Linux Specials

Jens Banning

LDAP unter Linux

Netzwerkinformationen in Verzeichnisdiensten verwalten

ADDISON-WESLEY

Die Deutsche Bibliothek — CIP-Einheitsaufnahme

Ein Titelsatz für diese Publikation ist bei
Der Deutschen Bibliothek erhältlich

Umwelthinweis: Dieses Buch wurde auf chlorfrei gebleichtem Papier gedruckt. Die Einschrumpffolie — zum Schutz vor Verschmutzung — ist aus umweltverträglichem und recyclingfähigem PE-Material.

10 9 8 7 6 5 4 3 2 1

04 03 02 01

ISBN 3-8273-1813-0

Lektorat: Susanne Spitzer, susanne.spitzer@gmx.net
Korrektorat: Friederike Daenecke, Zülpich
Produktion: TYPisch Müller, Arcevia, Italien, typmy@freefast.it
Satz: Hilmar Schlegel, Berlin
Umschlaggestaltung: Hommer DesignProduction, Haar bei München
Druck und Verarbeitung: Media Print, Paderborn

Printed in Germany

Pearson Education

ADDISON-WESLEY

Inhaltsverzeichnis

Pearson Education

ADDISON-WESLEY

Pearson Education

ADDISON-WESLEY

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Computernetzwerke bestehen heutzutage aus einer Vielzahl von Diensten und Anwendungen. Sie werden in der Regel unabhängig voneinander konfiguriert und administriert. Um diese Art der Netzwerkpflege zu vereinfachen, wurden die so genannten Verzeichnisdienste entwickelt. Sie erlauben die zentrale Bearbeitung vieler Dienste und Anwendungen und bieten dem Anwender die Möglichkeit, auf unterschiedlichen Wegen auf die Informationen zuzugreifen.

In diesem Buch werden die Punkte

- Struktur,
- Aufbau und
- Installation

detailliert beschrieben. Sie beziehen sich jeweils auf das Betriebssystem Linux und zeigen dem Anwender praxisnah die Möglichkeiten von Verzeichnisdiensten unter Linux.

Nach den fundamentalen Erläuterungen werden die verschiedenen Programme und Anwendungen, die zum Aufbau von Verzeichnisdiensten dienen, ausführlich beschrieben. Am Ende dieses Buches werden drei umfangreiche Beispielanwendungen vorgestellt. Ihnen folgt ein kurzer Ausblick auf die weitere Entwicklung in dieser Thematik.

Das vorliegende Buch eignet sich durch seine strukturierte Darstellung des Sachverhaltes, sein Glossar und den Index hervorragend als Nachschlagewerk bei der täglichen Arbeit.

Um die Informationen in diesem Buch klarer darzustellen, werden bestimmte Regeln für die Benutzung von Schriftarten konsequent verfolgt:

- Ein- und Ausgaben der Linux-Shell werden in der Schriftart `Typewriter` angegeben.
- Verfahren und Algorithmen sind grundsätzlich *kursiv* dargestellt.
- Eine vom Anwender zu drückende Taste wird **fett** wiedergegeben.
- Menüpunkte in Anwendungen werden in der Form Menüpunkt aufgeführt.

Die Eingabeaufforderung der Linux-Shell hat allgemein die Form:

```
root@host1:~ #
```

für den Benutzer `root`. Sind keine Administratorrechte notwendig, so wird dieses durch den Benutzer `user1` verdeutlicht, was sich im Prompt entsprechend widerspiegelt:

```
user1@host1:~ #
```

Die Rechner in diesem Buch haben die Namen `host1.intern`, `host2.intern`, … . Die zugehörigen IP-Nummern lauten `192.168.17.1`, `192.168.17.2`, … .

Die Grundlagen von Linux sollten dem Leser dieses Werkes vertraut sein.

2 Verzeichnisdienste

In diesem Kapitel werden die Funktionen der Verzeichnisdienste und die dazugehörigen Begriffe erläutert.

2.1 Allgemeines

Bevor die technische Realisierung von Verzeichnisdiensten beschrieben werden kann, müssen zunächst zwei grundlegende Fragen zu dieser Thematik geklärt werden:

- Was sind Verzeichnisdienste?
- Welchen Nutzen haben Verzeichnisdienste?

Was sind Verzeichnisdienste? Bei Verzeichnisdiensten handelt es sich um eine im Netzwerk verteilte Datenbank, die auf dem Client/Server-Prinzip basiert. In ihr können nahezu beliebige Informationen festgehalten werden. Diese Informationen können ferner dazu verwendet werden, um verschiedene Anwendungen zu konfigurieren und zu administrieren.

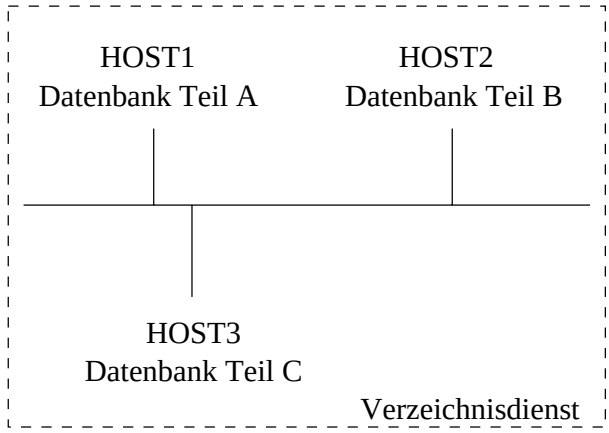

Abbildung 2.1: Verteilte Daten bei Verzeichnisdiensten

Dabei können die gewünschten Daten

- flexibel,
- dynamisch und
- sicher

aufbewahrt werden. Das bedeutet, dass die Struktur der Daten in Verzeichnisdiensten vom Anwender bestimmt werden kann. Es bestehen keine Vorgaben.

Ferner können die Informationen im laufenden Betrieb verändert werden. Letztlich ist es möglich, Sicherheitsregeln zu definieren, die die Zugriffsmöglichkeiten der Anwender auf die Informationen im Verzeichnisdient beschreiben.

Der Zugriff auf die Daten eines Verzeichnisdienstes erfolgt hauptsächlich lesend, da viele Anwendungen die bestehenden Daten lediglich nutzen. Schreibende Zugriffe sind relativ selten, da sie nur notwendig sind, wenn der Administrator den Datenbestand verändern oder erweitern möchte. Wenn man diesen Tatbestand berücksichtigt, bieten die Verzeichnisdienste im Hinblick auf den Zugriff eine wesentlich höhere Geschwindigkeit als andere Datenbanken. Dieses rührt daher, dass bei den üblichen Datenbanken das Verhältnis von Lese- zu Schreibanforderungen nicht bestimmt werden kann.

Welchen Nutzen haben Verzeichnisdienste? Als Basis des Verzeichnisdienstes dient eine im Netzwerk verteilte Datenbank (siehe Abbildung 2.1). Dabei können unterschiedliche Teile auf verschiedenen Rechnern im Netzwerk abgelegt werden. Ferner kann die Datenbank auch repliziert werden (siehe Abbildung 2.2). Dadurch erhöht sich die Ausfallsicherheit, weil die gewünschten Informationen beim Wegfall eines Rechners weiterhin verfügbar sind.

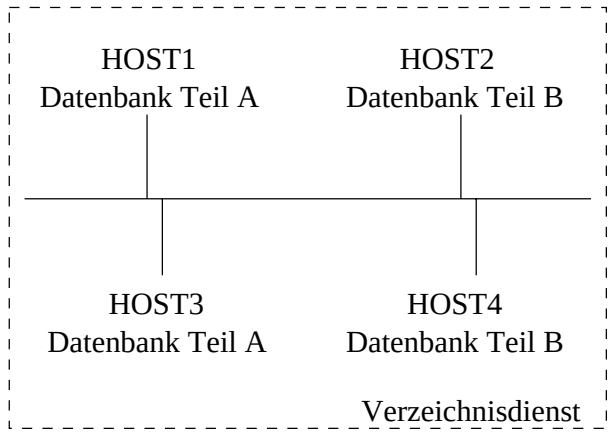

Abbildung 2.2: Ausfallsicherheit bei Verzeichnisdiensten

Ferner kann auf die Informationen von jedem Rechner im angeschlossenen Netzwerk zugegriffen werden. Gerade dieser Punkt ist ein wesentlicher Vorteil der Verzeichnisdienste. So können Client/Server-Anwendungen wie zum Beispiel das Network File System (NFS) von jedem beliebigen Client-Rechner im Netzwerk auf dem Server administriert werden. Außerdem können die Daten des Verzeichnisdienstes auf jedem Client zu dessen Konfiguration verwendet werden.

Der Zugriff erfolgt mit dem TCP/IP-Protokoll (siehe Abbildung 2.3).

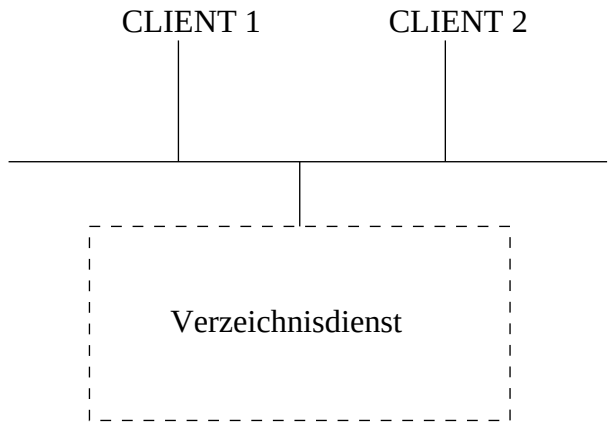

Abbildung 2.3: Zugriff auf Verzeichnisdienste

2.2 Aufbau

Nachdem die Verzeichnisdienste nun allgemein beschrieben wurden, befasst sich dieser Abschnitt mit dem Aufbau der Datenbank und der Informationen, die sich darin befinden. In diesem Zusammenhang sollen zunächst zwei Begriffe eingeführt werden.

- Verzeichnisdienste werden generell als DS bezeichnet. DS steht für den englischen Begriff Directory Services.
- Der Aufbau der Directory Services erfolgt prinzipiell nach dem so genannten X.500-Standard. Er wird im Folgenden kurz skizziert.

Bei der DS-Datenbank handelt es sich um eine baumartige Struktur. In ihr werden die gewünschten Informationen abgelegt. Der Baum (englisch Tree) hat grundsätzlich eine Wurzel (englisch Root). Unter ihr können weitere Strukturen ergänzt werden, die ihrerseits wiederum Unterstrukturen besitzen können. Der so resultierende Baum kann damit die Gegebenheiten des Netzwerkes beliebig repräsentieren.

Im dem Beispiel aus Abbildung 2.4 wird unter `Root` die Struktur mit dem Namen `Intern` angelegt. Darunter existieren zwei Bereiche (`Einkauf` und `Verkauf`), in denen sich einige Benutzer und einige Rechner befinden.

Die Einträge in dieser Datenbank haben natürlich Eigenschaften, die weitere Informationen enthalten. Betrachten wir nun die Einträge `Benutzer FlemmingJ` und `Rechner Host1` genauer (siehe Abbildung 2.5).

Hinter dem Anwender `FlemmingJ` steht demnach die Person `Jan Flemming`. `FlemmingJ` könnte zum Beispiel der Login-Name sein. Der Anwender ist ferner `Abteilungsleiter` und hat die Telefondurchwahl `123`.

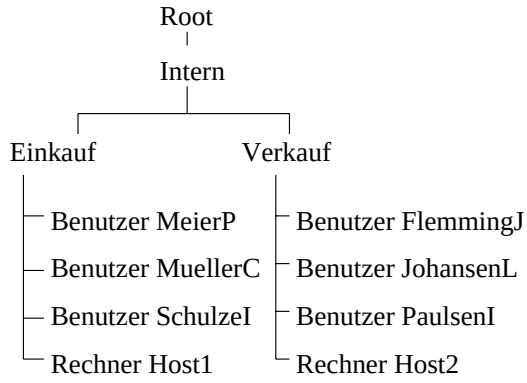

Abbildung 2.4: Beispiel eines Verzeichnisdienstes

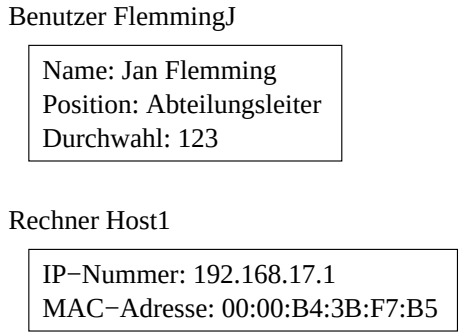

Abbildung 2.5: Eigenschaften der Verzeichnisdienst-Einträge

Der aufgeführte Rechner hat die IP-Nummer `192.168.17.1` und die MAC-Adresse `00:00:B4:3B:F7:B5`.

Auf diese Weise lassen sich zu den bestehenden Einträgen nahezu beliebig viele Eigenschaften definieren, die den Eintrag näher beschreiben. Diese Flexibilität ermöglicht es auch, neue Anwendungen zu entwickeln, die mit den Directory Services verwaltet werden können.

Da die Struktur der Directory Services somit nahezu beliebig angelegt werden kann, ist bezüglich der möglichen Informationen, die repräsentiert werden können, keinerlei Beschränkung vorhanden. So wäre es zum Beispiel möglich, die Zimmerbelegung eines Hotels in einem Verzeichnisdienst abzulegen (siehe Abbildung 2.6).

In der obersten Ebene unter der Wurzel ist das Hotel vorhanden. Es ist weiterhin in zwei Etagen strukturiert, die zum einen die Zimmer 101 und 102 und zum anderen die Zimmer 201 und 202 besitzen. Als Eigenschaft zu den Zimmereinträgen werden die

aktuellen Gäste erwähnt. Zu den Zimmern können nun natürlich auch zum Beispiel Telefonnummern, Preise und andere Informationen festgehalten werden.

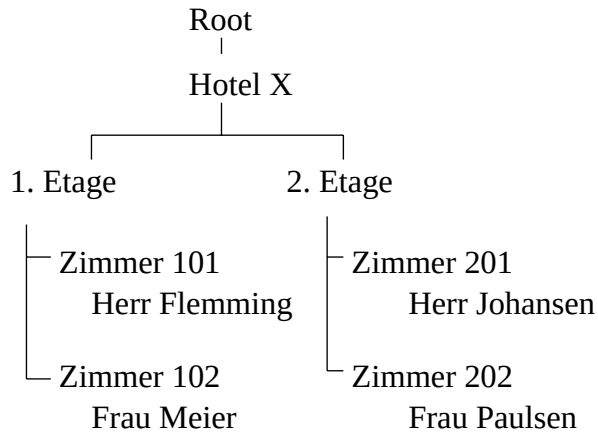

Abbildung 2.6: Beispiel eines Verzeichnisdienstes

Zusammenfassend können bezüglich des Aufbaus eines Verzeichnisdienstes die folgenden Punkte festgehalten werden:

- Die DS-Datenbank ist baumartig aufgebaut.
- Es existiert immer eine Wurzel.
- In der Baumstruktur existieren Einträge.
- Die Einträge haben Eigenschaften.

Im vorigen Abschnitt wurde erwähnt, dass die gesamte DS-Datenbank verteilt auf mehreren Rechnern abgelegt werden kann. Die Aufteilung der Daten erfolgt dabei anhand der strukturellen Einträge. So können bei dem Hotelbeispiel die Daten unterhalb der ersten Etage und die unterhalb der zweiten Etage auf unterschiedlichen Rechnern abgelegt sein. Auf wiederum einem dritten Rechner könnte ferner die gesamte Baumstruktur abgelegt sein (siehe Abbildung 2.7).

Auf diese Weise ist die Datenbank sowohl verteilt als auch repliziert. Für den Anwender ist dieses Verfahren jedoch absolut transparent. Er befragt lediglich einen Verzeichnisdienst. Wie und wo dessen Datenbestand aufbewahrt wird, bleibt ihm verborgen. Er profitiert lediglich von der erhöhten Ausfallsicherheit.

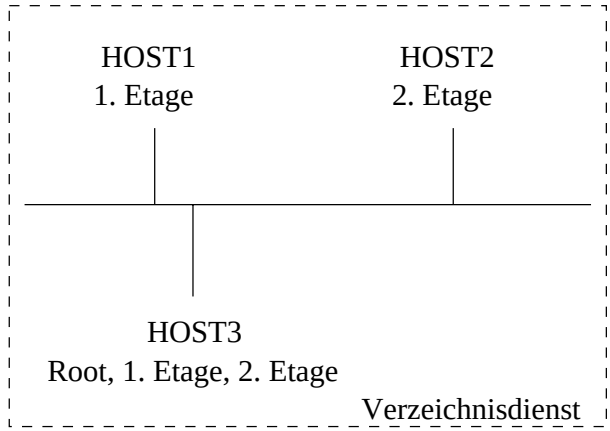

Abbildung 2.7: Aufteilung der Verzeichnisdienst-Daten

2.3 Funktion

Aus den zuvor genannten Merkmalen und Strukturen, die Verzeichnisdienste generell aufweisen, ergeben sich zwei Funktionen, die mit ihnen wahrgenommen werden können:

- Speichern von Informationen
- Verwenden der Informationen

Speichern von Informationen. Die triviale Aufgabe einer Datenbank ist die Speicherung von Informationen. Directory Services bieten dabei die Möglichkeit,

- Daten strukturiert abzuspeichern und
- Daten schnell zur Verfügung zu stellen.

Die strukturierte Speicherung der Daten erfolgt in der bereits bekannten Baumstruktur. Damit die Daten im Falle eines lesenden Zugriffs dem Anwender auch schnell zur Verfügung gestellt werden können, sollte es sich insgesamt um nicht allzu große Datenbestände handeln. Allgemeine Beschränkungen können an dieser Stelle nicht angegeben werden, da sie sehr von der verwendeten DS-Software abhängen.

Auf die so im Directory Services Tree abgelegten Informationen kann anschließend von jedem Client aus, der an das entsprechende Netzwerk angeschlossen ist, zugegriffen werden. In einem solchen Fall würde der Verzeichnisdienst lediglich als Informationsquelle benutzt.

Verwenden der Informationen. Ein weitaus wichtigerer Aspekt ist die direkte Verwendung der gegebenen Informationen zur Konfiguration anderer Dienste und Anwendungen. Es ist zum Beispiel wünschenswert, dass sich die im Verzeichnisdienst angegebenen Benutzer mit dem dort aufgeführten Namen am System anmelden können. Ferner ist auch eine Passwort-Verifizierung am Verzeichnisdienst wünschenswert. Ebenfalls sollte es möglich sein, andere Netzwerkdienste wie zum Beispiel das Network File System (NFS) für mehrere Rechner zentral konfigurieren zu können. Letzteres wird im Abschnitt 5.2 auf Seite 193 beschrieben.

In den Abbildungen 2.8 und 2.9 sind alle Daten des Verzeichnisdienstes auf dem Rechner `HOST1` abgelegt. Es existiert weder eine Unterteilung noch eine Replizierung der Informationen. Der Rechner `HOST2` übernimmt in dieser Konstellation den

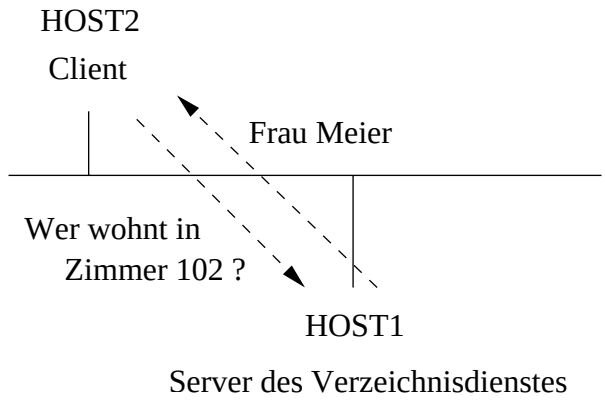

Abbildung 2.8: Client-Zugriff auf einen Verzeichnisdienst

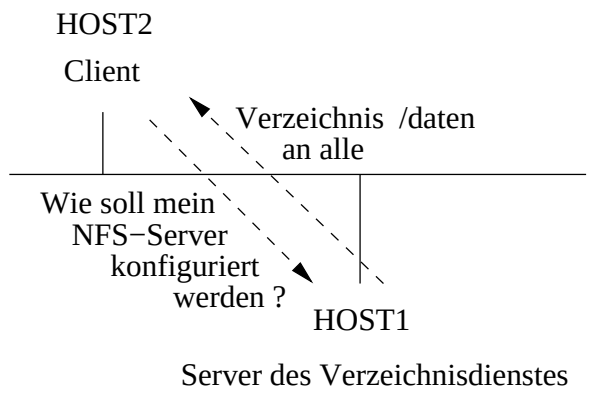

Abbildung 2.9: Client-Zugriff auf einen Verzeichnisdienst

Part des Clients, indem er den DS-Server zum einen befragt und zum anderen die gewonnenen Informationen zu seiner eigenen Konfiguration verwendet.

Directory Services bieten dem Anwender die Möglichkeit, auf die Informationen wie in den obigen Beispielen

- zuzugreifen und sie
- anderweitig weiterzuverwenden.

