verzeichnisdienste existieren eine Reihe von X.500-Standards. Diese X.500-Standards beschreiben, wie Verzeichnisdaten zur Verfügung gestellt und abgerufen werden sollen und wie die Verschlüsselung, Authentifizierung, Replikation und Verwaltung der Verzeichnisdaten gehandhabt werden sollen. Die X.500-Standards liefern die Funktionsmodelle und Begriffsbestimmungen für die Verzeichnisdienste, die nicht voll auf dem X.500-Standard basieren, in diesem Fall LDAP.

Die aktuelle LDAP Version 3

Die aktuelle Version von LDAP ist die Version 3, es wird zwar an vielen Orten noch die Version 2 verwendet, die Version 3 bietet aber einige Vorteile gegenüber Version 2. Trotzdem sollte man nicht vergessen, das es noch Clientanwendungen gibt, die nur die Version 2 unterstützen. Hier nun die Neuerungen der Version 3 gegenüber der Version 2:
- Authentifizierung durch SASL
- Verschlüsselung aller Daten durch TLS
- Möglichkeit der Verwendung von UTF-8
- Verweise auf andere LDAP-Server, die "Referrals"

LDAPv3 ist nicht abwärtskompatibel und sollte nicht mit LDAPv2 Servern in einer Umgebung betrieben werden.

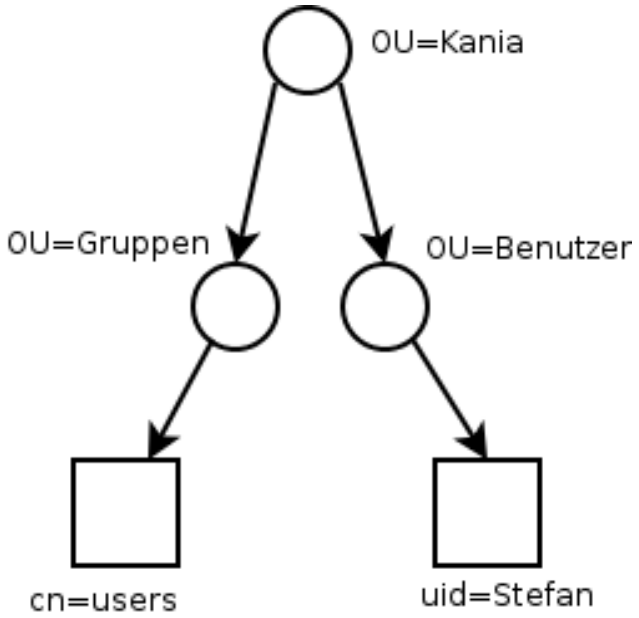
Der Einsatz von LDAP im Netzwerk

LDAP kann auf verschiedene Arten und Weisen im Netzwerk für die Verwaltung eingesetzt werden. Auch ist LDAP die Grundlage anderer Verzeichnisdienste wie zum Beispiel der Novell NDS oder des Microsoft Active Directories. Mit Hilfe von LDAP kann in Netzwerken die Verwaltung der Ressourcen vereinfacht werden. Auch ist mit LDAP ein "single-sign-on" im Netzwerk realisierbar.

Hier nun eine Liste der möglichen Anwendungen von LDAP:
- Verwaltung von Benutzern, Gruppen und Rechnern (passwd, groups, hosts)
- Verwaltung von IP-Diensten (services)
- Zuordnung von Protokollen (protocols)
- RPC- Zuordnungen (rpc)
- NIS Informationen
- Boot Informationen (MAC-Addresse und boot Parameter)
- Mountpoints für das Dateisystem zur Verwendung des "Automounters"
- Verwaltung von Mail-Aliasen
- Verwaltung der DNS Zonendateien
- Verwaltung der DHCP-Server Informationen
- Authentifizierung von Benutzen beim Squid und Apache