Was genau an Daten für die Konfiguration anderer Dienste und Anwendungen benutzt werden kann, hängt davon ab, welche der folgenden Zugriffsvarianten auf den DS möglich sind:

- Grafische Anwendungen
- Kommandos
- Funktionen
- Routinen

Grafische Anwendungen. Für die breite Akzeptanz der Directory Services unter Linux ist es sicherlich wichtig, grafische Anwendungen zur Verfügung zu haben, um auf die gegebenen Informationen zugreifen zu können. Da Verzeichnisdienste im Aufbau und auch im Zugriff standardisiert sind, ist es auch möglich, mit anderen Produkten und anderen Systemen auf die DS-Daten Zugriff zu bekommen. Ferner können die Daten oft auch mit herkömmlichen Internet-Browsern eingesehen werden.

Kommandos. Grafische Anwendungen haben den Nachteil, dass die auf diesem Wege erlangten Informationen des Verzeichnisdienstes nicht weiterverwendet werden können. So wäre es wünschenswert, die Daten mittels geeigneter Scripts in andere Konfigurationen einfließen zu lassen. Dazu ist es notwendig, dass Kommandos zur Verfügung stehen, die den scriptgesteuerten Zugriff in der Textkonsole erlauben.

Funktionen. Um auch neu zu entwickelnde Programme und Anwendungen auf DS-Daten basieren zu lassen, sind Schnittstellen erforderlich, die es dem Programmierer ermöglichen, aus seiner Umgebung heraus die Daten des Directory Service zu benutzen. Denkbar wären hier zum Beispiel Funktionen für die Programmiersprache C.

Routinen. Routinen haben die gleiche Funktionalität wie Funktionen. Jedoch handelt es sich hierbei um Makropakete, die die bereits genannte Funktionalität realisieren.

3 Das Lightweight Directory Access Protocol LDAP

3.1 Funktion

Im vorigen Kapitel wurden Verzeichnisdienste allgemein beschrieben. Diese Angaben sollen in diesem Kapitel technisch spezifiziert werden.

Verzeichnisdienste bestehen aus zwei grundlegenden Teilen:

- Datenbank
- Zugriff auf die Datenbank

Die Datenbank basiert auf dem so genannten X.500-Standard. In ihm ist der gesamte Aufbau des Dienstes beschrieben. In der Anfangsphase der Verzeichnisdienste erfolgte der Zugriff auf die X.500-konformen Daten mit dem Directory Access Protocol (DAP). Dieses Protokoll war jedoch sehr schwierig zu implementieren und fand aufgrund seiner Komplexität wenig Anklang. Letztlich konnte es sich nicht durchsetzen.

Eine vereinfachte Form des DAP ist das Lightweight Directory Access Protocol (LDAP). Es ist heutzutage der Standard, wenn es um den Zugriff auf X.500-Daten geht, und bietet die Möglichkeiten:

- Es basiert auf dem TCP/IP-Protokoll
- Es ist leicht zu implementieren
- Es ist funktionell und schnell

Basiert auf dem TCP/IP-Protokoll. Das Protokoll LDAP (Lightweight Directory Access Protocol) basiert auf dem TCP/IP-Protokoll. Da nahezu alle Netzwerke IP-basiert arbeiten, ist die Kommunikation zwischen dem Client und dem Server der Directory Services in der Regel gewährleistet. LDAP nutzt demnach die bestehenden Netzwerkverbindungen.

Leicht zu implementieren. Da die TCP/IP-Protokollfamilie zur Anwendung kommen kann, ist die Umsetzung von LDAP in Software-Produkten relativ einfach.

Funktionell und schnell. Das Directory Access Protocol (DAP) zeichnete sich durch einen sehr umfangreichen Protokollaufbau aus. Dieses ging auf Kosten der Geschwindigkeit bei der Kommunikation mit DAP. Beim Lightweight Directory Access Protocol ist der Protokollaufbau entsprechend vereinfacht, so dass die Kommunikation schneller durchgeführt werden kann. Trotz der Schnelligkeit realisiert es natürlich alle relevanten Funktionen, die zum Zugriff auf die X.500-Daten notwendig sind.

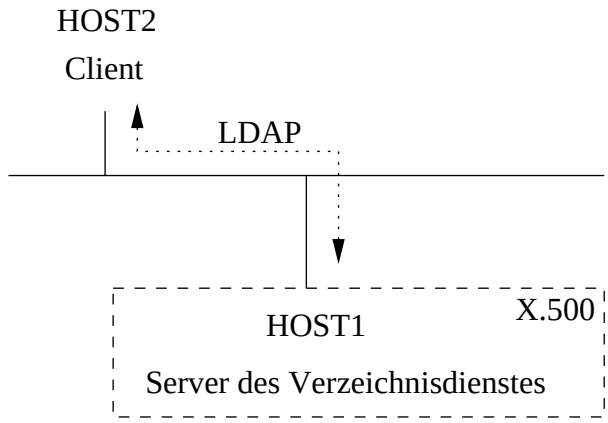

Abbildung 3.1: Zusammenhang zwischen X.500 und LDAP

Bei LDAP handelt es sich um ein klassisches Client/Server-Modell. Der Client stellt seine Anfragen an den LDAP-Server. Dieser führt die Anfragen auf der X.500-Datenbank aus und liefert die Ergebnisse entsprechend zurück (siehe Abbildung 3.2). Die Kommunikation zwischen Client und Server bedarf einer genaueren Betrachtung.

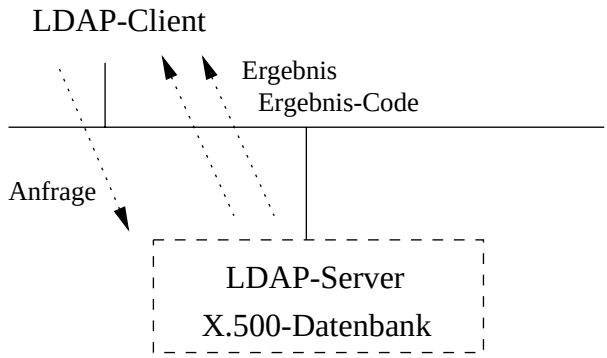

Abbildung 3.2: Kommunikation zwischen LDAP-Client und -Server

Die Daten, die per LDAP zwischen Server und Client ausgetauscht werden, sind mitteilungsorientiert. Das bedeutet, dass der Client seine Anfrage in einer LDAP-Mitteilung zusammenfasst und sie an den Server sendet. Der Server sendet die Ergebnis-Daten ebenfalls als Mitteilung (englisch Message) an den Client. Ferner wird ein Ergebnis-Code zum Client gesendet.

Es ist ferner möglich, dass der LDAP-Client mehrere Anfragen gleichzeitig sendet, ohne für die vorangegangenen bereits eine Antwort erhalten zu haben. Dieses ist ein Unterschied zu anderen Protokollen.

So kann eine Anfrage mit dem HyperText Transfer Protocol (HTTP) erst gesendet werden, wenn der Server eine vorherige Anfrage bereits beantwortet hat. LDAP-Anfragen sind von den Antworten unabhängig und werden jeweils mit einer gesonderten Kennung markiert, um die späteren Antworten wieder zuweisen zu können (siehe Abbildung 3.3).

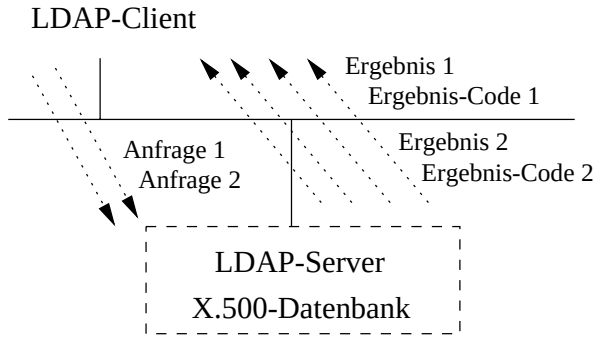

Abbildung 3.3: Gleichzeitiges Senden von LDAP-Anfragen

Bisher wurde immer angenommen, dass Anfragen zu genau einer Antwort führen. Es ist jedoch auch möglich, dass eine Anfrage mehr als eine Antwort zur Folge hat. Die Ergebnisse werden dann nacheinander in Mitteilungen an den Client gesendet (siehe Abbildung 3.4).

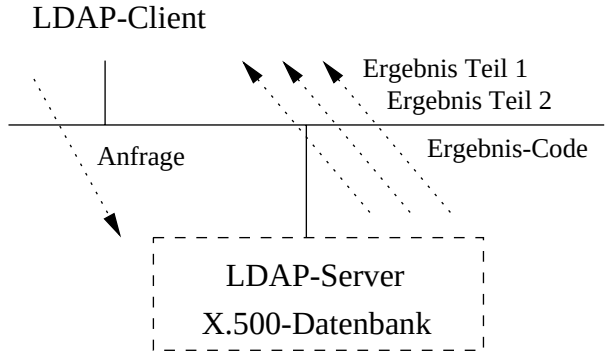

Abbildung 3.4: LDAP-Antworten mit mehreren Einträgen

3.2 Aufbau und Eigenschaften

Wie die LDAP-Mitteilungen aufgebaut sind und welche Eigenschaften sie haben, wird im Folgenden betrachtet. LDAP-Aktionen können in drei Kategorien eingeteilt werden:

- Beglaubigung
- Lesender Zugriff
- Schreibender Zugriff

Beglaubigung. Die wohl wichtigste Aktion, die der LDAP-Client durchführt, ist die Beglaubigung an der X.500-Datenbank. In der Regel sendet der Client dazu einen Benutzernamen und ein Passwort an den Server, wobei natürlich auch andere Beglaubigungsverfahren möglich sind. Sofern die Überprüfung von Benutzer und Passwort erfolgreich war, stehen dem Anwender anschließend die für ihn definierten Zugriffe zur Verfügung.

Lesender Zugriff. Die meisten Zugriffe auf X.500-Daten erfolgen in der Regel in der lesenden Form. Diese unterteilen sich in

- suchen und
- vergleichen.

Es kann also auch ein gegebener Wert mit dem der DS-Datenbank verglichen werden.

Schreibender Zugriff. Um die Daten des Directory Service zu verändern, ist es notwendig, dass der Anwender die nötigen Schreibrechte besitzt. Es ist dann möglich, das

- Hinzufügen,
- Löschen,
- Ändern und
- Umbenennen

von Einträgen durchzuführen.

Unter Berücksichtigung der zuvor genannten Aktionen kann der Datenaustausch zwischen LDAP-Client und LDAP-Server in die Schritte der Tabelle 3.1 aufgegliedert werden:

1. Zu Beginn sendet der Client dem Server die zur Beglaubigung an der X.500-Datenbank notwendigen Parameter zu. Hierbei handelt es sich in der Regel um einen Benutzernamen und ein Passwort.
2. Der LDAP-Server verifiziert den Parameter am Verzeichnisdienst und schickt dem Client das Ergebnis der Beglaubigung zu.
3. Falls der Benutzername und das Passwort korrekt waren, ist der Client nun mit dem Server verbunden. Er kann jetzt seine Anfragen senden, um Daten zu lesen oder zu bearbeiten. Anhand des Benutzernamens wird entschieden, welche Aktivitäten durchgeführt werden können.

Schritt	Aktion	Richtung
1	Beglaubigungswunsch	Client ⇒ Server
2	Ergebnis der Beglaubigung	Client ⇐ Server
3	Senden der Anfrage	Client ⇒ Server
4	Ergebnis-Daten der Anfrage	Client ⇐ Server
5	Ergebnis-Code der Anfrage	Client ⇐ Server
6	Verbindungsende signalisieren	Client ⇒ Server
7	Verbindung beenden	Client ⇐ Server

Tabelle 3.1: Datenaustausch zwischen LDAP-Client und -Server

4. Der Server sendet dann in gewohnter Weise das Ergebnis oder die Ergebnisse der Client-Anfrage.
5. Er sendet ebenfalls den bereits bekannten Ergebnis-Code.
6. Danach signalisiert der Client, dass er die Verbindung beenden möchte.
7. Der Server bestätigt dem Client abschließend das Ende der Verbindung.

Während eine Verbindung zwischen dem Client und dem Server besteht, gelten beide Seiten als miteinander verbunden. Dieses wird in der Regel mit dem englischen Begriff bind bezeichnet.

Nachdem beschrieben wurde, was das Lightweight Directory Access Protocol im Einzelnen leistet und wie die Kommunikation zwischen den Komponenten

- LDAP-Client,
- LDAP-Server und
- X.500-Datenbank

stattfindet, sollen nun kurz die zugrunde liegenden Protokolle betrachtet werden.

Die Kommunikation mit dem Protokoll LDAP erfolgt in IP-Netzen mit dem Transmission Control Protocol (TCP). Dieses Protokoll hat die folgenden Eigenschaften:

- Es stellt eine bidirektionale Verbindung zwischen Client und Server her.
- Es erfolgt eine verbindungsorientierte Übertragung der Daten. Auf diese Weise wird gewährleistet, dass gesendete Daten beim Empfänger ankommen, da dieser eine Quittung zum Sender zurückschickt. Trifft die Quittung beim Sender nicht ein, so werden die Daten erneut übertragen.
- Die Daten werden mit einer Prüfsumme versehen, um ihre Integrität zu sichern.

- Der Verbindungsaufbau und das Beenden der Verbindung erfolgen im gegenseitigen Einvernehmen. Das heißt, die jeweils andere Partei muss den entsprechenden Wunsch bestätigen.
- Ferner erfolgt die Datenübertragung nicht nur zu der Ziel-Adresse, sondern dort auch direkt zu einem entsprechenden Dienst. Die Zuweisung der Dienste erfolgt dabei über die TCP-Ports.

Weitere Einzelheiten zum Protokoll TCP sind in [2] nachzulesen.

Die Kommunikation zwischen dem LDAP-Client und dem LDAP-Server erfolgt über den TCP-Port 389. Der LDAP-Server wartet demnach auf Client-Anfragen an genau diesem Port. Die beiden Kommunikationspartner verständigen sich dabei grundsätzlich per Unicast. Der Server wird somit direkt über seine IP-Nummer oder seinen Namen angesprochen und antwortet auf dem entsprechend umgekehrten Wege.

3.3 Der LDAP-Server

Das zentrale Element ist der LDAP-Server, der im Folgenden beschrieben wird. Dabei wird angenommen, dass er ebenfalls die X.500-Datenbank betreut.

3.3.1 Die Datenhaltung

Die Daten sind in Verzeichnisdiensten baumartig strukturiert. Der Baum beginnt grundsätzlich mit dem Element Root (Wurzel). Unterhalb dieses Elements können die bereits im letzten Kapitel genannten Strukturen angelegt werden. Ferner können zu jedem Eintrag Eigenschaften festgelegt werden. Die Struktur der Daten wird detailliert in Abschnitt 3.3.2 auf Seite 17 beschrieben.

In diesem Abschnitt geht es um die Daten an sich. Dabei sind vor allem zwei Fragen zu klären:

- Welche Daten sollen gespeichert werden?
- Welches Format haben die Daten?

Welche Daten sollen gespeichert werden? Die Frage nach dem Umfang der zu speichernden Daten ist längst nicht so trivial, wie sie zunächst auf Anhieb erscheint. Folgende Fragen müssen geklärt werden:

- Welche Daten sind zur Umsetzung der gewünschten Funktionalität notwendig?
- Welche Daten sind sinnvoll, aber nicht zwingend erforderlich? Gerade diesem Punkt sollte besondere Aufmerksamkeit gewidmet werden. Zu viele Daten erschweren und verlangsamen die Suche im gesamten Verzeichnisdienst.

- Wie können die Daten aktuell gehalten werden? Hier ist zu entscheiden, wer welche Daten pflegt und ob für die Administration mitunter Automatismen wie Scripts usw. verwendet werden können.

Welches Format haben die Daten? Nachdem der Datenumfang nun bekannt ist, muss man die gewünschten Informationen bezüglich

- Format und
- Anzahl

näher spezifizieren. Hier ist zu entscheiden, wie viel Speicherplatz jedes Datenelement einnimmt. Eine Telefonnummer könnte zum Beispiel aus zwölf Zeichen (Bytes) bestehen. Nun ist zu ermitteln, wie viele dieser Datenelemente voraussichtlich in der Datenbank auftauchen. Wird die Berechnung Anzahl × Format für alle Datenelemente durchgeführt, so lässt sich in der Summe die Größe der X.500-Datenbank bestimmen. Sie ist in etwa ein Maß für die Geschwindigkeit, mit der Anfragen ausgeführt werden. Es sei nicht verschwiegen, dass wenige schlecht organisierte Daten mitunter eine schlechtere Perfomance haben als viele gut organisierte. Näheres dazu erfahren Sie ebenfalls in Abschnitt 3.3.2.

Ferner ist durch die Berechnung Anzahl × Format zu erkennen, wie viel Plattenplatz und Hauptspeicher für den Server ungefähr notwendig sind.

Zur Planung der Datenhaltung gehören im Prinzip auch noch die Punkte Partitionierung und Replizierung. Bei der Partitionierung handelt es sich um eine Aufteilung der DS-Datenbank. Die einzelnen Teile sind dann auf unterschiedlichen Servern abzulegen. Genauere Informationen zur Partitionierung und deren Gründe sind in Abschnitt 3.3.5 auf Seite 35 aufgeführt. Bei der Replizierung (3.3.4, Seite 32) handelt es sich um das redundante Speichern von Datenbeständen auf mehreren Rechnern im Netzwerk.

3.3.2 Das Schema

Nachdem Sie entschieden haben, welche Daten mit welchem Format zu speichern sind, müssen die Informationen in einer Struktur abgebildet werden. Die Struktur der Datenbank eines Directory Service wird mit dem Begriff Schema bezeichnet. Das Schema umfasst alle möglichen Einträge im DS-Tree sowie deren Eigenschaften in der strukturellen Darstellung.

Es gliedert sich in die folgenden Punkte:

- Objekte
- Attribute
- Regeln

Objekte. Die Einträge in der Baumstruktur werden als Objekte bezeichnet.

Attribute. Die Eigenschaften der Objekte heißen Attribute (siehe Abbildung 3.5).

Regeln. Welches Objekt an welcher Stelle im DS-Tree stehen darf und welche Attribute es aufweisen kann bzw. muss, wird mit diversen Regeln im Schema beschrieben.

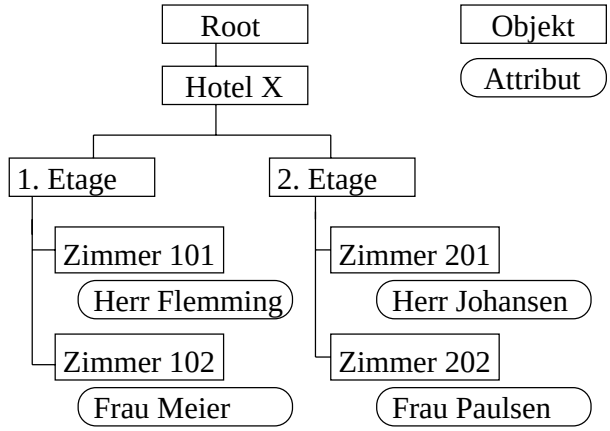

Abbildung 3.5: Objekte und Attribute

Objekte

Die Baumstruktur eines Verzeichnisdienstes (sie wird auch als Directory Information Tree (DIT) bezeichnet) besteht im Wesentlichen aus den Objekten. Sie werden in so genannten Objektklassen beschrieben. Diese zentralen Einträge des DIT haben die folgenden Aufgaben und Eigenschaften:

- Eindeutiger Name
- Eindeutige Identifikation
- Art des Objektes
- Notwendige Attribute
- Mögliche Attribute

Eindeutiger Name. Jede Objektklasse wird durch einen im gesamten Verzeichnisdienst eindeutigen Namen repräsentiert. So könnte zum Beispiel eine Klasse mit dem Namen `person` existieren. Im DIT können dann Objekte vom Typ dieser Klasse `person` stehen. `FlemmingJ` wäre ein Beispiel dafür. Bei den Namen der Objekte und Objektklassen ist darauf zu achten, dass diese für gewöhnlich nicht

case-sensitive sind. Zwischen Groß- und Kleinschreibung wird also in der Regel nicht unterschieden.

Eindeutige Identifikation. Der Name einer Objektklasse dient im Prinzip nur dazu, die entsprechenden Elemente leichter verwenden zu können. Identifiziert werden die Klassen anhand einer eindeutigen Kennung, der Object Identifier (OID). Diese nummerische Kennung führt als Synonym den zuvor genannten Namen.

Notwendige Attribute. In der Objektklasse wird ebenfalls definiert, welche Eigenschaften ein Objekt von dieser Klasse im DIT aufweisen muss.

Mögliche Attribute. Neben den zwingend notwendigen Eigenschaften, ist es jedoch auch möglich, eine Reihe weiterer Attribute anzugeben.

In der Baumstruktur des Directory Service können Objekte in zwei Bereiche eingeteilt werden:

- Containerobjekte
- Blattobjekte

Containerobjekte. Als Container werden Objekte bezeichnet, die ihrerseits andere Objekte enthalten können. Ein Beispiel für einen Container wäre die Abteilung `Einkauf`.

Blattobjekte. Entsprechend werden die Enden der Verzweigungen in der Baumstruktur als Blätter bezeichnet. Das Objekt `FlemmingJ` ist ein Blattobjekt unterhalb von `Einkauf` (siehe Abbildung 3.6).

Die Abbildung 3.6 verdeutlicht den Zusammenhang zwischen Containern und Blattobjekten.

Bei den Objekten `Intern`, `Einkauf` und `Verkauf` handelt es sich um Container, da sie unter sich weitere Einträge aufweisen. `Root` kann im Prinzip auch als Container bezeichnet werden. Bei den verbleibenden Objekten (`Benutzer MeierP` usw.) handelt es sich um Blätter des DIT.

Container lassen sich in vier verschiedene Klassen einteilen. Da sich die Abkürzungen dieser Klassen auf die englischen Bezeichnungen beziehen, werden diese an dieser Stelle auch verwendet.

- Root
- Country (`c`)
- Organization (`o`)
- Organizational Unit (`ou`)

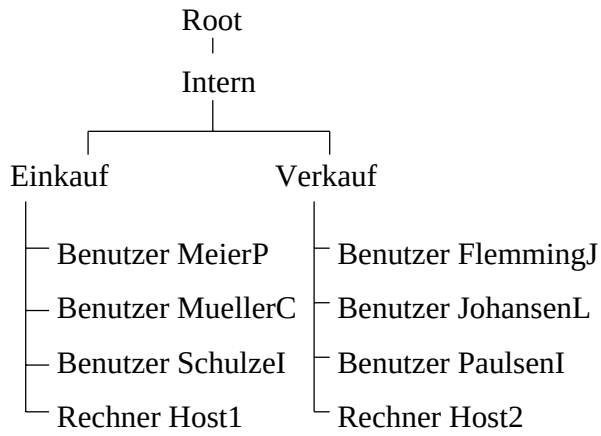

Abbildung 3.6: Container und Blattobjekte

Die Wurzel des Verzeichnisbaumes existiert grundsätzlich als fester Bestandteil eines DIT. Der Container Country kann verwendet werden, um ein Land anzugeben. Durch die Klasse Organization wird eine Einrichtung, eine Firma o. Ä. repräsentiert. Zur weiteren Unterteilung sind organisatorische Einheiten möglich. Sie könnten zum Beispiel eine Abteilung beschreiben. Blattobjekte werden generell als

- Common Name (`cn`)

bezeichnet.

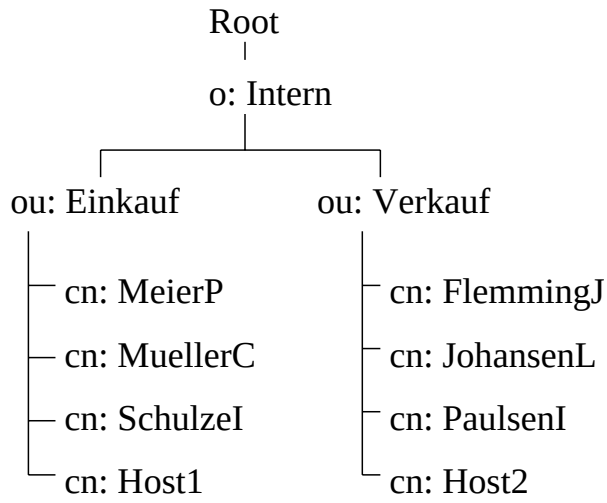

Abbildung 3.7: Container und Blattobjekte

Objekte im DIT können grundsätzlich mehreren Objektklassen angehören. Als Attribute eines so erstellten Objektes können dann die Eigenschaften aller verwendeten Klassen benutzt werden. Dabei addieren sich die notwendigen und die optionalen Eigenschaften. Nehmen wir an, dass zwei Objektklassen mit den Namen `person` und `host` existieren, mit den Eigenschaften, die in Tabelle 3.2 und 3.3 angegeben sind.

Attribut	Art
`Name`	notwendig
`Telefon`	optional

Tabelle 3.2: Objektklasse `person`

Attribut	Art
`Name`	notwendig
`IP-Nummer`	optional

Tabelle 3.3: Objektklasse `host`

Ein Objekt, welches beiden Klassen angehört, hat die Eigenschaften, die in Tabelle 3.4 aufgeführt sind.

Die Verwendung von Objekten, die aus mehreren Klassen stammen, ist ein durchaus gängiges Mittel, wenn es darum geht, verschiedenartige Elemente zu vereinen. Um wie im obigen Beispiel ein Objekt zu definieren, das einen Benutzer und den dazugehörigen Rechner beschreibt, können die bestehenden Klassen des Schemas einfach kombiniert werden.

Attribut	Art
`Name`	notwendig
`IP-Nummer`	optional
`Telefon`	optional

Tabelle 3.4: Objekt der Klassen `person` und `host`

Attribute

Welche Attribute in einem Objekt benutzt werden können und welche für die Erstellung des Objekts zwingend erforderlich sind, ist in den Definitionen der Objektklassen festgelegt. Auch die Attribute haben eine definierte Struktur und vorgegebene Aufgaben:

- Eindeutiger Name
- Eindeutige Identifikation
- Syntax
- Werte des Attributs

Eindeutiger Name. Genau wie Objekte werden auch Attribute durch einen eindeutigen Namen repräsentiert.

Eindeutige Identifikation. Der Name dient lediglich als Synonym für eine Identifikation.

Syntax. Attributen können Werte übergeben werden. Welcher Syntax diese Werte unterliegen, ist ebenfalls fest definiert. So könnte der Eigenschaft `Telefon` ein Wert mit maximal zwölf Bytes (zum Beispiel `1234`) übergeben werden. Ferner kann definiert werden, ob das Attribut case-sensitive sein soll und welche Zeichen es enthalten darf. Auf diese Weise können Buchstaben bei einer Telefonnummer generell unterbunden werden.

Werte des Attributs. Bei der Telefonnummer ist in der Regel nur ein Wert anzugeben. Doch es ist durchaus möglich, dass ein Attribut mehrere Werte annimmt. Falls zum Beispiel eine Eigenschaft `Qualifizierung` existiert, so kann sie die beiden Werte `Studium` und `Ausbildung` enthalten (siehe Tabelle 3.5). Auch dieses ist im Schema definiert.

Attribut	Wert
`Qualifizierung`	`Studium, Ausbildung`

Tabelle 3.5: Beispiel für ein Attribut mit mehreren Werten

Einige LDAP-Implementierungen verfolgen das Mittel der Attributhierarchie. Dabei sind bestimmte Eigenschaften Untertypen von anderen Attributen. So könnte eine Eigenschaft `Vorname` existieren, die vom Attribut `Name` abgeleitet ist (siehe Abbildung 3.8).

Auf die Pflege der Datenbank des Verzeichnisdienstes hat diese Hierarchie keine Auswirkung. Lediglich für Anwender, die Suchaktionen durchführen, ist hier eine gewisse Vereinfachung festzustellen. Fragt ein Anwender nach dem Attribut `Name`, so wird ihm auch die Eigenschaft `Vorname` mitgeteilt.

Die Attributhierarchie wird in diesem Buch nicht weiter betrachtet, da sie im Allgemeinen nicht implementiert ist.

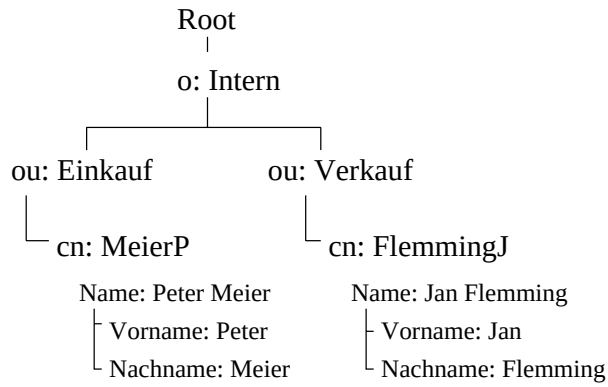

Abbildung 3.8: Attributhierarchie

Regeln

Bei der Erstellung eines Verzeichnisdienstes sind einige Regeln zu beachten. Durch sie wird definiert, an welcher Stelle im Directory Information Tree Container- und Blattobjekte stehen dürfen.

1. Als oberstes Element eines Verzeichnisdienstes existiert das Objekt Root. Es ist grundsätzlich vorhanden und kann weder verschoben noch verändert werden.
2. Unterhalb von Root ist es möglich, entweder ein Country-Objekt (`c`) oder eine Organisation (`o`) anzulegen.
3. Das Country-Objekt muss nicht existieren.
4. Das Country-Objekt darf nur einmal existieren.
5. Die Organisation muss direkt unter dem Country-Objekt stehen. Fehlt das Land, so muss die Organisation direkt unter Root stehen.
6. Es dürfen mehrere Organisationen auf gleicher Ebene existieren.
7. Unterhalb des Objekts `o` dürfen entweder Blattobjekte (`cn`) oder organisatorische Einheiten (`ou`) existieren.
8. Den organisatorischen Einheiten können nur weitere Objekte vom Typ `ou` oder Blattobjekte folgen.
9. Es dürfen mehrere Objekte der Klasse `ou` auf gleicher Ebene existieren.
10. Blattobjekte (`cn`) dürfen nur unter `o` und `ou` vorkommen.

Diese Regeln sollten unbedingt bei der Erstellung und Pflege eines Verzeichnisdienstes befolgt werden. Oft überwacht die verwendete Software die Einhaltung dieser Richtlinien.

In Abbildung 3.9 existiert ein Country-Objekt `de`, unter dem sich die Organisationen `Firma-1` und `Firma-2` befinden. Beide Firmen haben jeweils einen Einkauf und einen Verkauf, wobei der Verkauf der zweiten Firma in die Bereiche `Inland` und `Ausland` aufgeteilt ist. Auf Blattobjekte wird in dieser Abbildung verzichtet.

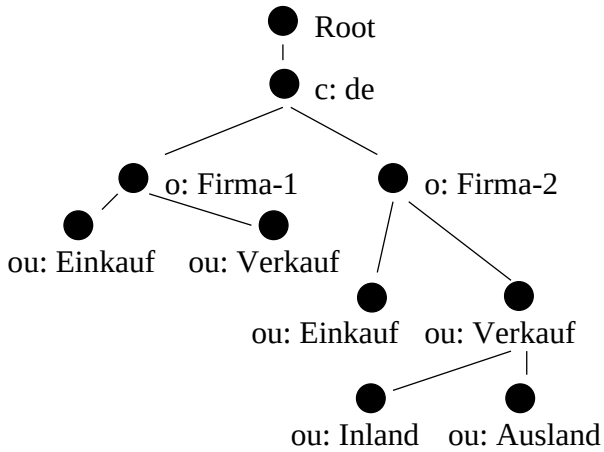

Abbildung 3.9: Regeln für Container

Ein DIT mit einer Organisation `Firma X`, drei Abteilungen `Einkauf`, `Verkauf` und `Buchhaltung` sowie drei Blattobjekten hätte demnach das Aussehen von Abbildung 3.10.

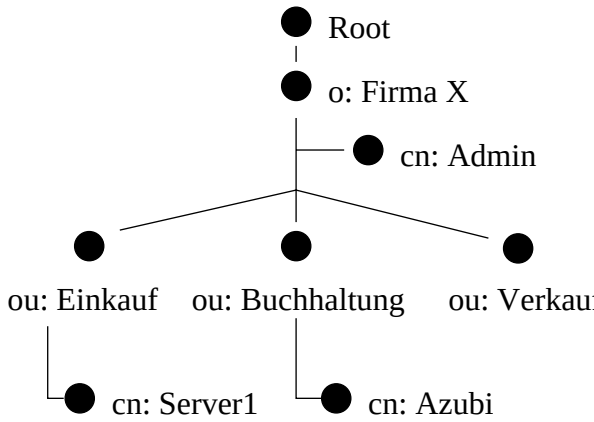

Abbildung 3.10: Regeln für Objekte

Das Objekt `Admin` ist hier der Administrator des gesamten Verzeichnisdienstes.

Auf diese Weise lässt sich im Prinzip jede vorhandene Struktur in Form eines Directory Information Tree abbilden. Bei einer mit dem Lightweight Directory Access

Protocol verwalteten X.500-Datenbank handelt es sich um ein Gerüst, dessen Struktur flexibel und funktionell organisiert werden kann.

Bei der Erstellung des DITs sollten neben den strukturellen Regeln auch folgende praktische Richtlinien beachtet werden:

- Es sollten nicht zu viele Containerobjekte auf einer Ebene stehen. Eine an die Gegebenheiten angepasste Unterteilung in Untercontainer erhöht die Performance der gesamten Datenbank.
- Falls LDAP-Clients in einem Container nach Objekten suchen, so erfolgt diese Suche sequentiell. Je mehr Einträge dort existieren, desto langsamer wird die Suche. Ein generelles Limit, wie viele Blattobjekte maximal in einem Container existieren sollten, hängt von der verwendeten LDAP-Software ab. Es ist jedoch davon auszugehen, dass es bei bis zu 1000 Objekten keine Probleme gibt.

`dn` und `rdn`

Bisher wurden die Objekte eines Directory Information Tree lediglich mit ihrem Namen und der Objektkennung bezeichnet. So gab es in Abbildung 3.9 zwei organisatorische Einheiten mit dem Namen `ou: Einkauf` und auch zwei mit dem Namen `ou: Verkauf`. Da Objekte im Verzeichnisdienst jedoch einen eindeutigen Namen haben müssen, kann es sich bei diesen Namen noch nicht um diejenigen handeln, die der Eindeutigkeit genügen. Jedes Objekt kann durch zwei Verfahren bezeichnet werden, die in Tabelle 3.6 aufgeführt sind.

Abkürzung	Beschreibung
`dn`	Distinguished Name
`rdn`	Relative Distinguished Name

Tabelle 3.6: `dn` und `rdn`

Bei dem Relative Distinguished Name handelt es sich um den Namen des Objekts in seinem übergeordneten Container (siehe Abbildung 3.11).

Wenn auf diesem Wege die Abteilung `Einkauf` der `Firma-1` angesprochen werden soll, so ist dies wegen der fehlenden Eindeutigkeit des Namens nicht möglich. In diesem Fall ist der Distinguished Name notwendig. Er setzt sich aus

- dem Namen des Objekts und
- den Namen der darüber liegenden Objekte

zusammen. Die Namen sind dabei in Richtung Baumwurzel aneinander zu reihen (siehe Abbildung 3.12).

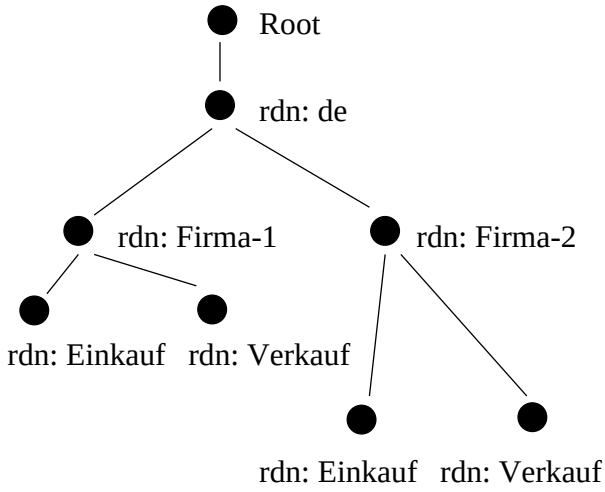

Abbildung 3.11: Relative Distinguished Name

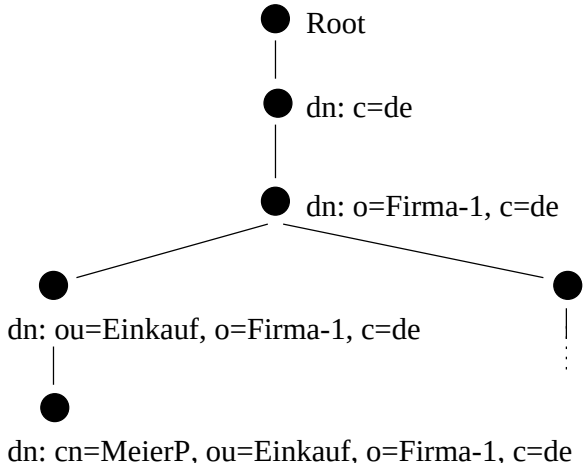

Abbildung 3.12: Distinguished Name

Der eindeutig kennzeichnende Name `dn` des Objekts `Einkauf` von `Firma-1` ist demnach:

```
dn: ou=Einkauf, o=Firma-1, c=de
```

Der Benutzer `MeierP` wird in dieser Schreibweise als

```
dn: cn=MeierP, ou=Einkauf, o=Firma-1, c=de
```

bezeichnet. Die Einkaufsabteilung der `Firma-2` hat den eindeutigen Namen:

```
dn: ou=Einkauf, o=Firma-2, c=de
```

Zusammenfassend kann festgehalten werden,

- dass Objekte mit gleichen Namen dann existieren dürfen, wenn sie sich im Distinguished Name unterscheiden, und
- dass Objekte mit gleichen Namen auf der gleichen Ebene in der Baumstruktur nicht möglich sind.

Context

Der Zugriff auf Informationen im Directory Information Tree erfolgt grundsätzlich aus einer Umgebung heraus. Diese Umgebung beschreibt den Container im Verzeichnisdienst, von dem aus auf die Daten zugegriffen wird. In vielen DITs wird dieser Sachverhalt mit dem Begriff Context bezeichnet. Er beschreibt die Position des Anwendungsprozesses in der Baumstruktur. Der Context wird vom LDAP-Client gewählt. Erfolgt zum Beispiel eine Suchaktion, so können neben den Distinguished Names auch Relative Distinguished Names verwendet werden. Der `rdn` ist somit die eindeutige Angabe eines Objekts ausgehend vom Context.

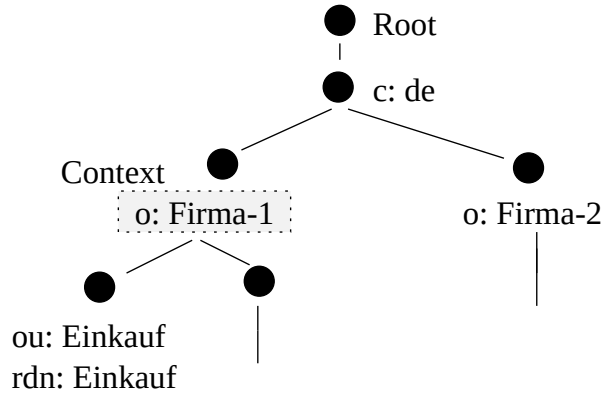

Abbildung 3.13: Context

Falls der Context in Abbildung 3.13 `Firma-1` lautet, so ist der `rdn` der Einkaufsabteilung `rdn: Einkauf`. Zwischen den Bezeichnungen `dn`, `rdn` und Context besteht ein Zusammenhang, der alle drei Begriffe nochmals verdeutlicht. Dabei wird der Context als `dn` angegeben.

Distinguished Name = Context + Relative Distinguished Name

Im zuvor genannten Beispiel bedeutet dies:

```
context: o=Firma-1, c=de
rdn: ou=Einkauf

dn: ou=Einkauf, o=Firma-1, c=de
```

Die Relativität beim `rdn` bezieht sich auf den Context. Bisher bestand der `rdn` lediglich aus dem Objektnamen. Dies ist jedoch nur dann korrekt, wenn als Ausgangspunkt (Context) der übergeordnete Container angenommen wird. Anderenfalls ist der `rdn` als Weg vom Ausgangspunkt zum gewünschten Zielobjekt zu beschreiben.

In Abbildung 3.14 gilt für die Verkaufsabteilung der zweiten Firma:

```
context: o=Firma-2, c=de
rdn: ou=Verkauf
```

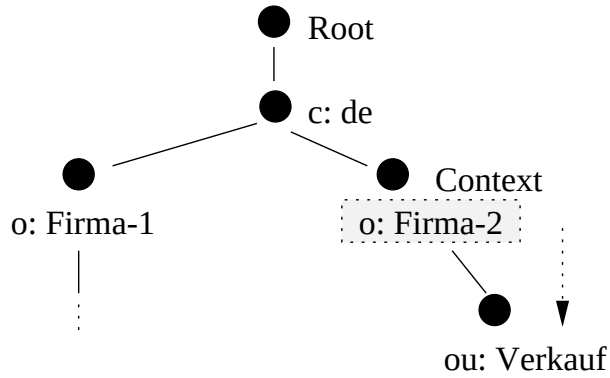

Abbildung 3.14: Context

Wird der Context jedoch auf die `Firma-1` gesetzt, so ist die Verkaufsabteilung der zweiten Firma relativ anders anzusprechen, nämlich mit

```
context: o=Firma-1, c=de
rdn: .., o= Firma-2, ou=Verkauf
```

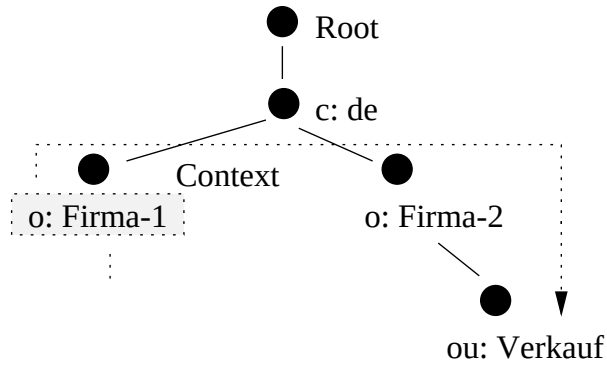

Abbildung 3.15: Context

Die beiden Punkte .. bedeuten, dass zunächst eine Ebene im Verzeichnisbaum nach oben in Richtung Wurzel gegangen werden muss, bevor zur `Firma-2` verzweigt werden kann.