Aufbau des LDAP Datenmodell

Das Datenmodell von LDAP besteht aus Objekten. Hier gelten fast die gleichen Regeln wie bei der objektorientierten Programmierung. Auch im LDAP gibt es Objekte, Klassen, Vererbung und Polymorphie.
Die Aufgabe von LDAP ist es, die Objekte abzubilden und miteinander in Beziehung zu bringen. Ein Objekt wird im LDAP als Verzeichniseintrag bezeichnet. Ein Verzeichniseintrag besteht aus einer Liste von Attributen des Objekts. Der Name eines Objekts wird im Verzeichnisdienst als "Distinguished Name" bezeichnet, der ähnlich wie der Name einer Datei im Dateisystem behandelt wird. Durch die verschiedenen Objekte entsteht so nach und nach eine Baumstruktur.

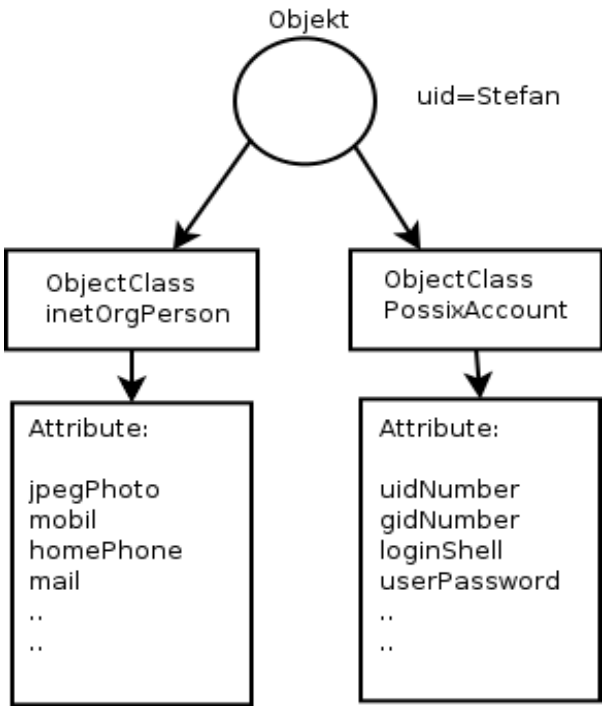


Die Objekte mit der Bezeichnung "ou" sind Containerobjekte, in diesen Objekten können weitere Objekte erzeugt werden. Diese Objekte werden zum Aufbau der Struktur verwendet.
Die Objekte mit der Kennzeichnung "cn" und "uid" dienen zur Verwaltung der Ressourcen, in diesem Fall eines Benutzer "uid=Stefan" und einer Gruppe "cn=users". Die Bezeichnungen werden aus den verschiedenen Objektklassen und Schemata abgeleitet,

auf die im Verlauf noch weiter eingegangen wird. Im Gegensatz zu der NDS oder dem ADS können im LDAP unterhalb des Blattobjektes weitere Objekte erzeugt werden. So ist es möglich unterhalb einen Benutzers eine weitere OU zu erzeugen, in der dann, zum Beispiel, das persönliche Adressbuch des Benutzers gespeichert werden kann.
In den nächsten Abschnitten werden die einzelnen Objekte des LDAP Verzeichnisdienst weiter erläutert.

Objekte

Ein Objekt im LDAP ist eine zu verwaltende Ressource, sie kann die unterschiedlichsten Typen wiederspiegeln. Ein Objekt kann sowohl ein Container, in dem weitere Objekte verwaltet werden, als auch ein Benutzer oder eine Gruppe sein. Eines ist bei allen Objekten aber gleich, alle Objekte haben Eigenschaften (Attribute). Die Attribute unterscheiden sich je nachdem, um was für ein Objekt es sich handelt. Das wichtigste Attribut eines Objekts ist der Objektname, da über dieses Attribut das Objekt im Verzeichnisdienst verwaltet wird. Ein Objekt kann für die Verwaltung der unterschiedlichsten Aufgaben verwendet werden. Für jede Aufgabe benötigt ein Objekt unterschiedliche Attribute, deshalb werden Objekte zu den Objektklassen zusammen gefasst. Wenn zum Beispiel ein Objekt Benutzer für die Anmeldung an einem Unix-System verwendet werden soll, muss das Objekt der "objectclass" PossixAccount angehören, da nur in der Objekt-Klasse alle Attribute enthalten sind, die für eine erfolgreiche Anmeldung benötigt werden.