Somit ist die Kombination von `rdn` und Context ein Kriterium, um Einträge im Verzeichnisdienst eindeutig zu beschreiben.

Zusammenfassend können folgende Aussagen festgehalten werden:

- Der Context beschreibt die Position im DIT. Er ist der Ausgangspunkt bei der Adressierung anderer Objekte.
- Bei dem Context handelt es sich grundsätzlich um ein Containerobjekt.
- Der Relative Distinguished Name wird in Abhängigkeit vom Context gebildet.
- Mit einer Kombination aus Context und `rdn` können Objekte im Verzeichnisdienst eindeutig bestimmt werden.

3.3.3 Der Namensraum

Bei dem Directory Information Tree (DIT) handelt es sich um eine echte Baumstruktur. Verzweigungen, die einen Kreislauf bilden, sind nicht zulässig. Die Aufteilung des DIT in Container- und Blattobjekte sowie deren Benennung wird als Namensraum bezeichnet. Der Sinn dieses Namensraumes ist natürlich die Beschreibung der Objekte mit logischen, umgangssprachlichen und funktionellen Begriffen, die dem Anwender des Verzeichnisdienstes eine leichte Identifizierung der DIT-Einträge erlaubt.

In den bisherigen Beispielen erfolgte die Benennung der Container- und Blattobjekte grundsätzlich anhand der widerzuspiegelnden Organisation. Es sind jedoch auch andere Punkte bei der Planung der Baumstruktur zu beachten:

- Organisation
- Objektarten
- Zugriffskontrolle
- Anwendungen
- Partitionierung
- Replizierung

Organisation. Die Ausrichtung an den organisatorischen Strukturen ist sicher das Verfahren, das in den meisten Fällen angewendet wird. Die Einträge im DIT werden dabei wie die zugrunde liegenden Abteilungen, Institute usw. bezeichnet. Der

Verzeichnisdienst bildet letztendlich ein Abbild der organisatorischen Gegebenheiten.

Objektarten. Ebenfalls denkbar wäre eine objektspezifische Aufteilung der Container. So ist es möglich, alle Benutzer zentral festzuhalten. Dieses hat zwei Vorteile:

1. Zum einen kann bei allen LDAP-Clients der gleiche Context eingestellt werden, damit sie auf ihre Benutzerobjekte zugreifen können.
2. Ferner kann der Administrator des Verzeichnisdienstes die DIT-Daten leichter pflegen, da er zum Beispiel nicht lange suchen muss, um herauszufinden, in welchem Container sich ein Benutzer befindet (siehe Abbildung 3.16).

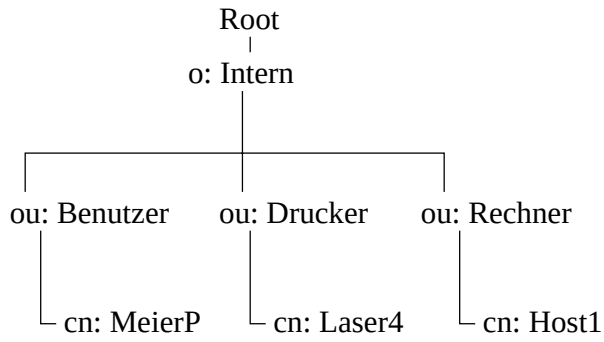

Abbildung 3.16: Namensraum nach Objektarten

Die Ausrichtung der Baumstruktur an den Objektarten ist eine eher ungebräuchliche Methode. Da die Anwender nicht nur Zugriff auf ihre Benutzerobjekte benötigen, sondern oft auch auf Rechner, Drucker usw., gestaltet sich die Adressierung der gewünschten Einträge meist sehr aufwendig, da der Context stets gewechselt werden muss.

Zugriffskontrolle. Eine wichtige Richtlinie bei der Gestaltung des Namensraumes ist die Berücksichtigung von Zugriffskontrollen.

Nehmen wir an, es existiert eine Struktur mit dem Namen `Intern`. Sie hat zwei Standorte (`Hamburg` und `Berlin`). In jedem dieser Standorte gibt es einen `Einkauf` und einen `Verkauf`. Falls in jedem Standort für jede Abteilung ein DIT-Administrator benannt werden soll, so muss er auf alle entsprechenden Daten schreibenden Zugriff erhalten. Dabei soll der Administrator für den Einkauf in Berlin natürlich keinen Zugriff auf die Einkaufsobjekte aus Hamburg haben und umgekehrt. Würde eine Organisation `o=Intern` erstellt und darunter direkt `ou=Einkauf` und `ou=Verkauf`, so wäre die Definition der Zugriffsrechte sehr schwierig. Zwar ist es möglich, die gewünschte Realisierung umzusetzen, eine Aufteilung wie in der Abbildung 3.17 hat jedoch einen entscheidenden Vorteil.

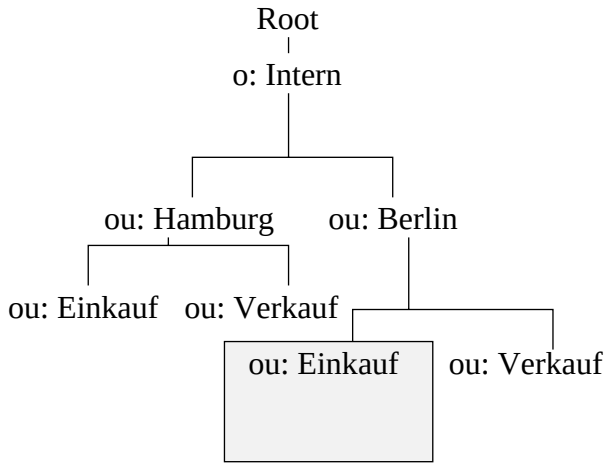

Abbildung 3.17: Zugriffskontrolle

Es ist möglich, dem Administrator des Einkaufs in Berlin auf den Container `ou=Einkauf, ou=Berlin, o=Intern` schreibende Rechte zu geben. Da sich diese Rechte auf die Objekte unterhalb des genannten Containers vererben, ist die Rechtevergabe mit diesem einen Schritt bereits beendet. Die Person, die zum Administrator ernannt worden ist, hat nun automatisch schreibende Rechte auf alle Objekte, die neu in dem Bereich angelegt werden, den sie betreut.

In der Praxis wird beim Erstellen des Namensraums neben der Organisation auch dieses Merkmal berücksichtigt.

Die Sicherheit von Verzeichnisdiensten wird in Abschnitt 3.3.6 auf Seite 39 beschrieben.

Anwendungen. Falls diverse Anwendungen die Daten der Verzeichnisdienste nutzen, so muss sichergestellt werden, dass diese Daten auch tatsächlich in der gewünschten Form an der gewünschten Stelle vorhanden sind.

Partitionierung. Bei der Partitionierung handelt es sich um das Aufteilen der Datenbank auf mehrere Server. Sie wurde bereits kurz beschrieben und wird im weiteren Verlauf dieses Kapitels (Abschnitt 3.3.5) mit all ihren Vor- und Nachteilen genau erläutert. Das Aufteilen des gesamten DIT kann in der Regel nur anhand von Containergrenzen geschehen. Auch dieses ist beim Design zu beachten. In Abbildung 3.17 könnten die Subtrees `Berlin` und `Hamburg` auf Servern an den entsprechenden Standorten abgelegt sein. Als Subtree werden alle Objekte unterhalb eines Containers betrachtet.

Replizierung. Da auch die Replizierung, also das redundante Speichern von gleichen Informationen auf mehreren Rechnern, an Containergrenzen orientiert ist,

gelten die gleichen Aussagen wie bei der Partitionierung. Die Replizierung wird im folgenden Abschnitt 3.3.4 genau beschrieben.

Ein Begriff, der ebenfalls in die Rubrik des Namensraumes fällt, ist das Suffix. Es gibt auf einem LDAP-Server an, welcher Subtree des gesamten DIT als Datenbank auf dem Server gespeichert ist.

Sind alle Daten des Verzeichnisdienstes auf dem Server gespeichert, so muss das Suffix nicht näher bestimmt werden. Anderenfalls enthält es den Distinguished Name des Containers, an dem der Subtree beginnt.

Ein Subtree ist ein Teilbaum. Er enthält alle Objekte von der Wurzel des Teilbaumes bis zu den Blättern.

Mit dem Begriff Suffix werden wir uns in Zusammenhang mit der Partitionierung ausführlicher beschäftigen.

3.3.4 Die Replizierung

Ein wesentliches Merkmal von Verzeichnisdiensten ist die Möglichkeit der Replizierung. Das bedeutet, dass die dem Verzeichnisdienst zugehörige Datenbank nicht nur auf einem Server gespeichert wird, sondern dass beliebig viele Duplikate auf anderen Servern existieren können. Der Rechner, der dabei die ursprünglichen Daten enthält, wird als der Master bezeichnet. Die Kopien tragen entsprechend der Funktionalität den Namen Replica (siehe Abbildung 3.18).

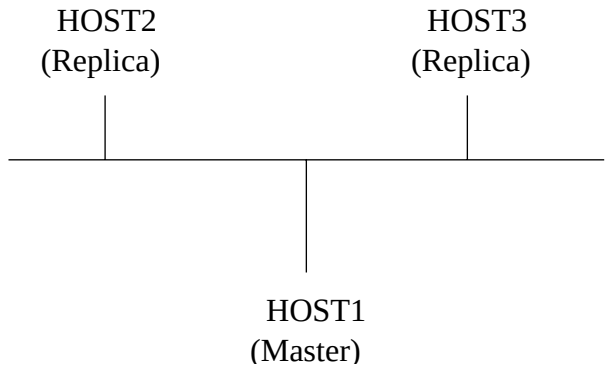

Abbildung 3.18: Master- und Replica-Server

Bevor beschrieben werden kann, wie eine Replizierung durchgeführt wird, sind zunächst die Gründe für diesen Vorgang zu nennen:

- Ausfallsicherheit
- Lastverteilung
- Zugriffsgeschwindigkeit

Ausfallsicherheit. Der Hauptgrund für die Nutzung der Replizierung in LDAP-Verzeichnisdiensten ist die Erhöhung der Ausfallsicherheit. Da Master und Replica die gleichen Daten zur Verfügung stellen, können Clients wahlweise den Originalserver oder den Rechner mit der Kopie ansprechen. Fällt zum Beispiel der Master aus, ist es fast ohne Datenverlust möglich mit einem Replica-Host weiterzuarbeiten (siehe Abbildung 3.19). Warum mitunter ein leichter Datenverlust auftritt, wird im weiteren Verlauf dieses Abschnittes erklärt.

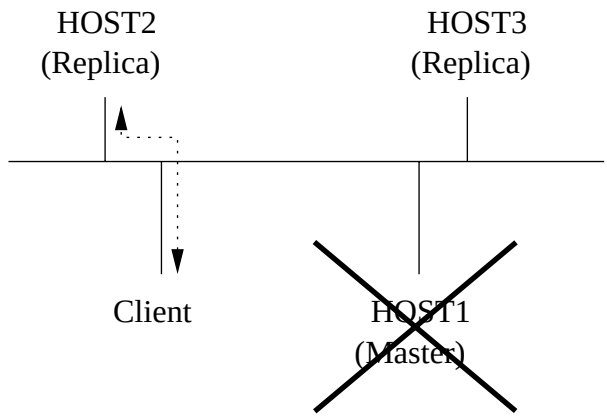

Abbildung 3.19: Ausfallsicherheit

Lastverteilung. Außerdem kann bei der Nutzung von replizierten Datenbanken eine Lastverteilung erfolgen. Dieses ist dadurch umzusetzen, wenn die LDAP-Clients unterschiedliche Server befragen. Es werden demnach nicht alle Anfragen an den Master gesendet, sondern es werden auch die Replicas genutzt (siehe Abbildung 3.20). Die Gesamtheit der LDAP-Anfragen ist zwar auch bei Verwendung der Replizierung die gleiche, jedoch verteilen sich die Verbindungen der Clients auf mehrere Hosts, wodurch jeder einzelne nur einen Teil der Last zu tragen hat.

Zugriffsgeschwindigkeit. Wenn Verzeichnisdienste zum Beispiel über WAN-Grenzen hinweg eingesetzt werden, ist es unüblich, dass der Zugriff auf den DIT über das Wide Area Network erfolgt. Dies rührt daher, dass solche Weitverkehrsstrecken in der Regel wesentlich langsamere Verbindungen haben als die Netze an einem Standort. Würde der Zugriff über diese Netze erfolgen, so wären unnötige Zeitverzögerungen die Folge. Daher wird die Replizierung in der Praxis so eingesetzt, dass an jedem Standort mindestens ein Master- bzw. Replica-Server vorhanden ist, den die Clients nutzen können (siehe Abbildung 3.21).

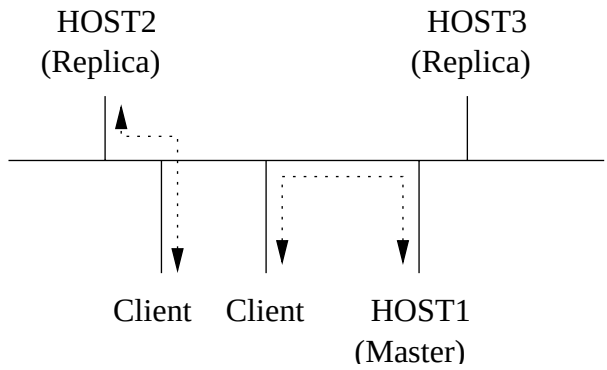

Abbildung 3.20: Lastverteilung

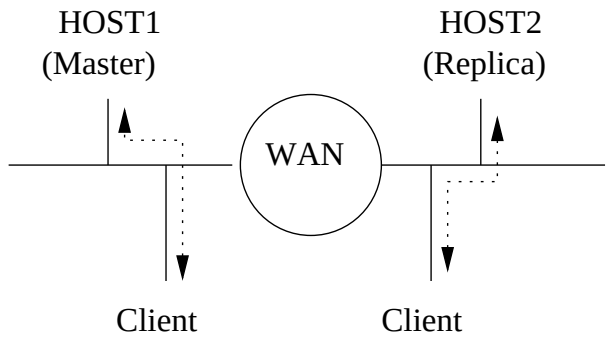

Abbildung 3.21: Zugriffsgeschwindigkeit

Bei der praktischen Umsetzung der Replizierung gibt es verschiedene Mechanismen, die beachtet werden müssen. Sie haben eine Auswirkung darauf, wie genau die Kopie und das Original übereinstimmen.

- Intervall
- Komplette Replizierung
- Inkrementelle Replizierung

Intervall. Der Master-Server ist derjenige, an dem alle Änderungen in der Datenbank des Verzeichnisdienstes durchgeführt werden. Da die Replica-Server Duplikate dieser Daten erhalten müssen, müssen die Informationen vom Master zur Replica transportiert werden. Das Intervall spiegelt dabei einen entscheidenden Faktor wider. Es beschreibt, in welchen Zeitperioden dieser Datenabgleich stattfindet. Werden die Daten zum Beispiel einmal pro Stunde übertragen, so ist der Replica-Server im Prinzip eine Stunde lang nicht up to date. Das heißt, die Kopie des Directory Information Tree und das Original stimmen nur bedingt überein. Je

kleiner das Zeitintervall ist, desto exakter ist die Kopie. Findet die Datenübertragung sehr oft statt, so hat dies auch Nachteile. Für jeden Verbindungsaufbau wird das Netzwerk entsprechend belastet. Erfolgt das Update über ein Wide Area Network, so ist es für diesen Zweck notwendig, dass Wählleitungen aktiviert werden. Jede Datenübertragung im WAN stellt somit einen Kostenfaktor dar. Letztlich sollte vor der Konfiguration bestimmt werden, wie aktuell die Kopie mindestens sein muss.

Komplette Replizierung. Die Daten des Masters müssen den Replica-Servern mitgeteilt werden. Dabei ist es möglich, jeweils die komplette Master-Datenbank zu übertragen. Dieses stellt zwar ein sicheres und einfaches Verfahren dar, doch verursacht es unnötigen Verkehr im Netzwerk, da dem Replica-Server Datensätze mitgeteilt werden, die dieser bereits gespeichert hat.

Inkrementelle Replizierung. In der Praxis erfolgt eine komplette Replizierung hauptsächlich dann, wenn ein neuer Replica-Server eingerichtet wird. Er erhält dann einmalig alle Daten des Masters. Anschließend werden in dem definierten Intervall lediglich die geänderten Einträge aus der Datenbank des Originals übernommen. Auf diese Weise werden nur sehr geringe Datenmengen im Netzwerk transportiert. Der Nachteil dieser Methode ist, dass ein misslungenes Update nicht wieder korrigiert wird. Es ist deshalb ratsam, in größeren Zeitabständen gelegentlich eine komplette Replizierung durchzuführen.

3.3.5 Die Partitionierung

Bei einer Partitionierung wird der gesamte Verzeichnisdienst in mehrere Teile aufgeteilt. Dabei werden Subtrees der Baumstruktur aus folgenden Gründen auf unterschiedlichen Servern abgelegt:

- Zu viele Objekte
- Zugriffsgeschwindigkeit
- Aufwand bei der Replizierung

Zu viele Objekte. Die Anzahl der im Verzeichnisdienst gespeicherten Objekte stellt ein wesentliches Merkmal für die Schnelligkeit der gesamten LDAP-Implementierung dar. So gibt es für jede Software eine maximale Anzahl von Objekten pro Datenbank. Sie liegt meist bei mehreren Tausend Einträgen pro Container. In der gesamten Baumstruktur sind bis zu 100 000 Objekte keine Seltenheit. In großen Computernetzen, die mit LDAP verwaltet werden, ist es dennoch möglich, dass die Beschränkungen der zugrunde liegenden Software erreicht werden. Es gilt dann, den Verzeichnisdienst so aufzuteilen, dass die einzelnen Datenbanken wieder unterhalb des Limits liegen (siehe Abbildung 3.22).

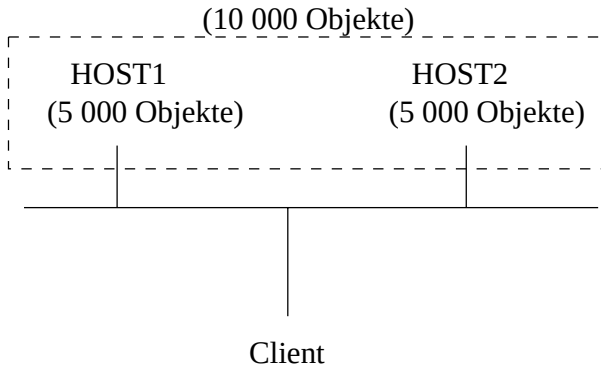

Abbildung 3.22: Partitionierung

Zugriffsgeschwindigkeit. Beim Zugriff auf die einzelnen Objekte und deren Attribute ist die Geschwindigkeit für den Anwender von entscheidender Bedeutung. Dabei ist die Geschwindigkeit, mit der eine Anfrage vom Directory Service beantwortet wird, von der Größe der Datenbank abhängig. Je mehr Objekte in einem Container existieren, desto länger dauert ein Zugriff auf eines dieser Objekte. Dies rührt daher, dass die Objekte zwar hierarchisch strukturiert sind, die Suche innerhalb der Hierarchie jedoch sequentiell erfolgt. Falls keine akzeptablen Zugriffszeiten mehr erzielt werden, ist eine Partitionierung unumgänglich.

Aufwand bei der Replizierung. Falls die Daten des Directory Service auf andere Server im Netzwerk repliziert werden, ist das Datenvolumen, das bei der Replizierung übertragen wird, von der Menge der Aktivitäten auf dem Master-Server abhängig. Werden dort zu viele Aktionen durchgeführt, kann es mitunter zu einer Überbelastung des Netzwerks kommen. Auch in diesem Fall ist eine Aufteilung der Verzeichnisdienst-Daten ratsam.

Nachdem die Gründe für eine Partitionierung erläutert wurden, soll nun die Vorgehensweise bei der Aufteilung sowie deren Zusammenhang mit der Replizierung beschrieben werden, und zwar mittels:

- Grenzen
- Regeln

Grenzen. Die Datenbank eines Directory Information Tree wird grundsätzlich an den Grenzen von Containern aufgeteilt. Dadurch bildet der am Container beginnende Teilbaum eine Partition. Sie wird auf einem anderen Server abgelegt und ist mit Querverweisen in die gesamte Baumstruktur eingebunden. Das bereits bekannte Beispiel der Firma `intern` könnte demnach so partitioniert sein, wie in der Abbildung 3.23 zu sehen ist.