Im folgenden ein Beispiel für eine "ObjektClass". Hier wird auch deutlich, das bestimmte Attribute vergeben werden müssen, andere vergeben werden können:

```
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 ) )
```

Attribute

Attribute sind die Eigenschaften von Objekten. Ein Attribut besteht aus:
- Einem Namen, mit dem das Attribut innerhalb eines Objekts eindeutig referenziert werden kann.
- Unterschiedlicher Anzahl von Werten. Es gibt Attribute die können ohne Wert bleiben, es gibt Attribute die müssen genau einen Wert haben und es gibt Attribute die können mehrere Werte haben.

Um die Wiederverwendbarkeit von Attributen in verschiedenen Objekt-Klassen zu ermöglichen, werden Attribute getrennt von Objekten verwaltet, und zwar in Form von Attributtypen. Der "attributeType" enthält die folgenden Komponenten:
- Der Name und der OID, die den Attributtyp eindeutig beschreibt. Die OIDs werden später noch genauer erklärt.
- Eine für den Menschen lesbare Beschreibung (description). Die Beschreibung ist optional
- Optionale Definitionen darüber, welche Regeln für die Suche gelten. Hier gibt es die Möglichkeiten der Gleichheit (EQUALITY), das Auswerten von Teilzeichenketten (SUBSTR) und die alphabetische Anordnung (ORDERING)
- Eine Syntaxbeschreibung, meist in Form einer OID. Dadurch wird festgelegt, um was für ein Art von Objekt es sich hier handelt, zum Beispiel eine Zahl, eine Zeichenkette oder ein anderes Objekt.
- Ein Qualifier, der festlegt ob das Attribut nur einen Wert (SINGLE-VALUE) hat, oder mehrere Werte (COLLECTIVE) haben kann. Hierbei kann über den Wert (LENGTH) auch noch die Länge der Werteliste begrenzt werden.

Hier ein Beispiel für eine Attributbeschreibung:

```
attributetype ( 2.5.4.5 NAME 'serialNumber'
      DESC 'RFC2256: serial number of the entity'
      EQUALITY caseIgnoreMatch
      SUBSTR caseIgnoreSubstringsMatch
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.44{64} )
```

Verzeichniseinträge

Ein Objekt wird im LDAP-Baum durch einen Verzeichniseintrag repräsentiert. Ein Verzeichniseintrag wird dadurch erstellt, dass man einem Objekt bestimmte Objektklassen zuweist und die Attribute mit Werten füllt. Die Verzeichniseinträge entsprechen somit den "Instanzen" in der objektorientierten Programmierung.
Ein Verzeichniseintrag besteht aus einer Menge von Attributen und gehört einer oder

mehreren Objektklassen an. Durch die Objektklassen wird festgelegt, welche Attribute obligatorisch und welchen Attribute fakultativ sind.

Vererbung

Ein weiteres Konzept aus der OOP, dass im LDAP eine Rolle spielt, ist die Vererbung. Eine Objektklasse kann als Subklasse einer anderen Klasse definiert werden und erbt dann deren Attribute. Um eine bestehenden Klasse um Attribute zu erweitern, kann einfach eine neue Klasse von der bestehenden Klasse abgeleitet werden, somit brauchen bestehende Definitionen nicht wiederholt werden.

Klassenzugehörigkeit und Polymorphie

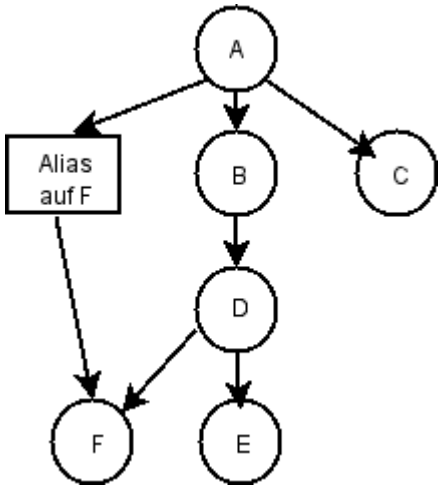
Ein Verzeichniseintrag hat ein Attribut vom Typ "objectClass", das festlegt, welcher Klasse der Eintrag angehört. Da das Attribut "objectClass" "multiValued" ist, kann ein Verzeichniseintrag zu mehreren Objektklassen gehören, ähnlich der Mehrfachvererbung in der OOP.
Bei den Verzeichniseinträgen im LDAP ist diese Polymorphie die Regel. Jedes Objekt muss entweder der Klasse "top" oder "alias" angehören. Ein Eintrag von der Klasse "top" kennzeichnet hier ein echtes Objekt, ein Eintrag von der Klasse "alias" kennzeichnet hingegen nur einen Verweis auf ein echtes Objekt. Mehr zu Aliasen später.

Der Verzeichnisbaum

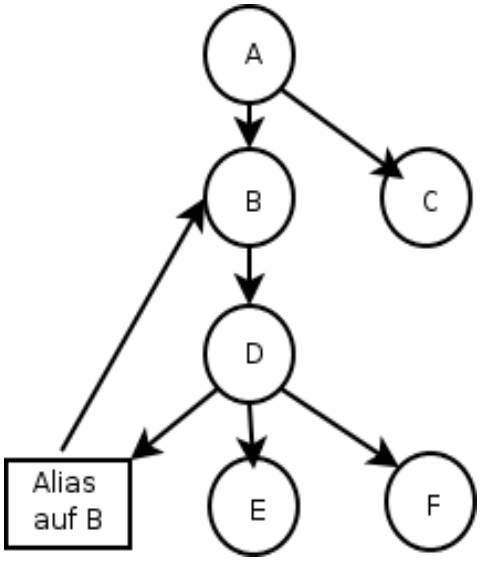
Das besondere Merkmal eines Verzeichnisdienstes ist die Struktur, in der er angeordnet ist. Die Objekte werden in einer Baumstruktur verwaltet. Dieser Baum wird im X.500 als DIT "Directory Information Tree" bezeichnet. Der Baum kommt dadurch zustande, dass manche Verzeichniseinträge anderen übergeordnet sind. Objekte, die weitere Objekte beinhalten können, werden als "organisational unit (ou)" bezeichnet. Objekte, die keine weiteren Objekte enthalten können, werden als Blätter "leaves" bezeichnet. Auf Grund der Baumstruktur ist es jetzt möglich, die Administration oder die Suche auf einen Teilbaum zu beschränken.

Aliase

Oftmals ist der DIT kein richtiger Baum, da es Objekte in höheren Ebenen gibt, die auf Objekte in weiter unten liegenden Eben verweisen. Diese Objekte werden im LDAP als alias bezeichnet und haben nur die Funktion eines Links, ähnlich eines Softlinks im Dateisystem. Hier ein Beispiel:



Es können aber nicht beliebige Aliase gesetzt werden, es müssen auf jeden Fall die "deadlocks" vermieden werden, bei denen eine Alias im Kreis auf sich selber zeigt. Hier ein Beispiel für so einen "deadlock":



Ein Alias muss deshalb immer oberhalb des Objektes definiert werden, auf das er zeigt.

Verwaltung der Namen im DIT mit dem Distinguished Name

Damit ein Objekt im DIT eindeutig ist, müssen die Objekte im DIT über ihren Namen referenziert werden können. Da der Name des Objekt zur Unterscheidung verwendet wird, spricht man hier vom "Distinguished Name (DN)". Der "Distinguished Name" setzt sich aus mehreren kleinen Bausteinen zusammen den "Relative Distinguished Name (RDN)" .