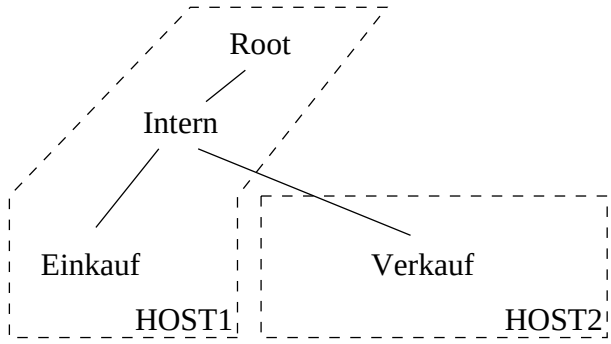

Abbildung 3.23: Partitionierung; Beispiel

Die Wurzel des Baumes befindet sich zusammen mit `Intern` und dem `Einkauf` auf dem Server `HOST1`. Der Verkauf liegt auf `HOST2`.

Clients aus dem Verkauf stellen bei dieser Partitionierung lediglich eine Verbindung zum `HOST2` her. Falls sie Anfragen außerhalb dieses Containers absetzen möchten, werden diese automatisch mit der Hilfe von Querverweisen aufgelöst und an den entsprechenden Stellen beantwortet.

Regeln. Bezüglich der Aufteilung des Verzeichnisbaumes in Partitionen sind einige Regeln zu beachten (siehe Abbildung 3.24).

- Bei der Wurzel einer Partition muss es sich um die Wurzel des gesamten Verzeichnisdienstes oder um einen Container handeln.
- Die Partition muss sich immer auf einen kompletten Teilbaum beziehen.
- Partitionen dürfen keine Löcher enthalten.

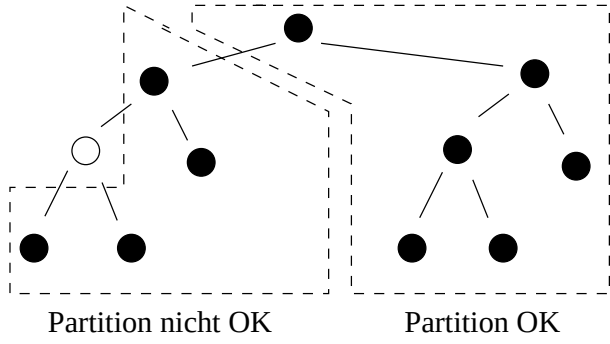

Abbildung 3.24: Regeln zur Partitionierung

Die folgende Aufzählung enthält einige grundsätzliche Aussagen, die den Zusammenhang zwischen Partitionierung und Replizierung verdeutlichen.

1. Der gesamte Verzeichnisdienst kann als eine große Partition eingerichtet werden.
2. Die Datenbank des gesamten Verzeichnisdienstes kann mit den Mitteln der Replizierung auf mehrere Server dupliziert werden. Das Original wird dabei als Master, die Kopie als Replica bezeichnet.
3. Die Baumstruktur kann in mehrere Partitionen aufgeteilt werden.
4. Jede Partition repräsentiert einen Teilbaum des Verzeichnisdienstes.
5. Jede Partition kann mit den Mitteln der Replizierung dupliziert werden.
6. Jede Partition ist im Original auf einem Master-Server vorhanden.
7. Jede Partition kann auf beliebig viele Replica-Server dupliziert werden.
8. Die Replizierung der verschiedenen Partitionen erfolgt unabhängig.

Mit diesen Aussagen ist es demnach möglich, die Partitionen des Directory Information Tree unabhängig voneinander zu replizieren.

In Abbildung 3.25 sind zwei Standorte über eine WAN-Verbindung aneinander gekoppelt. Der Verzeichnisdienst ist in zwei Partitionen unterteilt. Die erste enthält die Wurzel (Root) sowie `Standort-A` und umfasst neben dem Master zwei Replicas. Die zweite Partition besteht lediglich aus `Standort-B`. Sie wird nur auf einen Replica-Server dupliziert.

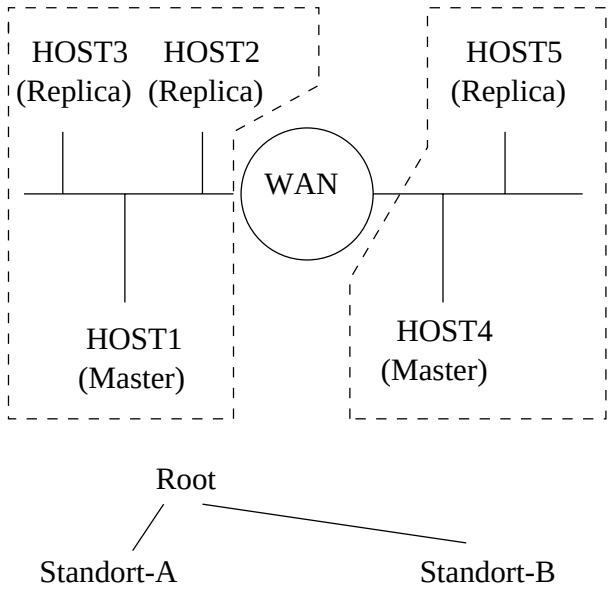

Abbildung 3.25: Partitionierung und Replizierung

Die Vorteile dieser Unterteilung bestehen darin, dass an jedem Standort ausschließlich die Daten gespeichert sind, die dort benötigt werden. Dieses erhöht die Geschwindigkeit beim Zugriff. Ferner werden die Daten an den Standorten ein- bzw. zweimal repliziert, wodurch die Ausfallsicherheit und die Lastverteilung gewährleistet werden. Falls die beiden Standorte keine Informationen des jeweils anderen benötigen, so werden auch keine LDAP-Daten über die WAN-Verbindung getauscht. Es ist dennoch möglich, den Verzeichnisdienst als Ganzes zu verwalten.

Die Verwaltung des DS-Tree sowie die Anfragen an Teile des Dienstes, die nicht direkt in der Datenbank vorhanden sind, werden über sogenannte Querverweise realisiert. Dabei gibt es Verweise in zwei Richtungen:

- Falls eine Anfrage an eine Partition gestellt wird, die in einem übergeordneten Teil des DIT beantwortet werden muss, so wird der Verweis der Partitionswurzel in Richtung Tree-Wurzel verwendet.
- Falls die Anfrage an einen untergeordneten Teil eines Baumes weitergeleitet werden muss, so wird der Verweis eines Containers aus der Partition verwendet.

Über diese beiden Verweise sind alle Teile der gesamten Baumstruktur miteinander verbunden. Querverweise werden in der Regel mit dem Begriff `referral` (Empfehlung) bezeichnet.

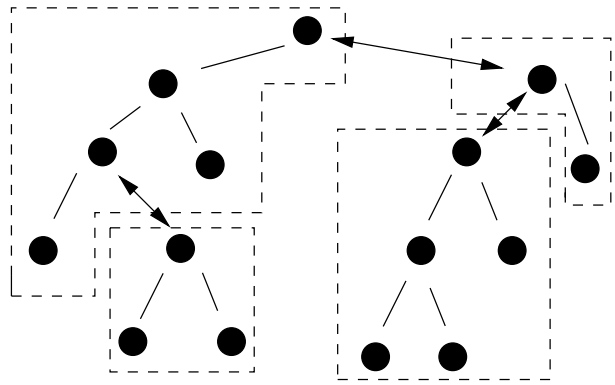

Abbildung 3.26: Querverweise der Partitionierung

3.3.6 Die Sicherheit

Die Sicherheit spielt wie in allen Netzwerkbereichen auch bei den Directory Services eine wichtige Rolle. In diesem Abschnitt wird beschrieben, welche Sicherheitsmechanismen zur Verfügung stehen und welchen potenziellen Angriffen die Implementierung der Verzeichnisdienste ausgesetzt ist.

Die Aufgaben der Sicherheit lassen sich in drei Bereiche einteilen:

- Zugriffsschutz
- Protokollierung
- Netzwerkverbindung

Zugriffsschutz. Der Zugriff auf die Informationen der Verzeichnisdienste ist durch bestimmte Maßnahmen zu schützen. Da es in der Regel unerwünscht ist, allen Anwendern lesende und schreibende Rechte auf alle Informationen zu geben, erfolgt die Zuweisung, für welche Aktionen welcher Benutzer zugelassen ist, anhand des Benutzernamens, verbunden mit dem Benutzerpasswort. Anhand dieses eindeutigen Merkmals wird entschieden, auf welche Daten der Anwender zugreifen kann. Pro Verzeichnisdienst existiert in der Regel ein Administrator (zum Beispiel mit dem Namen `Admin`), der alle Rechte auf alle DS-Einträge besitzt. Dabei ist es möglich, die Rechte zum

- Vergleichen,
- Suchen,
- Lesen,
- Schreiben und
- Löschen

für jedes Objekt und jedes Attribut zu vergeben.

Protokollierung. Ein genauso wichtiger Aspekt der Sicherheit ist das Protokollieren der durchgeführten Aktionen. Auf diese Weise kann genau eingesehen werden, welcher Anwender zu welchem Zeitpunkt welche Aktionen durchgeführt hat. Das regelmäßige Analysieren der Protokollinformationen stellt einen wichtigen Mechanismus dar, um etwaige Eindringlinge oder unerwünschte Aktionen zu erkennen.

Netzwerkverbindung. Die Verbindung vom Client des Verzeichnisdienstes zum Server erfolgt über das zugrunde liegende Netzwerk. Sie basiert auf dem TCP/IP-Protokoll und wird über den TCP-Port 389 abgewickelt. Die Daten, die auf diese Weise zwischen den LDAP-Komponenten ausgetauscht werden, dürfen von anderen Teilnehmern des Netzwerks nicht abgehört werden. Deshalb müssen sie absolut abhörsicher zwischen den Teilnehmern ausgetauscht werden. Dieses lässt sich mit einer RSA-Verschlüsselung (zum Beispiel mit der Secure Shell, SSH) realisieren.

Die Verbindung zwischen zwei Partnern im Netzwerk basiert bei der Secure Shell auf drei Schlüsseln:

- einem öffentlichen Schlüssel (public Key)

- einem privaten Schlüssel (private Key)
- einem Sitzungsschlüssel (Session Key)

Für jeden Rechner, der an der sicheren Kommunikation teilhaben soll, existiert ein öffentlicher und ein privater Schlüssel. Der öffentliche Schlüssel kann von jedermann eingesehen und den anderen Partnern im Netzwerk mitgeteilt werden. Der private Schlüssel hingegen verbleibt unter strengster Geheimhaltung auf jedem Rechner und wird niemandem mitgeteilt. Die Verschlüsselung der Daten erfolgt dann wie folgt:

- Rechner A und Rechner B wollen eine sichere Verbindung über die Secure Shell (SSH) aufbauen.
- Rechner A baut die Verbindung zu Rechner B auf. Beide Rechner tauschen ihre öffentlichen Schlüssel untereinander aus.
- Die Daten, die Rechner A an Rechner B sendet, werden mit dem öffentlichen Schlüssel von B codiert. Nur der zugehörige private Schlüssel von B kann die Daten entschlüsseln.
- Die Daten, die Rechner B an Rechner A sendet, werden mit dem öffentlichen Schlüssel von A codiert. Nur der zugehörige private Schlüssel von A kann die Daten entschlüsseln.
- Die Daten, die mit einem öffentlichen Schlüssel codiert sind, können nur mit dem zugehörigen privaten decodiert werden.
- Auf der Basis dieser Tatsachen vereinbaren die Teilnehmer der SSH-Kommunikation einen Sitzungsschlüssel. Dieser wird dem jeweils anderen Partner zu Beginn der Datenübertragung codiert mitgeteilt.

Bei den öffentlichen und privaten Schlüsseln handelt es sich um Primfaktoren einer Zahl X. Diese beiden Faktoren sind Primzahlen, deren Produkt die Zahl X ist. Primzahlen sind dadurch definiert, dass sie nur durch die Zahl 1 und durch sich selbst teilbar sind. Für die Primfaktorzerlegung einer Zahl gibt es keine feste Formel. Die Berechnung erfolgt mit diversen Algorithmen, die im Prinzip darauf basieren, die gewünschten Faktoren zu erraten. Je größer die Zahl X ist, desto aufwendiger ist dieses Verfahren. Demzufolge ist die Größe der Zahl X ein Maß für die Sicherheit. Die SSH verwendet dazu 1024-Bit-Zahlen, für die es heutzutage unmöglich ist, eine Primfaktorzerlegung durchzuführen.

Die zuvor aufgeführten Sicherheitsaspekte dienen dazu, den nicht autorisierten Zugriff auf die Informationen der LDAP-Daten zu unterbinden. Falls nun zum Beispiel Passwörter ausspioniert wurden, so kann anhand der Protokolle festgestellt werden, wann ein Anwender welche Aktionen durchgeführt hat. Durch eine Überprüfung, ob die Veränderungen tatsächlich von der realen Person durchgeführt wurden, lässt sich dieses Problem relativ schnell erkennen.

Falls die Datenübertragung im Netzwerk unverschlüsselt (ohne SSH etc.) stattfindet, so kann jeder Rechner im gleichen Netzwerksegment den Datentransport belauschen. Hierbei können neben Passwörtern auch direkt die transferierten Daten unbemerkt vom Angreifer gelesen werden. Diese Art von Angriff wird als *Network Sniffing* bzw. *Protocol Analyzing* bezeichnet.

Die aufgeführten Angriffstechniken sollen lediglich einen Einblick in die Problematik bieten, die mit dem Thema Sicherheit verbunden ist und den Anwender der LDAP-Software dazu motivieren, sich dieser Problematik anzunehmen. Da es sich bei der Sicherheit um ein generell wichtiges Thema handelt, das für alle Netzwerkkommunikationen relevant ist, werden sich die praktischen Darlegungen in den nächsten Kapiteln dieser Thematik annehmen.

3.4 Der LDAP-Client

Im folgenden Abschnitt wird die Funktionsweise des LDAP-Clients beschrieben.

3.4.1 Der Datenzugriff

Der Zugriff eines LDAP-Clients auf den LDAP-Server stellt den elementaren Vorgang bei der Nutzung von Verzeichnisdiensten dar. Der Datenzugriff kann dabei auf drei verschiedene Arten erfolgen:

- Anfragen mit einem Ergebnis
- Anfragen mit mehreren Ergebnissen
- Parallele Anfragen

Anfragen mit einem Ergebnis. Die erste Möglichkeit einer LDAP-Anfrage besteht darin, dass die Suchoperation des Clients zu genau einem Ergebnis führt. Der Server sendet dann neben den Ergebnis-Daten einen Ergebnis-Code (siehe Tabelle 3.7).

Schritt	Aktion	Richtung
1	Senden der Anfrage	Client ⇒ Server
2	Ergebnis-Daten der Anfrage	Client ⇐ Server
3	Ergebnis-Code der Anfrage	Client ⇐ Server

Tabelle 3.7: Anfragen mit einem Ergebnis

Anfragen mit mehreren Ergebnissen. Falls die Suche per LDAP mehr als ein Ergebnis liefert, so werden die Teilergebnisse nacheinander zum Client gesendet (siehe Tabelle 3.8). Der Ergebnis-Code wird am Ende der Übertragung geliefert.

Schritt	Aktion	Richtung
1	Senden der Anfrage	Client ⇒ Server
2	Erstes Teilergebnis der Anfrage	Client ⇐ Server
3	Zweites Teilergebnis der Anfrage	Client ⇐ Server
4	Drittes Teilergebnis der Anfrage	Client ⇐ Server
.	...	Client ⇐ Server
X	Ergebnis-Code der Anfrage	Client ⇐ Server

Tabelle 3.8: Anfragen mit mehreren Ergebnissen

Parallele Anfragen. Das Lightweight Directory Access Protocol (LDAP) erlaubt es den Anwendungsprozessen, mehrere Anfragen gleichzeitig abzusenden. Auf diese Weise kann zum Beispiel neben der ersten eine zweite Anfrage generiert werden, auch wenn die erste noch nicht beantwortet ist.

In diesem Punkt stellt LDAP eine Möglichkeit zur Verfügung, die in anderen Protokollen (z. B. HTTP) nicht vorhanden ist.

Schritt	Aktion	Richtung
1	Senden der ersten Anfrage	Client ⇒ Server
2	Senden der zweiten Anfrage	Client ⇒ Server
3	Ergebnis-Daten der ersten Anfrage	Client ⇐ Server
4	Ergebnis-Daten der zweiten Anfrage	Client ⇐ Server
5	Ergebnis-Code der ersten Anfrage	Client ⇐ Server
6	Ergebnis-Code der zweiten Anfrage	Client ⇐ Server

Tabelle 3.9: Parallele Anfragen

3.4.2 Die Verwendung der Daten

Für welchen Zweck die so erlangten Daten des Verzeichnisdienstes verwendet werden, kann vom Anwender flexibel bestimmt werden. Beschränkungen existieren nicht. Grundsätzlich bestehen jedoch zwei Möglichkeiten, was mit den Daten des DIT geschehen soll:

- Information
- Konfiguration

Information. Zum einen können die Elemente des Directory Service lediglich dazu verwendet werden, bestimmte Informationen zu erlangen. Hier kann es sich um

Telefonnummern, E-Mail-Adressen, IP-Nummern, Hostnamen usw. handeln. Der LDAP-Server dient somit lediglich als Informationsquelle (siehe Abbildung 3.27).

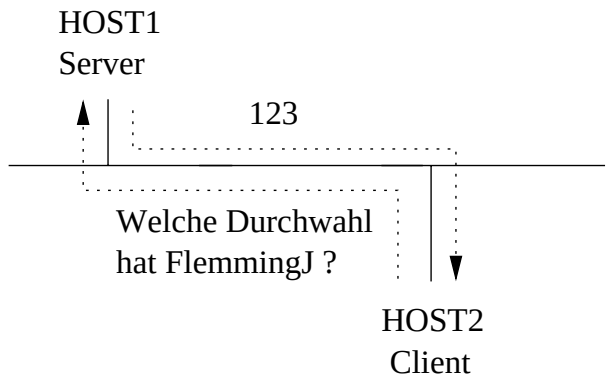

Abbildung 3.27: Verzeichnisdienst als Informationsquelle

Konfiguration. Zum anderen ist es möglich, die Daten des Verzeichnisdienstes zur Konfiguration anderer Anwendungen und Dienste zu verwenden. Der Directory Information Tree bietet in diesem Fall die Möglichkeit, zentral die Ressourcen im Netzwerk zu verwalten. So ist es denkbar, dass für jeden Host im Netzwerk ein eigenständiges Objekt existiert. Als Eigenschaft könnte dann vermerkt sein, welche Verzeichnisse der Host per NFS anderen Rechnern zur Verfügung stellen soll. Aufgrund dieser Daten kann das Network File System mit den folgenden Schritten anschließend konfiguriert und aktiviert werden (siehe Abbildung 3.28).

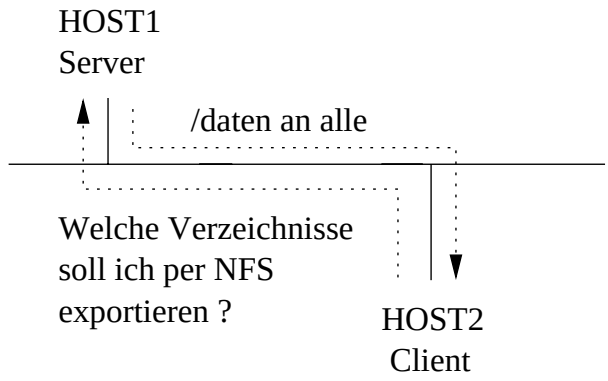

Abbildung 3.28: Verzeichnisdienst als Konfigurationsquelle

1. Der Client fragt den LDAP-Server nach den Konfigurationsparametern für eine Anwendung oder einen Dienst (zum Beispiel NFS).

2. Die Anwendung oder der entsprechenden Dienst wird am Client mit den vom LDAP-Server erlangten Daten konfiguriert.

In der Praxis werden in einem Verzeichnisdienst sowohl informative Daten als auch Daten, die zur Konfiguration von anderen Diensten verwendet werden, abgespeichert.

3.4.3 Nutzung anderer Dienste

Für den Zugriff auf Verzeichnisdienste müssen dem Anwender mehrere bedienerfreundliche Möglichkeiten zur Verfügung stehen. Der Anwender sollte auf mehreren Wegen seine Anfragen an die LDAP-Datenbank absetzen können. Dieses wird durch die Nutzung von anderen, bereits bekannten Diensten realisiert, die als LDAP-Client fungieren. Der Zugriff lässt sich in die folgenden Bereiche gliedern:

- Kommandos
- Spezielle Anwendungen
- Bekannte Netzwerkdienste

Kommandos. Mit Kommandos, die zum Beispiel in der Linux Shell `bash` ausgeführt werden können, existiert eine Möglichkeit, den Directory Service ohne grafische Anwendung zu kontaktieren. Der Nachteil solcher Kommandos besteht darin, dass die gewünschten Informationen in der Regel anhand von Parametern angegeben werden müssen. Das Erlernen und Beherrschen dieser Mechanismen bereitet vielen Endanwendern jedoch Schwierigkeiten. Der große Vorteil eines Zugriffs über Kommandos ist, dass der LDAP-Zugriff auf diese Weise in Shell-Scripts benutzt werden kann. Auch die so erlangten Ergebnisse können flexibel und dynamisch weiterverwendet werden.

Spezielle Anwendungen. Personen, die sich mit der Nutzung von Verzeichnisdiensten beschäftigen möchten, verwenden in der Regel keine Kommandos. Sie werden lediglich von Administratoren verwendet. Anwender haben ferner die Möglichkeit, grafische Anwendungen zu nutzen, die einen intuitiven Zugriff gewährleisten. Diese speziellen Programme müssen zwar ebenfalls erlernt werden, stellen jedoch eine benutzerfreundliche Oberfläche zur Verfügung. Mit grafischen Anwendungen ist es allerdings nicht möglich, die LDAP-Daten zur Konfiguration anderer Dienste zu verwenden.

Bekannte Netzwerkdienste. Um Anwendern den Kontakt zum Directory Service weiter zu vereinfachen, existieren Schnittstellen zwischen bereits bekannten Anwendungen und LDAP. So ist es zum Beispiel möglich, über einen Internet-Browser auf den Verzeichnisdienst zuzugreifen und sich in diesem zu bewegen. Auch können die Anfragen in Form von elektronischer Post abgesetzt werden. Die Nutzung von Netzwerkdiensten, die dem Anwender bereits bekannt sind, erhöht die Akzeptanz gegenüber LDAP.

Damit zum Beispiel eine Verbindung zwischen einem Internet-Browser und dem LDAP-Server zustande kommt, werden so genannte Gateways eingesetzt. Sie sind für den Anwender unsichtbar und nehmen die Anfragen, die mit dem HyperText Transfer Protocol (HTTP) getätigt wurden, entgegen und wandeln sie in LDAP-Anfragen um, die dem LDAP-Server dann übermittelt werden (siehe Abbildung 3.29). Ein Gateway ist somit ein Protokollkonverter zwischen dem Anwendungsprotokoll (im Beispiel HTTP) und LDAP.

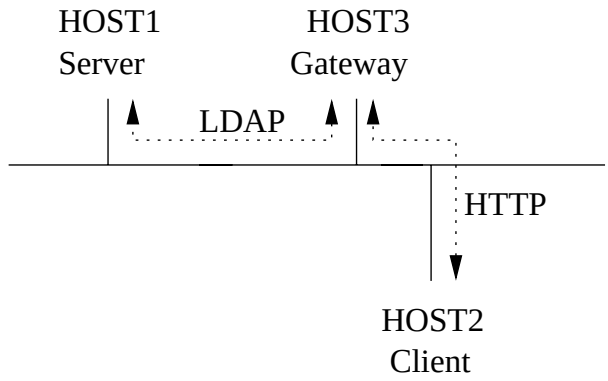

Abbildung 3.29: LDAP-Gateways

4 LDAP unter Linux

Nachdem die grundlegenden Funktionen und Begrifflichkeiten von Verzeichnisdiensten dargestellt wurden, befasst sich dieses Kapitel detailliert mit der praktischen Umsetzung der LDAP-Komponenten unter Linux.

4.1 Die Universität von Michigan

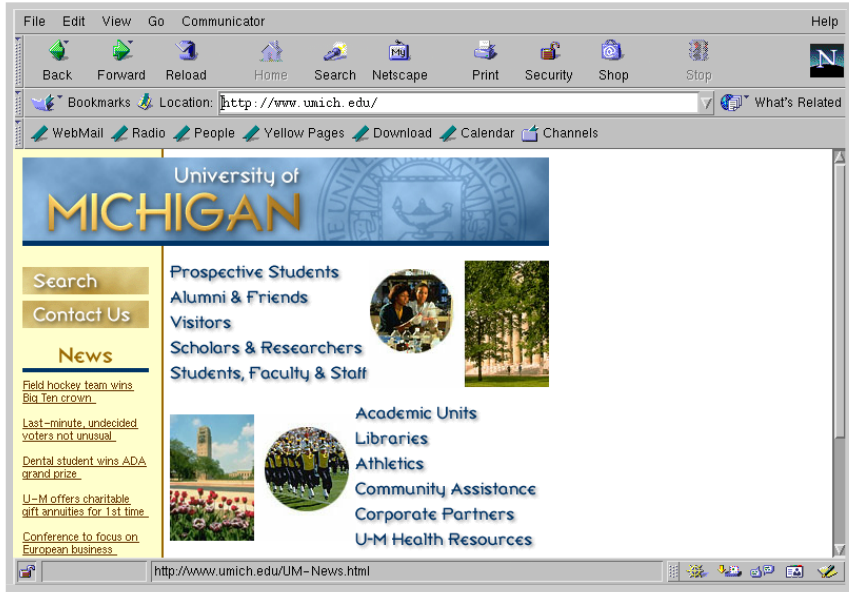

Abbildung 4.1: Homepage der Universität von Michigan

Die zugrunde liegende LDAP-Software unter Linux basiert auf einem Projekt der Universität von Michigan (`www.umich.edu`). Die dort durchgeführten Darlegungen wurden anschließend von einer weiteren Organisation in Form eines Software-Paketes implementiert.

Bei dieser Organisation handelt es sich um OpenLDAP. Sie ist unter der Internet-Adresse `www.openldap.org` zu erreichen. Neben der neusten Version sind auch weitere Informationen verfügbar. Sie umfassen Fragen und Probleme im Zusammenhang mit der Software.

Die von der Organisation OpenLDAP entwickelte Software mit dem gleichen Namen zeichnet sich durch

- ein Höchstmaß an Stabilität und
- ein Höchstmaß an Flexibilität

aus. Das Software-Paket besteht aus den Komponenten:

- LDAP-Server
- LDAP-Client
- LDAP-Gateways

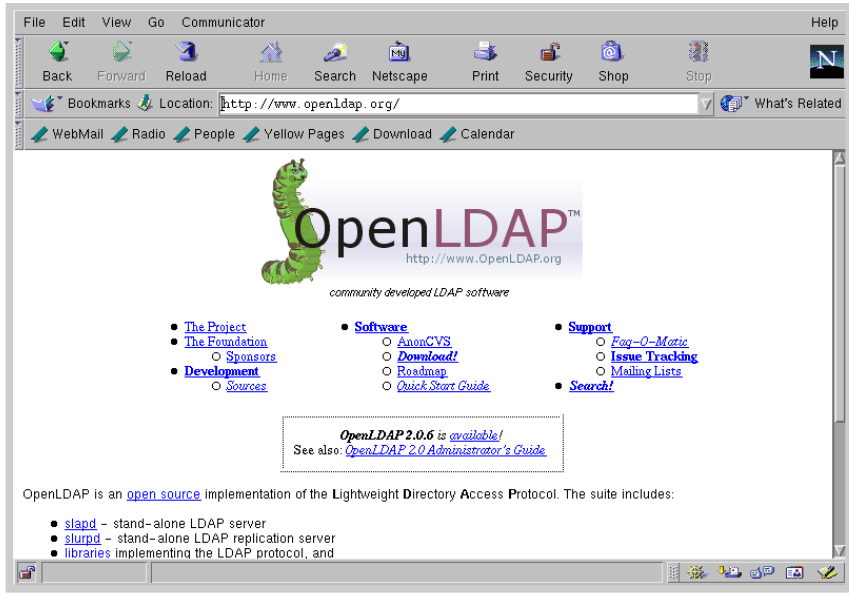

Abbildung 4.2: Homepage von OpenLDAP

Neben einer Implementierung für das Betriebssystem Linux existieren auch Umsetzungen für andere Unix-Varianten. Die Erläuterungen in diesem Buch gelten demzufolge zu einem großen Teil nicht nur für Linux, sondern auch für Unix.

4.2 Installation von OpenLDAP v1.2

Bei dem Software-Paket OpenLDAP handelt es sich um eine freie und offene LDAP-Implementierung. Neben der kostenlosen Nutzung sind auch die Quellen zu diesem Paket verfügbar.

Unter Linux wird Software in den meisten Distributionen durch so genannte RPM-Pakete repräsentiert. RPM steht dabei für Red Hat Package Manager. Dieses Format

wurde unter anderem auch von SuSE übernommen. Die Installation der entsprechenden Pakete erfolgt in jeder Distribution mit einem Anwendungsprogramm, in dem das Paket lediglich auszuwählen ist. Bei der SuSE-Distribution erfolgt der Aufruf mit

```
root@host1:~ # yast
```

oder:

```
root@host1:~ # yast2
```

Bei Red Hat ist es das Programm:

```
root@host1:~ # linuxconf
```

Die in diesem Kapitel aufgeführten Beschreibungen beziehen sich auf eine SuSE-Distribution. Pfade etc. können auf anderen Systemen zum Teil abweichen.

Bei der Installation der Software sind neben dem eigentlichen OpenLDAP-Projekt noch weitere Pakete zu installieren.

Paket	Beschreibung
`openldap`	LDAP-Komponenten
`ldaplib`	LDAP-Bibliothek
`kldap`	Grafische Anwendung
`webgw`	HTTP-LDAP-Gateway
`pam_ldap`	Passwort-Verifizierung

Tabelle 4.1: Benötigte Software-Pakete

Das fundamentale Software-Paket, das aus LDAP-Server, -Client und diversen -Gateways besteht, befindet sich im Paket `openldap`. Die Bibliothek mit Funktionen, die in Programmiersprachen genutzt werden können, ist in `ldaplib` zu finden. Eine grafische Anwendung, mit der der Verzeichnisdienst durchsucht werden kann, wird durch das Paket `kldap` repräsentiert. Das Gateway `webgw` konvertiert das Protokoll HTTP nach LDAP. `pam_ldap` ist dafür zuständig, dass die Verifizierung eines Benutzerpasswortes nicht anhand der Datei `/etc/shadow`, sondern anhand des Verzeichnisdienstes erfolgt. Tabelle 4.1 zeigt eine Übersicht der benötigten Software.

Ohne eine konfigurierte LDAP-Datenbank belegen die Anwendungen den folgenden Platz auf der Festplatte.

- `openldap` ≈ 6,4 Mbyte
- `ldaplib` ≈ 0,2 Mbyte
- `kldap` ≈ 0,2 Mbyte

- `webgw` ≈ 0,6 Mbyte
- `pam_ldap` ≈ 0,3 Mbyte

In der Summe benötigen die Pakete somit knapp 8 Mbyte.

Der RAM-Bedarf lässt sich nicht generell beziffern und muss im Einzelfall bestimmt werden.

4.3 Die ausführbaren Programme

Das Software-Paket OpenLDAP gliedert sich, wie bereits erwähnt, in die Bereiche Server, Client und Gateways. In ihnen sind jeweils ausführbare Programme und Kommandos vorhanden, mit denen die entsprechenden Aktionen gesteuert werden.

Programm	Beschreibung
`slapd`	LDAP-Server
`ldif2ldbm`	Datenkonvertierung
`ldbmcat`	Datenbank-Abzug
`slurpd`	LDAP-Replizierung

Tabelle 4.2: Dämonen des LDAP-Servers

slapd Bei dem Programm `slapd` handelt es sich um den Dämon, der den LDAP-Server realisiert. `slapd` steht für Standalone LDAP Daemon.

slurpd Der Dämon `slurpd` ist für die Replizierung zuständig. Die Abkürzung `slurpd` bedeutet Standalone LDAP Update Replication Daemon.

ldif2ldbm Die Einträge des DIT werden vom LDAP-Server in einer Datenbank mit dem Format LDBM (LDAP Database) gespeichert. Neben diesem Binärformat existiert auch eine textbasierte Variante namens LDIF (LDAP Directory Interchange Format, siehe Abschnitt 4.5.6). Die Konvertierung einer vom Administrator erstellten LDIF-Textdatei in das LDBM-Format erfolgt mit dem Kommando `ldif2ldbm`. Es wird dazu verwendet, um einen neuen Verzeichnisdienst aufzusetzen, dessen Grundgerüst mittels LDIF definiert wurde.

lbmcat Mit dem Kommando `ldbmcat` ist es dann möglich, alle Einträge des Verzeichnisdienstes auf der Textkonsole auszugeben.

Tabelle 4.2 fasst die Dämonen des LDAP-Servers übersichtlich zusammen. Der LDAP-Server wird detailliert im Abschnitt 4.5 auf Seite 54 beschrieben.

In Abschnitt 4.6 auf Seite 96 werden die ausführbaren Kommandos und Programme des LDAP-Clients beschrieben. Sie sind an der Linux-Eingabeaufforderung einzugeben und steuern den Zugriff auf die Daten des LDAP-Servers.

Für den LDAP-Client gibt es die in Tabelle 4.3 aufgeführten Kommandos.

Programm	Beschreibung
`ldapsearch`	Suchen
`ldapadd`	Hinzufügen
`ldapmodify`	Ändern
`ldapdelete`	Löschen
`ldappasswd`	Passwort ändern
`ud`	Benutzer bearbeiten
`kldap`	Grafische Anwendung
`netscape`	Internetbrowser

Tabelle 4.3: Kommandos des LDAP-Clients

ldapsearch Suchen von Objekten und Attributen im Verzeichnisdienst.

ldapadd Hinzufügen von Einträgen per LDAP in der X.500-Datenbank.

ldapmodify Ändern von Einträgen in der Datenbank.

ldapdelete Löschen von Elementen der Datenbank.

ldappasswd Mit dem Kommando `ldappasswd` kann das Passwortattribut eines Benutzerobjekts verändert werden.

ud Mit dem Programm `ud` ist es möglich, die Benutzerobjekte in der Datenbank des Servers mit einem interaktiven, textbasierten Programm zu bearbeiten. `ud` steht für LDAP User Directory.

kldap Hinter dem Programm `kldap` verbirgt sich eine grafische und anwenderfreundliche Oberfläche, mit der es möglich ist, auf den Verzeichnisdienst zuzugreifen. Sie basiert auf dem K Desktop Environment (KDE).

netscape Außerdem bietet der Internet-Browser `netscape` die Möglichkeit, die Daten seines Adressbuches direkt aus dem LDAP-Server zu gewinnen.

Auf der Seite 141 werden in Abschnitt 4.7 die LDAP-Gateways beschrieben. Sie bilden jeweils die Schnittstellen zwischen bereits bekannten Netzwerkdiensten und dem LDAP-Server. Tabelle 4.4 zeigt eine Übersicht der Dämonen der LDAP-Gateways.

in.xfingerd Finger-Anfragen von Clients an den Dämon `in.xfingerd` werden von diesem an den LDAP-Server weitergeleitet und von dort entsprechend beantwortet.

go500/go500gw Der Zugriff auf die X.500-konforme Datenbank des LDAP-Servers über den Dienst Gopher erfolgt mit `go500` und `go500gw`.

Programm	Beschreibung
`in.xfingerd`	Finger-Gateway
`go500/go500gw`	Gopher-Gateway
`rcpt500`	Mail-Gateway
`web500gw`	Web-Gateway

Tabelle 4.4: Dämonen der LDAP-Gateways

rcpt500 Das eMail-Gateway wird durch `rcpt500` realisiert.

web500gw Über `web500gw` kann der Zugriff über das HTTP-Protokoll erfolgen.

4.4 Die Konfigurationsdateien

Zu den Komponenten

- LDAP-Server,
- LDAP-Client und
- LDAP-Gateways

existieren neben den Kommandos diverse Konfigurationsdateien, mit denen das Verhalten beeinflusst werden kann (siehe Tabelle 4.5).

Datei	Beschreibung
`slapd.conf`	Allgemeine Server-Einstellungen
`slapd.oc.conf`	Objektdefinitionen
`slapd.at.conf`	Attributdefinitionen
`slapd.replog`	Logdatei für die Replizierung

Tabelle 4.5: Konfigurationsdateien des LDAP-Servers

slapd.conf In dieser Datei werden allgemeine Einstellungen des Servers festgehalten. Sie geben Aufschluss über die Datenbank, den Administrator usw.

slapd.oc.conf In die Datei `slapd.conf` können weitere Dateien eingebunden werden. Bei der Datei `slapd.oc.conf` handelt es sich um eine solche, in der die möglichen Objekte enthalten sind.

slapd.at.conf Auch die Datei `slapd.at.conf` wird in die allgemeine Konfigurationsdatei des Servers geladen. Sie enthält die Definitionen von Attributen.

slapd.replog Falls die LDAP-Datenbank repliziert werden soll, werden in der Datei `slapd.replog` alle Aktionen protokolliert. Diese Datei enthält ebenfalls Informationen darüber, auf welchen Rechner sie repliziert werden soll.

Auch auf dem Client existieren einige Konfigurationsdateien, die unter anderem den Server angeben, auf dem die LDAP-Datenbank installiert ist (siehe Tabelle 4.6).

Datei	Beschreibung
`ldap.conf`	Allgemeine Client-Einstellungen
`ldapfilter.conf`	Definition von Filtern
`ldaptemplates.conf`	Definition von Templates
`ldapsearchprefs.conf`	Definition von Suchvorgaben
`ldapfriendly`	Definition von Ländercodes
`ud.conf`	Konfiguration des Programms `ud`

Tabelle 4.6: Konfigurationsdateien des LDAP-Clients

ldap.conf In dieser Datei werden allgemeine Werte festgehalten. Sie beschreiben zum Beispiel den Rechner und den TCP-Port, an dem die LDAP-Datenbank zu finden ist.

ldapfilter.conf Filterdefinitionen, die von einigen Funktionen genutzt werden, sind in dieser Datei abgelegt.

ldaptemplates Unter einem Template versteht man im Prinzip eine Auflösung von einem Attribut zu einem verständlichen Namen. Erfolgt beispielsweise eine Client-Suche nach `cn`, so könnte über das Template die Ausgabe –Objektname– statt `cn` erscheinen.

ldapsearchprefs.conf Auf welches Objekt und auf welches Attribut welches Suchkriterium angewendet werden kann und wie sich dieses beim Client äußert, ist in dieser Datei angegeben.

ldapfriendly Falls ein Country-Objekt im Verzeichnis erstellt werden soll, so sind in der Datei `ldapfriendly` Auflösungen von dem zweistelligen Countrycode in den realen Namen des Landes vorhanden.

ud.conf Das bereits erwähnte Programm `ud` wird über die Datei `ud.conf` konfiguriert.

Bei den aufgeführten Gateways existiert im Prinzip nur eine wichtige Konfigurationsdatei. Es ist die Datei `web500gw.conf` (siehe Tabelle 4.7).

Datei	Beschreibung
`web500gw.conf`	Konfigurationsdatei des HTTP-Gateways

Tabelle 4.7: Konfigurationsdateien der LDAP-Gateways

4.5 Der LDAP-Server

Im Folgenden wird der LDAP-Server beschrieben, dem in der Kommunikation die zentrale Rolle zukommt.

4.5.1 Die Datei `slapd.conf`

Die Konfigurationsdatei des LDAP-Servers der OpenLDAP-Organisation ist die Datei `slapd.conf`. Der Ort, an dem sie abgespeichert ist, variiert bei den Linux-Distributionen. In einer SuSE-Umgebung liegt sie in `/etc/openldap/`.

Sie wird hauptsächlich vom Dämon `slapd` benutzt. Jedoch sind auch einige Einstellungen für die Replizierung mit dem Programm `slurpd` vorhanden.

Die Datei `slapd.conf` ist eine Textdatei, die mit jedem Editor bearbeitet werden kann. Zeilen, die mit dem Zeichen # beginnen, werden als Kommentar gedeutet und haben keine Auswirkung auf die Funktionalität.

Beispiel 4.1: Kommentare

```
root@host1:~ # cat /etc/openldap/slapd.conf
#
# Konfigurationsdatei des LDAP-Servers
#
# slapd.conf
#
# Erstellt von Jens Banning
#
root@host1:~ #
```

Argumente zu etwaigen Parametern sind in der Regel durch Leerzeichen voneinander getrennt. Falls ein einzelner Wert jedoch Blanks enthält, so kann er mit doppelten Anführungsstrichen (zum Beispiel `"Jan Flemming"`) umschlossen werden.

Allgemeines

Zu Beginn der Konfigurationsdatei sind allgemeine Einstellungen zum Server-Dienst vorzunehmen. Sie werden im Folgenden in Hinblick auf

- Beschreibung,
- Syntax und
- Beispiel

näher erläutert.

Der Parameter `include`

Die Einstellungen, die in der Datei `slapd.conf` zu tätigen sind, können schnell sehr umfangreich und unübersichtlich werden. Daher ist es möglich, sie auf mehrere Dateien zu verteilen. Jede dieser Dateien kann dann in die Konfiguration eingebunden werden. Dazu ist der Parameter `include` zu verwenden. Er hat folgende Syntax

```
include <Dateiname>
```

Die Dateien `slapd.at.conf` und `slapd.oc.conf` können auf diese Weise in die Konfiguration eingeschlossen werden:

```
include /etc/openldap/slapd.at.conf
include /etc/openldap/slapd.oc.conf
```

Der Parameter `pidfile`

Jeder Prozess unter Linux wird durch eine eindeutige Prozesskennung, die Process Identification (PID), repräsentiert. Die PID des Dämons `slapd` wird in der Datei protokolliert, die auf diesem Parameter folgt:

```
pidfile <Dateiname>
```

Im Beispiel wird die PID in einer Datei im Verzeichnis `/var/run/` abgelegt:

```
pidfile /var/run/slapd.pid
```

Der Parameter `argsfile`

Neben der Prozesskennung wird der Aufruf des Dämons `slapd` mit all seinen Parametern in einer Datei festgehalten. Sie wird nach dem Schema

```
argsfile <Dateiname>
```

in der Konfigurationsdatei angegeben:

```
argsfile /var/run/slapd.args
```

Der Parameter `loglevel`

Der LDAP-Server ist in der Lage, diverse Meldungen und Aktivitäten in der Logdatei `/var/log/messages` zu protokollieren. Die Menge der Protokolleinträge kann allgemein mit der Definition eines Loglevels festgelegt werden. Dem allgemeinen Aufruf

```
loglevel <Wert>
```

ist ein Wert zu übergeben, der sich aus der Addition der Zahlen in Tabelle 4.8 ergibt.