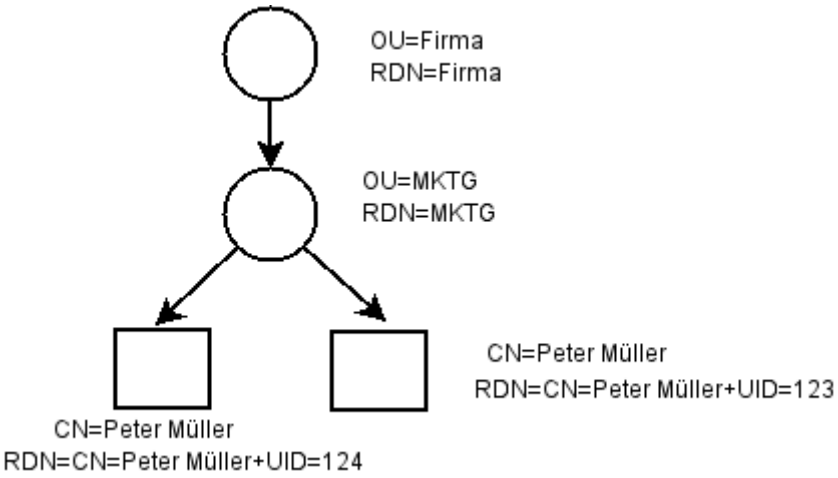
Der RDN wird dazu verwendet, die Einträge unterhalb der selben übergeordneten OU zu unterscheiden. Einträge in anderen OU's des selben DIT können den selben RDN verwenden. Hier gilt also das gleiche Prinzip wie im Dateisystem. Der RDN besteht aus einem oder mehreren Attributen, diese können im Prinzip beliebig gewählt werden, solange es sicher gestellt ist, dass der Eintrag auf seiner Hierarchieebene eindeutig ist.

Der "Common Name" eines Objektes

Die meisten Objekte im DIT spiegeln ein reales Objekt wieder. Reale Objekte haben in der Regel einen Namen. Zum Beispiel haben die Benutzer, die im DIT verwaltet werden, einen Vor- und einen Nachnamen. Dieser Name wir im DIT als "Common Name" verwaltet. So kann der Benutzer Stefan Kania im DIT mit dem Attribut cn="Stefan Kania" verwaltet werden. Das Attribut "cn" wir dann gleichzeitig als RDN des Objekts verwendet.

Zusammengesetzte RDN

Oft reicht es nicht aus nur ein Attribut als RDN zu verwenden, was ist zum Beispiel wenn es in einer Abteilung eines Unternehmens zwei Mitarbeiter mit dem Namen Peter Müller gibt, die RDN's aber auf jeden Fall den CN enthalten sollen. Dann muss es eine Lösung geben, wie mehrere Attribute zusammengesetzt werden können, so dass der RDN wieder eindeutig ist. Das folgende Beispiele soll das verdeutlichen:



Der RDN setzt sich hier aus den Attributen CN und UID zusammen und wird somit wieder eindeutig. Das "+" Zeichen wird im RDN zur Verbindung von Attributen verwendet.

Vom RDN zum DN

Jedes Objekt hat einen eindeutigen absoluten Namen, den DN. Ähnlich den Namen im Dateisystem setzt sich der DN aus dem Pfad durch den DIT zusammen. Für die beiden Benutzer würde der DN so aussehen:
cn=Peter Müller+uid=123,ou=MKTG,ou=firma
cn=Peter Müller+uid=124,ou=MKTG,ou=firma