Level	Beschreibung
1	Funktionsaufrufe
2	Paket-Debugging
4	Detailliertes Debugging
8	Verbindungsmanagement
16	Gesendete und empfangene Pakete
32	Filteraktionen
64	Konfigurationsdatei-Aktionen
128	Zugriffsaktionen
256	Verbindungen/Operationen/Ergebnisse
512	Gesendete Einträge
1024	Kommunikationen
2048	Prüfung der Einträge

Tabelle 4.8: Übersicht der Loglevel

Das Verbindungsmanagement zusammen mit Verbindungen/Operationen/Ergebnissen hat in der Addition den Wert 8 + 256 = 264. Falls zum Beispiel nur alle Verbindungen, Operationen und Ergebnisse protokolliert werden sollen, so ist dies mit dem Loglevel 256 möglich:

```
loglevel 256
```

Der Parameter `schemacheck`

Die Menge der möglichen Objekte und Attribute wird als Schema des Verzeichnisdienstes bezeichnet. Der LDAP-Server der OpenLDAP-Organisation ist in der Lage, das Bearbeiten des DIT daraufhin zu überwachen, ob das Schema eingehalten wird. Aktionen, wie das Hinzufügen eines unbekannten Objekts sind in diesem Fall nicht möglich. Ob die Prüfung erfolgt, wird mit dem Parameter

```
schemacheck <on|off>
```

eingestellt. Ihm wird der Wert `on` oder `off` (Default) übergeben:

```
schemacheck on
```

Der Parameter `sizelimit`

Falls ein Client nach Einträgen im Verzeichnisdienst sucht, so werden ihm diese mitgeteilt. Hat die Suche mehrere Ergebnisse, so werden ihm maximal so viele übermittelt, wie mit dem Parameter `sizelimit` angegeben wurde:

```
sizelimit <Wert>
```

Der Defaultwert liegt bei 500, also:

```
sizelimit 500
```

Der Parameter `timelimit`

Über `timelimit` wird die maximale Anzahl von Sekunden angegeben, die der LDAP-Server für die Beantwortung von Anfragen aufwendet:

```
timelimit <Sekunden>
```

Der Standardwert beträgt 3600 Sekunden (1 Stunde), also:

```
timelimit 3600
```

Der Parameter `srvtab`

Falls die Beglaubigung der Clients über das Kerberos-Verfahren erfolgt, so gibt dieser Parameter die Datei an, in der die Schlüssel abgelegt sind.

```
srvtab <Dateiname>
```

Diese Einstellung ist nur relevant, wenn die Beglaubigung nach dem genannten Verfahren erfolgt (kein Standard). Man schreibt dann:

```
srvtab /etc/srv.tab
```

Die Deklaration `objectclass`

Das zentrale Element der LDAP-Datenbank sind die Objekte. Sie werden in Objektklassen spezifiziert und können somit im gesamten Verzeichnisdienst verwendet werden. Die allgemeine Syntax dieser Deklaration ist:

```
objectclass <Name>
        requires
                <Attribut 1>,
                <Attribut 2>,
                <…>
        allows
                <Attribut 1>,
                <Attribut 2>,
                <..>
```

Neben dem eindeutigen Namen der Klasse sind alle Attribute mit Namen zu nennen, die unbedingt zu einem Objekt dieser Klasse angegeben werden müssen (`requires`). Ferner werden die optionalen Eigenschaften definiert (`allows`). Die Deklaration muss für jede gewünschte Klasse separat aufgeführt werden. So beschreibt die folgende Klasse Objekte des Typs `person` mit den für dieses Objekt notwendigen und optionalen Eigenschaften:

```
objectclass person
        requires
                objectClass,
                sn,
                cn
        allows
                description,
                seeAlso,
                telephoneNumber,
                userPassword
```

Die Verwendung der `objectclass`-Deklaration ist nur dann sinnvoll, wenn der Parameter `schemacheck` eingeschaltet ist.

Die Deklaration `attribute`

Attribute können in den Deklarationen für die Objektklassen beliebig verwendet werden. Falls für ein Attribut jedoch eine Definition der zugrunde liegenden Syntax angegeben werden soll, so muss dieses über die `attribute`-Deklaration erfolgen. Sie muss für jede Eigenschaft separat existieren und ist nach der Struktur

```
attribute <Name> <Syntax>
```

zu verwenden. Dem eindeutigen Namen folgt eine der Bezeichnungen, die in Tabelle 4.9 angegeben sind.

Syntax	Beschreibung
`bin`	binary
`ces`	case exact string
`cis`	case ignore string
`dn`	distinguished name
`tel`	telephone number string

Tabelle 4.9: Syntax der Attribute

Neben der binären Syntax definieren `ces` und `cis` Texte, die zum einen die Groß- und Kleinschreibung unterscheiden und zum anderen nicht. Die Syntax `dn` beschreibt einen Distinguished Name eines anderen Objektes, `tel` eine Telefonnummer. Falls ein Attribut nicht bezüglich seiner Syntax spezifiziert wird, so wird `cis` verwendet.

Die folgende Liste gibt einige Attribut-Definitionen an:

```
attribute       photo                                   bin
attribute       audio                                   bin
attribute       userpassword                            ces
attribute       telephonenumber                         tel
attribute       homephone                               tel
attribute       member                                  dn
attribute       owner                                   dn
attribute       seealso                                 dn
attribute       manager                                 dn
attribute       dn                                      dn
```

Aufgrund der Defaultvorgabe ist es nicht notwendig, `cis`-Attribute über diese Deklaration anzugeben.

Nachdem nun generelle Einstellungen für den LDAP-Server vorgenommen wurden, dienen die folgenden Parameter dazu, die Art und das Verhalten der Datenbank zu bestimmen.

Der Parameter `database`

Mit dem Parameter `database` wird das Format der X.500-Datenbank definiert:

```
database <Format>
```

Der Server der OpenLDAP-Organisation unterstützt zwei verschiedene Formate, jedoch sollte grundsätzlich `ldbm` verwendet werden:

```
database ldbm
```

Diese Einstellung ist auch in der Konfigurationsdatei vorhanden, die bei der Installation der Software erstellt wird.

Der Parameter `lastmod`

Der LDAP-Server ist in der Lage, vier Attribute automatisch zu pflegen. Diese sind in Tabelle 4.10 zusammengefasst:

Attribut	Beschreibung
`creatorsName`	Name des Objekterstellers
`createTimestamp`	Zeitpunkt der Erstellung
`modifiersName`	Name des Objektbearbeiters
`modifyTimestamp`	Zeitpunkt der Bearbeitung

Tabelle 4.10: Automatisch geführte Attribute

Falls diese Eigenschaften vom System mit Werten gefüllt werden sollen, so muss dieser Wunsch über den Parameter `lastmod` mitgeteilt werden:

```
lastmod <on|off>
```

Per Default ist diese Funktion ausgeschaltet:

```
lastmod off
```

Falls mehrere Administratoren die Datenbank pflegen, ist diese Funktion jedoch sehr hilfreich. Die Zeitangaben erfolgen dann gemäß der Syntax:

```
<Jahr><Monat><Tag><Stunde><Minute><Sekunde>
```

Die Jahreszahl wird vierstellig, die anderen Werte werden zweistellig aufgeführt.

```
20001108075430
```

Der Parameter `cachesize`

Über diesen Eintrag wird definiert, wie viele Elemente der Datenbank im Hauptspeicher zwischengespeichert werden. Er wird nach der Syntax

```
cachesize <Wert>
```

verwendet. Standardmäßig werden 1000 Einträge gecacht.

```
cachesize 1000
```

Der Parameter `dbcachesize`

Neben der Anzahl der Einträge im Cache kann auch dessen Größe in Bytes angegeben werden:

```
dbcachesize <Wert>
```

Wird dieser Wert nicht bestimmt, so werden 100 000 Bytes angenommen:

```
dbcachesize 100000
```

Der Parameter `dbcachenowsync`

Mit diesem Parameter wird angegeben, dass Änderungen an der Datenbank, die zunächst im Cache gehalten werden, nicht sofort mit den Daten auf der Festplatte synchronisiert werden. Die Zugriffsgeschwindigkeit erhöht sich zwar in diesem Fall, dieses geht jedoch auf Kosten der sicheren Datenhaltung.

```
dbcachenowsync
```

Dieser Parameter wird in der Regel nicht verwendet.

Der Parameter `directory`

Bisher wurde die Datenbank anhand von mehreren Einstellungen näher spezifiziert. Eine sehr wichtige Definition beschreibt das Verzeichnis, in dem die Dateien der LDBM-Datenbank abgespeichert werden sollen. Es ist nach dem Schema

```
directory <Verzeichnisname>
```

anzugeben. In unserem Beispiel liegen die Daten in `/var/lib/ldap/`. Es sei darauf hingewiesen, dass das angegebene Verzeichnis unbedingt bereits vor dem Start des Servers existieren muss.

```
directory /var/lib/ldap
```

Wird dieser Parameter nicht explizit in der Konfigurationsdatei `slapd.conf` aufgeführt, so wird das Verzeichnis `/var/run/openldap-ldbm/` für die Speicherung der Daten verwendet.

Der Parameter `index`

Die Suche nach Attributen im Verzeichnisbaum erfolgt im Prinzip sequenziell und kann je nach Größe der Datenbank mitunter sehr lange dauern. Dieses kann durch Verwendung von Indexdateien zum Teil wesentlich verbessert werden. So ist es mit dem folgenden Parameter möglich, die Index-Dateien vom System anlegen und pflegen zu lassen:

```
index <Attribute|default> <pres|eq|approx|sub|none>
```

Auf diese Weise können für die angegebenen Attribute oder für eine von der Software vorgegebene Defaultliste Indexdateien erstellt werden, die zum Beispiel für eine genaue (`eq`) oder eine annähernde (`approx`) Suche verwendet werden:

```
index cn,sn eq,approx
```

Auch dieser Parameter ist im Prinzip nicht notwendig, da die per Standard vorgegebene Indizierung ausreichend ist.

Der Parameter `mode`

Zum Abschluss der allgemeinen Einstellungen in der Konfigurationsdatei zum Thema Datenbank sei die `mode`-Definition genannt. Sie gibt die Rechtestruktur für vom System neu erstellte Indexdateien an. Für die Verwendung der Form

```
mode <Rechte>
```

wird per Default der Wert 0600 angenommen. Dadurch hat der Eigentümer der Datei Lese- und Schreibrechte. Alle anderen Anwender haben auf die Indexdateien keine Zugriffsrechte:

```
mode 0600
```

Partitionierung

Die bisherigen Parameter haben Auswirkungen auf das generelle Verhalten sowohl des LDAP-Servers als auch der Datenbank. In diesem Abschnitt werden zwei Parameter beschrieben, die lediglich für die Partitionierung relevant sind.

Der Parameter `suffix`

Als Erstes sei der Parameter mit dem Namen `suffix` genannt. Mit ihm wird der Distinguished Name des Containers beschrieben, ab dem der LDAP-Server die Daten gespeichert hat. Anders formuliert bedeutet dies, dass der Server Anfragen, die auf den angegebenen Container enden (suffix=Nachsilbe), selbst anhand seiner Datenbank zu beantworten versucht. Die Definition dieser Nachsilbe erfolgt nach der Syntax:

```
suffix <dn>
```

Für jeden LDAP-Server ist es notwendig, mindestens eine Definition des Suffixes vorzunehmen. Es können jedoch mehrere Anweisungen vorhanden sein. Dieses würde bedeuten, dass der Server Anfragen behandelt, die auf mehrere DNs enden.

Im Beispiel

```
suffix "o=Intern"
```

werden alle Anfragen behandelt, die sich auf die Organisation `Intern` beziehen. Falls in der Angabe des Distinguished Name Leerzeichen auftauchen, so kann der ganze Ausdruck mit Anführungszeichen (`"ou=Verkauf, o=Intern"`) umschlossen werden.

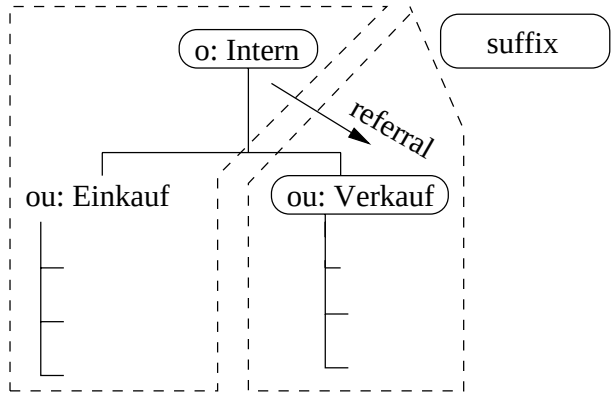

Abbildung 4.3: `suffix` und `referral`

Der Parameter `referral`

Falls eine Client-Anfrage aufgrund des falschen Suffixes nicht anhand der lokalen

Datenbank beantwortet werden kann, gibt es die Möglichkeit, diese Anfrage an einen anderen LDAP-Server weiterzuleiten. Dieser Server wird über die Struktur

```
referral ldap://<Hostname>
```

angegeben. Sie muss in den globalen Optionen am Anfang der Konfigurationsdatei stehen, da sie von der zugrunde liegenden Datenbank unabhängig ist. Beispiel:

```
referral ldap://host2.intern
```

Da es mit dem Parameter `referral` nur möglich ist, unabhängig von der Position im Verzeichnisbaum auf einen anderen Server zu zeigen, wird dieses Verfahren in der Praxis eher selten eingesetzt. Außerdem hängt es vom Client ab, ob er den mit `referral` angegebenen Rechner konsultiert. Besser ist es, Querverweise an Containern als Attribut zu definieren (siehe Abschnitt 4.5.2 und 4.5.6)

Replizierung

Neben der Partitionierung bildet die Replizierung ein wichtiges Mittel, um die Ausfallsicherheit und die Zugriffsgeschwindigkeit von LDAP-Diensten zu erhöhen. Dazu werden die Daten eines Rechners auf einen oder mehrere andere Rechner dupliziert.

Der Mechanismus der Replizierung gliedert sich in drei Schritte:

1. Der Replica-Server wird einmalig mit der Datenbank des Servers initialisiert. Dieses kann mit dem Kommando `ldif2ldbm` erfolgen, indem auf diese Weise ein Abzug des Masters auf dem Replica-Server eingerichtet wird.
2. Der Master muss nun dazu angeregt werden, alle Änderungen in einer Logdatei zu protokollieren.
3. Diese Logdatei wird dann vom Dämon `slurpd` verwendet, um die Änderungen auch auf dem Replica-Rechner umzusetzen.

Der Parameter `replogfile`

Die Logdatei für die Replizierung wird vom Master dann erstellt, wenn in der Konfigurationsdatei `slapd.conf` vermerkt wird, unter welchem Namen sie existieren soll.

```
replogfile <Dateiname>
```

Da es sich um eine Datei mit variablem Inhalt handelt, könnte sie zum Beispiel im Verzeichnis `/var/run/` abgelegt werden:

```
replogfile /var/run/slapd.replog
```

Der Parameter `replica`

Nachdem der Master-Server nun dazu angeregt wurde, alle Änderungen in der Logdatei für die Replizierung zu protokollieren, erfolgt in der Konfigurationsdatei noch eine Definition des Replica-Servers. Dabei ist zum einen dessen IP-Adresse oder dessen DNS-Name anzugeben. Ferner wird der TCP-Port definiert. Auch ist es notwendig anzugeben, mit welchem Namen und welchem Passwort sich der Master bei der Replica-Datenbank anmelden kann:

```
replica host=<Hostname>:<Port>
        binddn=<dn>
        bindmethod=simple
        credentials=<Passwort>
```

Als `binddn` ist das Objekt auf dem Replica-Server anzugeben, zu dem sich der Master verbinden soll. Es muss entsprechend die Rechte in der Datenbank besitzen, um die Änderungen auch durchführen zu können. Mit `bindmethod` ist die Art der Beglaubigung anzugeben. Da die Sicherheit der Datenübertragung im weiteren Verlauf dieses Buches mit der Secure Shell (SSH) realisiert wird, wird hier die Methode `simple` gewählt. Sie steht für eine Beglaubigung mit einem Passwort. Als Alternative wäre Kerberos möglich. Letztlich ist über `credentials` das Passwort im Klartext anzugeben. Im nächsten Abschnitt wird bei der Definition der Zugriffsrechte erwähnt, wie dass Passwort auch verschlüsselt aufgeführt werden kann.

```
replica host=192.168.17.2:389
        binddn="cn=Admin, o=Intern"
        bindmethod=simple
        credentials=linux
```

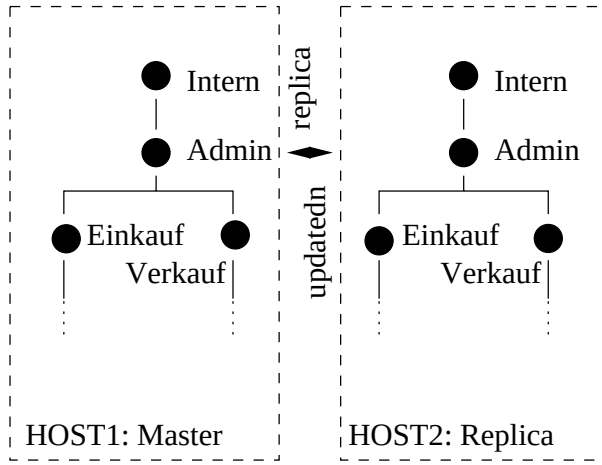

Abbildung 4.4: `replica` und `updatedn`

Der Parameter `updatedn`

Die Einstellungen am Master der Replizierung sind mit den zuvor genannten zwei Parametern abgeschlossen. Schließlich ist noch der Replica-Server dahingehend vorzubereiten, dass er das inkrementelle Update entgegennehmen kann. Der Replica-Server ist im Prinzip genauso ein LDAP-Server wie der Master. Falls er jedoch Informationen des Masters in seine Datenbank aufnehmen soll, so ist in dessen Konfigurationsdatei `slapd.conf` festzuhalten, mit welchem Distinguished Name sich der Master anmelden kann. Dies kann mit der folgenden Syntax geschehen:

```
updatedn <dn>
```

Die Angabe des `dn` bezieht sich auf die Datenbank der Replica. Sie muss mit der `dn` des Parameters `replica` am Master übereinstimmen. Beispiel:

```
updatedn "cn=Admin, o=Intern"
```

Zugriffsrechte

Bei den Zugriffsrechten muss man zwischen zwei Bereichen von Parametern unterscheiden.

- Parameter, die den Administrator der Datenbank beschreiben
- Parameter, die die Rechte auf Objekte und Attribute definieren

Der Administrator der Datenbank wird mit zwei Parametern in der Konfigurationsdatei festgelegt.

Der Parameter `rootdn`

Über die Option `rootdn` wird der Distinguished Name des Verzeichnisdienst-Administrators angegeben. Da der Administrator derjenige ist, der (ohne Einschränkung von Rechten) die Objekte in der Datenbank erstellt, folgt daraus, dass er sich am DIT anmelden muss, bevor er dort überhaupt existiert. Deshalb ist es nicht notwendig, dass der dem Parameter mitgeteilte Wert existiert:

```
rootdn <dn>
```

Es sollte jedoch zu den ersten Aktionen gehören, ein entsprechendes Objekt zu erstellen, damit der Verzeichnisdienst mit den realen Gegebenheiten übereinstimmt. Ein weiterer Vorteil dieses administrativen Zugriffs ist, dass sich der Administrator nicht selbst vom Zugriff ausschließen kann. Löscht er versehentlich sein Objekt im Directory Information Tree, so hat er weiterhin alle Rechte auf die Datenbank.

```
rootdn "cn=Admin, o=Intern"
```

Der Parameter `rootpw`

Zusätzlich ist es natürlich notwendig, den Zugriff auf die Datenbank über das Objekt des Administrators mit einem Passwort zu schützen. Für die Art und Weise, wie Passwörter im Verzeichnisdienst angegeben werden, existieren vier Möglichkeiten:

- Angabe des Passwortes im Klartext
- Angabe eines mit SHA codierten Passwortes
- Angabe eines mit MD5 codierten Passwortes
- Angabe eines mit CRYPT (Linux-Standard) codierten Passwortes

Bei der Syntax, in der das Passwort des Administrators angegeben werden kann, ist eine der folgenden Varianten zu wählen:

```
rootpw <Passwort>
rootpw {SHA}<Passwort>
rootpw {MD5}<Passwort>
rootpw {CRYPT}<Passwort>
```

Bei der Verschlüsselung `{CRYPT}` handelt es sich um ein Passwort, das das Kodierungsverfahren verwendet, das auch generell unter Linux für die Passwörter in der Datei `/etc/shadow` benutzt wird. Ein Passwort im Klartext würde als

```
rootpw linux
```

eingegeben. Die folgende Zeile enthält das Geheimwort nicht mehr im Klartext, ist aber mit der vorigen identisch.

```
rootpw {CRYPT}zsOQLoUVZvS6E
```

Es sei darauf hingewiesen, dass in diesem Buch lediglich mit dem Passwort `linux` gearbeitet wird. In der Praxis muss man natürlich ein sicheres Wort verwenden, welches sich am besten aus den jeweils ersten Buchstaben der Wörter eines Satzes zusammensetzt.