Kodierung von RDN

Sonderzeichen, wie zum Beispiel das Komma, das als Trennzeichen verwendet wird, müssen im RDN durch einen Backslash entwertet werden. Dadurch verlieren sie ihre besondere Bedeutung.
Ein anderes Problem sind die Umlaute, am einfachste ist es, "ö" durch "oe", "ä" durch "ae", "ü" durch "ue" und "ß" durch "ss" zu ersetzen, aber es gibt eine Möglichkeit dieses zu umgehen. Mit dem Programm "iconv" können Dateien von einem Zeichensatz in einen anderen konvertiert werden.
Die Objekte im LDAP werden später über die LDIF-Dateien eingespielt. Bei den LDIF-Dateien handelt es sich um ASCII-Text Dateien in denn die Attribute eines Objekts definiert sind. Diese Dateien können mit "iconv" von ISO8859-1 in UTF-8 umgewandelt werden. Hier ein Beispiel:
Hier die Datei vor der Umwandlung:

```
dn: uid=stmü,ou=users,dc=home,dc=local
givenName: Stefan
sn: Müller
loginShell: /bin/bash
uidNumber: 500
gidNumber: 100
shadowExpire: 0
uid: stka
objectClass: top
objectClass: posixAccount
objectClass: shadowAccount
objectClass: inetOrgPerson
shadowLastChange: 12381
cn: Stefan Müller
homeDirectory: /home/stmü
mail: stka@home.local
```

Jetzt wir die Datei mit dem folgenden Kommando nach UTF-8 umgewandelt:

```
iconv -f iso8859-1 -t utf-8 -o user-utf8.ldif user.ldif
```

Das Ergebnis sieht dann so aus:

```
dn: uid=stmÃŒ,ou=users,dc=home,dc=local
givenName: Stefan
sn: MÃŒller
loginShell: /bin/bash
uidNumber: 500
gidNumber: 100
shadowExpire: 0
uid: stka
objectClass: top
objectClass: posixAccount
objectClass: shadowAccount
objectClass: inetOrgPerson
shadowLastChange: 12381
cn: Stefan MÃŒller
homeDirectory: /home/stmÃŒ
mail: stka@home.local
```

Jetzt kann die Datei mit Sonderzeichen eingefügt werden. Damit openLDAP die Informationen verwenden kann, muss in der Konfigurationsdatei "slapd.conf" der folgende Eintrag als erste Zeile eingetragen werden:

```
ucdata-path /usr/share/openldap/ucdata
```

Jetzt werden die Umlaute im LDAP richtig angezeigt. Die meisten grafischen Werkzeuge für den LDAP führen die Umwandlung automatisch durch. Hier muss dann nicht erst eine Umwandlung von Hand durchgeführt werden.

Schema

Bis jetzt wurde immer von einzelnen Objekten, Objektklassen und Attributen gesprochen. Jetzt geht es darum zu sehen, wie die einzelnen Teile zu einem Schema zusammen gefasst und verwaltet werden. Ein Schema definiert die folgenden Punkte:

- Die verwendeten Attributstypen.
- Die verwendeten Objektklassen.
- Filter- und Matching- Regeln bei Vergleichsoperationen.
- Rechte zum Anlegen oder Modifizieren von Datensätzen.

Im LDAP werden die Schemata textorientiert verwaltet. Die Dateien der Schemata werden im openLDAP in einem eigenen Verzeichnis verwaltet. In diesem Verzeichnis befinden sich verschieden Dateien mit der Endung *.schema. Hier eine Auszug aus der Liste der Schemata die zur Verfügung stehen:

- *core.schema*
  Eine Sammlung von wichtigen Informationen für dem Betrieb eines LDAP Servers
- *cosine.schema*
  Auch in diesem Schema befinden sich ein Menge an Attributen und Objektklassen die als Grundlage für die Verwaltung von Objekten dienen.

- *inetorgperson.schema*
  In diesem Schema werden zusätzliche Informationen verwaltet, die im Bezug auf die Benutzerverwaltung wichtig werden können.
- *samba.schema*
  Hier sind alle Attribute und Objektklassen abgelegt, die für die Veraltung eines samba-Servers über LDAP notwendig sind.

Bei der Einbindung eines Schemas ist darauf zu achten, zu welchem anderen Schema Abhängigkeiten bestehen. Die Abhängigkeiten sind in den jeweiligen Schemendateien nachzulesen. Auf Grund der Abhängigkeiten ist es wichtig, die Reihenfolge in der die Schemata bei dem Start des LDAP Servers geladen werden einzuhalten.

Aufbau eines Schemas

Die Schemata sind nach einem einfachen Prinzip aufgebaut. Neben Kommentarzeilen, die durch eine Raute "#" eingeleitet werden, listet das Schema die Attribute und Objektklassen (attributetype / objectclass) auf. Hinter diesen Schlüsselwörtern folgen in runden Klammern der eingefasste Bereich, in dem die Spezifikationen abgelegt werden.

Referenzierung von Objekten durch Namen oder OID

Ein Eintrag in einem Schema kann auf zwei verschiedene Arten referenziert werden.

- Zum Einen durch einen Namen, der im Prinzip beliebig vergeben werden kann. Als Beispiel hier für ein Auszug aus dem "NIS" Schema:

```
NAME 'loginShell'
```

- Zum Anderen durch einen "Object Identifier" kurz OID. Dieser OID ist Bestandteil eines weltweit hierarchisch verwalteten Adressierungssystems. Hier das passende Beispiel aus dem "NIS" Schema

```
attributetype  1.3.6.1.1.1.1.4
```

Der OID ist eine durch Punkte getrennte dezimale Ziffernfolge. Im Gegensatz zu dem Namen ist der OID immer eindeutig. Ein inoffizielle Liste der OIDs kann unter www.alvestrand.no gefunden werden. Hier ein kleiner Auszug aus der Liste:

OID	***Bedeutung***
1.	Durch die ISO zugewiesen
1.3.6.1	Der Internet OID
1.3.6.1.1.1	RFC2307 NIS Netgroup
1.3.6.1.1.1.1	NISSchema Syntax
1.3.6.1.1.1.1.4	loginShell im NIS Schema

Eigenes Schema entwerfen

Statt ein bestehendes Schema zu ändern sollte man lieber ein eigenes Schema entwickeln. Das hat den Vorteil, das weiterhin die Standard Schemata verwendet werden können, und dass auf anderen Systemen die eigenen Änderungen schnell eingearbeitet werden können. Will man selber ein Schema entwickeln, kann man sich unter der folgenden Adresse einen OID zuweisen lassen:
www.iana.org/cgi-bin/enterprise.pl . Mit diesem Formular erhält man einen OID vom Type 1.3.6.1.4.1.* darunter können nun die eigenen Attribute erstellt und nummeriert werden. Neben einem eigenen OID sollte man für die Namen einen Prefix festlegen, mit dem alle Attributnamen im eigenen Schema beginnen. Will man zum Beispiel ein Schema für die Verwaltung von Tauchgebieten erstellen, kann jedes Attribut mit dem Prefix "stka" beginnen. Beispiele für Attributnamen wären dann "NAME stkaTauchplatzName" oder "NAME stkaMaxTauchTiefe".
Auch können hier eigene Attributtypen und eigene Objektklassen beschrieben werden. Ein Ausführung würde an dieser Stelle allerdings zu weit gehen.

Kommandos zur Verwaltung des LDAP

Zur Verwaltung der Einträge im LDAP stehen verschiedene Kommandos zur Verfügung, mit denen die unterschiedlichsten Aufgaben durchgeführt werden können. Die hier aufgeführten Kommandos sind nicht vollständig. An dieser Stelle soll nur ein genereller Überblick und den Aufbau der Kommandos gegeben werden. Zu allen Kommandos gibt es über "man-pages" im System weitere Hilfen. Bestimmte Parameter sind für alle Kommandos identisch. Hier die wichtigsten Parameter:

Parameter	***Beschreibung***
-x	Es wird ein "simple bind" durchgeführt. Das bedeutet es wird nicht versucht den Benutzer über eine SASL Datenbank zu authentifizieren.
-h <hostname	Auf welchem LDAP Server soll gesucht werden
-D <binddn>	Als welcher Benutzer soll das Kommando ausgeführt werden.
-W	Fragt nach Kommandoeingabe nach dem Passwort des Benutzers
-w <passwort>	Übergibt bei Kommandoausführung das <passwort> für den mit -D angegeben Benutzer
-Z[Z]	Ein einfaches -Z versucht ein Verbindung über TLS zum Server herzustellen. Ein zweites -Z verlangt zwingend eine TLS Verbindung zum Server

Das Kommando "ldapsearch"

Mit dem Kommando "ldapsearch" können Objekte auf Grund bestimmter Attribute im DIT gesucht werden. Wobei alle Objekte aufgelistet werden, die auf das angegebene Suchmuster passen. Wenn kein Suchmuster angegeben wird, verwendet das Kommando "ldapsearch" den default Filter (objektClass=*) in diesem Fall werden alle Objekte im DIT und deren Objektklassen angezeigt. Hier einige Parameter des Kommandos "ldapsearch":

Parameter	*Beschreibung*
-L	Das Suchergebnis wird immer im LDIF-Format ausgegeben. Ein einzelnes "-L" reduziert die Ausgabe auf das ältere LDIFv1 Format, ein zweites "-L" unterdrückt die Kommentare, ein drittes "-L" unterdrückt Angaben zur LDIF-Version
-b <searchbase>	Angabe über den Startpunkt der Suche.
-s base|one|sub	Mit dem Parameter wird die Suchtiefe bestimmt. "base" durchsucht ein bestimmtes Objekt nach einer Eigenschaft, "one" durchsucht ein Ebene des DIT, "sub" durch sucht alle untergeordneten Einträge. Default ist hier "sub"

Hier zwei Beispiele:

```
ldapsearch -x -h localhost -D "cn=manager,dc=home,dc=stka"
"(cn=Stefan Kania)" -W
```

Hier wird der gesamte DIT als Benutzer "Manager" nach einen CN "Stefan Kania" durchsucht. Bevor die Suche beginnt wird nach dem Passwort des Benutzers "Manager" gefragt.

```
ldapseach -x -h localhost -LLL "(UID=stka)" mail cn
```

Hier wird als "anonymer Benutzer" nach einer UID "stka" gesucht. Das Ergebnis wird in stark gekürzter Form ausgegeben. Es werden nur die Attribute "mail" und "cn" angezeigt.

Das Kommando ldapadd

mit dem Kommando "ldapadd" werden neue Objekte zum DIT hinzugefügt. Die Informationen für die Objekte kommen aus einer LDIF-Datei. Hier die wichtigsten Parameter für das Kommando "ldapadd":

Parameter	*Beschreibung*
-c	Fehler werden angezeigt, aber die LDIF-Datei wird weiter abgearbeitet. Normalerweise wird nach einem Fehler die Abarbeitung gestoppt.
-S <Datei>	Fügt Objekte hinzu, die auf Grund von Fehlern übersprungen wurden und in der <Datei> abgelegt sind. Sehr sinnvoll zusammen mit -c
-f <Datei>	Objektinformationen befinden sich in der <Datei>

Hier ein Beispiel:

```
ldapadd -x -h localhost -D "cn=manager,dc=home,dc=stka" -W -f
user.ldif
```

Alle Objekte aus der Datei "user.ldif" werden erzeugt. Zusätzlich gibt es noch ein Kommando "ldapmodify" mit dem bestimmte Attribute von Objekten geändert, hinzugefügt oder entfernt werden können.

Dieser Teil sollte eine Übersicht über die Funktionsweise von LDAP vermitteln, eine vollständige Erklärung aller Möglichkeiten und Optionen ist in der Kürze nicht möglich und auch nicht gewollt.
Nach der Theorie kann es nun mit der Realisierung einer LDAP Umgebung in Form eines Workshops weiter gehen.