Der Parameter `defaultaccess`

Zugriffsrechte auf den Verzeichnisdienst können für jedes Objekt und jedes Attribut definiert werden. Dabei kann genau entschieden werden, wer welche Aktionen ausführen darf. Um die Realisierung der gewünschten Rechtestruktur umzusetzen, können spezielle Regeln definiert werden. Versucht ein Anwender, eine Aktion an der Datenbank auszuführen, so werden die einzelnen Regeln geprüft. Falls keine der Definitionen zutrifft, kommt eine Defaultregel zur Anwendung. Sie wird über den Parameter

```
defaultaccess <none|compare|search|read|write|delete>
```

definiert (siehe auch Tabelle 4.11). Hierbei es ist möglich, per Default keine Zugriffe zu erlauben (`none`), lediglich Vergleiche zuzulassen (`compare`), das Suchen zu erlauben (`search`) oder insgesamt den lesenden Zugriff (`read`) zu gestatten. `read` beinhaltet `compare` und `search`. Das Schreibrecht (`write`) schließt alle zuvor genannten Rechte mit ein. Mit (`delete`) können Einträge entfernt werden.

Recht	Beschreibung
`none`	Keine Rechte
`compare`	Vergleichen
`search`	Suchen
`read`	Lesen
`write`	Schreiben
`delete`	Löschen

Tabelle 4.11: Rechte

Fehlt die Definition des Standardzugriffs, so kommt `read` zur Anwendung. Dadurch haben alle Anwender lesende Rechte auf die Objekte und Attribute im Verzeichnisbaum:

```
defaultaccess read
```

Die Deklaration `access`

Mit dem Parameter `defaultaccess` wurde bisher definiert, dass alle Anwender des Verzeichnisdienstes auf alle Objekte lesenden Zugriff haben. Falls für einige Objekte spezielle Rechte festgelegt werden müssen, die vom Defaultwert abweichen, so kann dies über die folgende Deklaration geschehen, die gemäß der Syntax

```
access to <Objekt>
       by <Objekt> <Recht>
       by <Objekt> <Recht>
       by <…>    <…>
```

zu benutzen ist. Mit der `access`-Anweisung, die mehrfach in der Konfigurationsdatei `slapd.conf` vorkommen kann, wird zu einem Objekt oder einem Attribut eine Access Control List (ACL) definiert. Sie beschreibt die genauen Zugriffsmöglichkeiten auf das Element. Falls vom Client aus der Zugriff auf eine Ressource im Verzeichnisdienst erfolgen soll, so werden die per `access` definierten Regeln der Reihe nach abgearbeitet. Trifft eine Regel auf den Zugriffswunsch zu, so kommt sie zur Anwendung. Eventuell vorhandene weitere Regeln, die ebenfalls anwendbar wären, werden ignoriert.

In der obigen Syntax kann für `<Objekt>` der Distinguished Name eines Objektes eingesetzt werden. Ferner ist es möglich, das Objekt mit einem Attribut zu kombinieren und durch die Verwendung von Mustern (zum Beispiel `*`) mehrere Objekte gleichzeitig anzusprechen:

```
access to attr=userPassword
       by self write
       by * none

access to attr=telephoneNumber
       by self write
       by * read

access to dn="cn=Admin,o=Intern"
       by * none

access to *
       by * read
```

Die erste Definition besagt, dass das Attribut `userPassword` des eigenen Objekts verändert werden kann. Das Schlüsselwort `self` hinter `by` beschreibt das eigene Objekt. Alle anderen Einträge im Verzeichnisdienst bekommen keine Rechte auf das Passwort eines Benutzers. `none` bedeutet, dass andere Objekte gar nicht sehen, dass ein Passwort-Attribut vorhanden ist.

Das Verhalten bei der Telefonnummer ist ähnlich wie bei dem Passwort, nur dass alle Anwender lesenden Zugriff auf das Attribut haben.

Die dritte Regel legt fest, dass alle Anwender keinen Zugriff auf das Objekt des Administrators besitzen. Dadurch bleibt ihnen die Existenz von `cn=Admin, o=Intern` verborgen. Die letzte Regel definiert sicherhaltshalber nochmals den Standard.

Mit den obigen Regeln wird der über `rootdn` angegebene Administrator keineswegs beeinträchtigt, da die Zugriffsbeschränkungen für ihn nicht gelten.

Die genannten Rechte bilden eine praktische Realisierung für einen Master-Server. Am Replica-Host sollten lediglich lesende Rechte vorhanden sein, da er nur eine Kopie des Masters ist. Anderenfalls wäre es möglich, zwischen Master und Replica einen inkonsistenten Zustand herzustellen, da die Replica Daten enthalten könnte, die auf dem Master nicht vorhanden sind. Falls doch Änderungen an den Daten des Replica-Servers vorgenommen werden sollen, so müssen diese dem Master mitgeteilt werden.

Es gilt demnach, zwischen

- Read Only Replica (RO) und
- Read Write Replica (RW)

zu unterscheiden. Welche in der Praxis eher anzuwenden ist, hängt vom Einzelfall ab. Eine generelle Aussage kann nicht getroffen werden. Meistens wird es sich jedoch um den RO-Typ handeln.

Der Parameter `readonly`

Um die Datenbank generell in den Modus ‚Read Only' zu versetzen, kann der folgende Parameter verwendet werden:

```
readonly <on|off>
```

Er ist per Standard ausgeschaltet:

```
readonly off
```

Beispiel

Mit den bekannten Parametern soll nun ein komplettes Beispiel angegeben werden. Dabei handelt es sich um einen Verzeichnisdienst, der mit der Organisation `Intern` beginnt. Der Rechner `HOST1` dient als Master (IP-Adresse `192.168.17.1`), `HOST2` mit der Adresse `192.168.17.2` als Replica. Eine Partitionierung findet nicht statt.

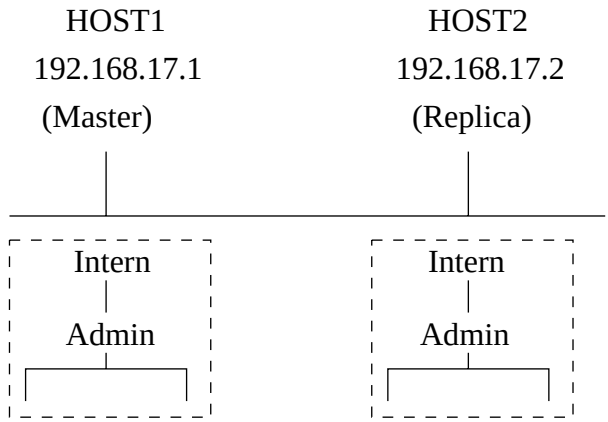

Abbildung 4.5: Beispiel einer LDAP-Konfiguration

Als Erstes betrachten wir die Konfigurationsdatei des Masters. Sie befindet sich bei einer SuSE-Distribution im Verzeichnis `/etc/openldap/`.

Beispiel 4.2: Konfigurationsdatei des Masters

```
root@host1:~ # cat /etc/openldap/slapd.conf
#
# Konfigurationsdatei des LDAP-Servers (Master)
```

```
#
# slapd.conf
#
# Erstellt von Jens Banning
#

#
# Allgemeines
#

include         /etc/openldap/slapd.at.conf
include         /etc/openldap/slapd.oc.conf

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

loglevel        256
schemacheck     on

sizelimit       500
timelimit       3600

#
# Datenbank
#

database        ldbm
lastmod         off

cachesize       1000
dbcachesize     100000

directory       /var/lib/ldap
suffix          "o=Intern"

rootdn          "cn=Admin, o=Intern"
rootpw          {CRYPT}zsOQLoUVZvS6E

#
# Replizierung
#

replogfile      /var/run/slapd.replog
replica         host=192.168.17.2:389
                binddn="cn=Admin, o=Intern"
                bindmethod=simple credentials=linux

#
```

```
# Rechte
#

defaultaccess   read

access to attr=userPassword
       by self write
       by * none

access to attr=telephoneNumber
       by self write
       by * read

access to dn="cn=Admin,o=Intern"
       by * none

access to *
       by * read

root@host1:~ #
```

Zu Beginn werden über `include` zwei weitere Dateien in die Konfiguration des LDAP-Servers mit eingebunden. Es handelt sich zum einen um Objekt- und zum anderen um Attributdefinitionen, die in den nächsten beiden Abschnitten dieses Buches beschrieben werden. Über `pidfile` und `argsfile` wird jeweils eine Datei angegeben, in der die Prozesskennung bzw. der genaue Programmaufruf des Dämons `slapd` festgehalten wird. Der Loglevel wird auf 256 gesetzt, und die Einträge im Verzeichnisdienst werden mit dem definierten Schema abgeglichen. Letztlich liefert eine Anfrage an den Server maximal 500 Ergebnisse zurück. Der Server verwendet für eine Client-Anfrage nicht mehr als eine Stunde.

Diese allgemeinen Einstellungen müssen am Anfang der Konfigurationsdatei stehen. Ihnen folgen Angaben zur Datenbank. Sie wird im Format LDBM verwaltet. Die automatische Aktualisierung von Attributen, die Änderungen am Objekt protokollieren, wird nicht durchgeführt (`lastmod off`). Es werden 1000 Einträge und 100 Kbyte Daten im Cache gehalten. Die Datenbank beginnt mit der Organisation `Intern` und ist im Verzeichnis `/var/lib/ldap/` abgespeichert. Administrativ kann auf die Datenbank mit dem Distinguished Name `cn=Admin, o=Intern` zugegriffen werden. Es ist dazu das Passwort `linux` zu verwenden.

Der Master soll alle Änderungen an seiner Datenbank für die Replizierung protokollieren. Dies geschieht in der Datei `/var/run/slapd.replog`. In dieser Datei wird auch festgelegt, zu welchem Rechner die Daten später mit dem Dämon `slurpd` dupliziert werden sollen. Es ist die IP-Adresse `192.168.17.2`. Dort ist der LDAP-Server am TCP-Port 389 anzusprechen. Die Beglaubigung an der Replica erfolgt mit `cn=Admin, o=Intern`. Das Passwort ist `linux`.

Der administrative Benutzer hat unabhängig von den definierten Zugriffsregeln alle Rechte im gesamten Verzeichnisdienst. Alle anderen Objekte haben, sofern nicht anders definiert, lediglich Leserechte auf alle Einträge im Verzeichnisdienst. Ihr eigenes Passwort und ihre Telefonnummer dürfen sie jedoch ändern. Fremde Anwender können die Telefonnummer, nicht jedoch das Passwort einsehen. Ferner haben alle Objekte keinen Zugriff auf das Administrator-Objekt. Dadurch ist es anderen Anwendern auch nicht möglich, dieses Objekt überhaupt zu sehen. Zum Schluss der `access`-Deklarationen wird der Defaultwert nochmals aufgeführt.

Insgesamt ist in dieser Datei darauf zu achten, dass die Parameter und Deklarationen in einer gewissen Reihenfolge auftreten müssen. Es ist daher ratsam, wie im Beispiel

- Allgemeines,
- Datenbank,
- Replizierung und
- Rechte

in dieser Reihenfolge aufzuführen. Anderenfalls könnten Eintragungen nicht in die Konfiguration einfließen.

Die zugehörige Konfigurationsdatei des Replica-Servers hat die in Beispiel 4.3 erwähnten Einstellungen.

Beispiel 4.3: Konfigurationsdatei des Replica-Servers

```
root@host2:~ # cat /etc/openldap/slapd.conf
#
# Konfigurationsdatei des LDAP-Servers (Replica)
#
# slapd.conf
#
# Erstellt von Jens Banning
#

#
# Allgemeines
#

include         /etc/openldap/slapd.at.conf
include         /etc/openldap/slapd.oc.conf

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

loglevel        256
schemacheck     on
```

```
sizelimit       500
timelimit       3600

#
# Datenbank
#

database        ldbm
lastmod         off

cachesize       1000
dbcachesize     100000

directory       /var/lib/ldap
suffix          "o=Intern"

rootdn          "cn=Admin, o=Intern"
rootpw          {CRYPT}zsOQLoUVZvS6E

#
# Replizierung
#

updatedn        "cn=Admin, o=Intern"

#
# Rechte
#

defaultaccess   read

access to attr=userPassword
       by self read
       by * none

access to dn="cn=Admin,o=Intern"
       by * none

access to *
       by * read

root@host2:~ #
```

Die Konfigurationsdatei des Replica-Servers stimmt bis auf zwei Bereiche mit der des Masters überein. Für die Kopie der Datenbank sind andere Einstellungen im Bereich der

- Replizierung und der
- Rechte

notwendig.

Im Abschnitt der Replizierung ist anzugeben, mit welchem Objekt sich der Master-Server bei der Replica anmelden möchte. Es ist nur eine Replizierung über dieses Objekt möglich. Der Distinguished Name, der auf `updatedn` folgt, muss mit dem Wert für `binddn` in der `replica`-Definition des Masters übereinstimmen.

Im Beispiel handelt es sich um eine Read-Only-Kopie der Verzeichnisdatenbank. Das bedeutet, dass Clients die Replica lediglich als Informationsquelle benutzen können. Änderungen sind nicht möglich. Neben dem per Standard definierten lesenden Zugriff wird angegeben, dass jeder Benutzer (jedes Objekt) ausschließlich sein eigenes Passwort lesen kann. Das administrative Objekt `cn=Admin, o=Intern` bleibt allen Anwendern verborgen.

4.5.2 Die Datei `slapd.oc.conf`

Wie in dem zuvor genannten Beispiel wurden in die Konfigurationsdatei zwei weitere Dateien eingebunden. Dieses geschah aus Gründen der Übersichtlichkeit. In der ersten Datei mit dem Namen `slapd.oc.conf` sind alle Objektklassen definiert. Die `objectclass`-Definitionen geben an, welche Objekte im Verzeichnisdienst möglich sind und welche Attribute sie aufweisen können bzw. müssen. Letztlich erfolgt in dieser Datei die Festlegung des Schemas.

Da die Datei sehr umfangreich ist, werden an dieser Stelle lediglich vier Objekte vorgestellt. Es handelt sich um

- die Organisation,
- die organisatorische Einheit,
- die Person und
- den Querverweis auf eine andere Partition.

Beispiel 4.4: Organisation

```
objectclass organization
        requires
                objectClass,
                o
        allows
                businessCategory,
                description,
```

```
                destinationIndicator,
                facsimileTelephoneNumber,
                internationaliSDNNumber,
                l,
                physicalDeliveryOfficeName,
                postOfficeBox,
                postalAddress,
                postalCode,
                preferredDeliveryMethod,
                registeredAddress,
                searchGuide,
                seeAlso,
                st,
                streetAddress,
                telephoneNumber,
                teletexTerminalIdentifier,
                telexNumber,
                userPassword,
                x121Address
```

Die Objektklasse der Organisation benötigt zwei Attribute. Es handelt sich einerseits um den Namen der Objektklasse. Er wird also nochmals als Attribut festgehalten. Anhand dieser Eigenschaft können grafische Anwendungen das Objekt mit einem Symbol darstellen. Ferner muss das Attribut `o` definiert werden. Es enthält den Namen der Organisation (zum Beispiel `Intern`). Alle unter `allows` aufgeführten Eigenschaften können optional zu einem Objekt dieser Klasse angegeben werden. Sie sind jedoch nicht notwendig. Beachten Sie, dass dem Objekt kein Attribut gegeben werden kann, das nicht in dieser Deklaration aufgeführt ist. Das Schema wäre sonst verletzt.

In den bisherigen Beispielen war meistens eine Organisation vorhanden, die in organisatorische Einheiten unterteilt war (siehe Beispiel 4.5).

Beispiel 4.5: Organisatorische Einheit

```
objectclass organizationalUnit
        requires
                objectClass,
                ou
        allows
                businessCategory,
                description,
                destinationIndicator,
                facsimileTelephoneNumber,
                internationaliSDNNumber,
                l,
                physicalDeliveryOfficeName,
                postOfficeBox,
```

```
                postalAddress,
                postalCode,
                preferredDeliveryMethod,
                registeredAddress,
                searchGuide,
                seeAlso,
                st,
                streetAddress,
                telephoneNumber,
                teletexTerminalIdentifier,
                telexNumber,
                userPassword,
                x121Address
```

Neben der `objectClass` ist ferner die Eigenschaft `ou` anzugeben. Sie könnte den Wert `Verkauf` haben. Alle anderen Attribute sind optional.

Ein wesentliches Merkmal von Verzeichnisdiensten ist die Repräsentation von Personen, die im Netzwerk als Benutzer tätig sind (siehe Beispiel 4.6).

Beispiel 4.6: Person

```
objectclass person
        requires
                objectClass,
                sn,
                cn
        allows
                description,
                seeAlso,
                telephoneNumber,
                userPassword
```

Blattobjekte werden generell mit `cn` bezeichnet, was der Klasse Person als Eigenschaft entsprechend mitgeteilt werden muss (`FlemmingJ`). Neben der Objektklasse ist es auch notwendig, den Nachnamen der Person über das Attribut `sn` (Surname) zu definieren. Das `userPassword` ist nicht zwingend erforderlich.

Abschließend sei ein Objekt erwähnt, das für die Partitionierung verwendet wird (siehe Beispiel 4.7).

Beispiel 4.7: Querverweis

```
objectclass referral
        requires
                ref,
                objectClass
```

Dieses Objekt dient dazu, einen Verweis auf eine andere Partition des Verzeichnisdienstes herzustellen. Neben der Objektklasse wird ferner der eigentliche Querverweis angegeben. Diese beiden Attribute sind unbedingt und ausschließlich aufzuführen.

Bevor ein Beispiel gegeben werden kann, sei nochmals darauf hingewiesen, dass ein Objekt mehreren Klassen angehören kann. Die Definitionen für `requires` und `allows` ergänzen sich dann entsprechend. Würde demnach ein Objekt erstellt, das den Klassen ‚organisatorische Einheit' und ‚Querverweis' angehört, so müssen die folgenden Attribute zwingend definiert werden:

```
requires
        objectClass,
        ref,
        ou
```

Für die optionalen Eigenschaften gilt das gleiche Kombinationsverfahren.

Falls sich zum Beispiel die Daten des Verkaufs auf einer anderen Partition befinden, so ist ein Objekt mit dem Namen `Verkauf` zu erstellen, das beiden zuvor erwähnten Klassen angehört. Die Eigenschaft `ou` hat den Wert `Verkauf`, und `ref` würde der Wert

```
ldap://host2.intern/ou=Verkauf, o=Intern
```

übergeben. Dadurch ist der Verweis auf die zweite Partition (sie hat das `suffix` `ou=Verkauf`) erzeugt worden. Dieser Verweis kann dann von dem Kommando, das zum Suchen in der Datenbank verwendet wird, analysiert und verfolgt werden. Werden die Daten eines Querverweises mit einer eigenen Funktion oder einem

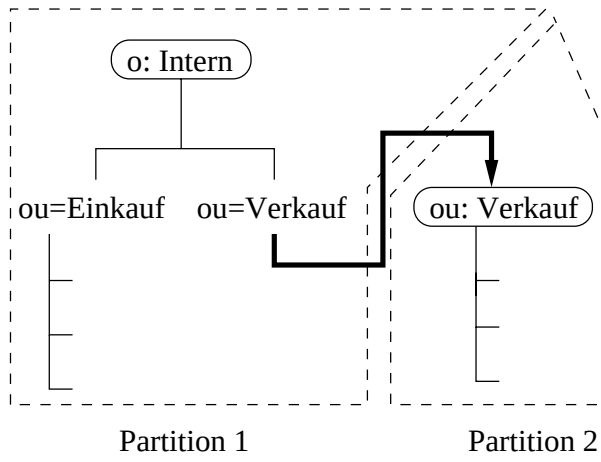

Abbildung 4.6: Beispiel für einen Querverweis

eigenen Skript analysiert, so kann die Angabe des Zielrechners natürlich auch gemäß einer anderen Syntax erfolgen.

Sofern in den ausführlichen Beispielen in Kapitel 5 ab Seite 177 weitere Objekte benutzt werden, werden sie zuvor angegeben.

4.5.3 Die Datei `slapd.at.conf`

In der Datei `slapd.oc.conf` sind alle Objekte des Verzeichnisdienstes mit den dazugehörigen Attributen definiert. Bisher wurden diese Attribute jedoch nicht näher spezifiziert. Mit der `attribute`-Deklaration ist es möglich, die Eigenschaften von Objekten bezüglich deren Syntax näher zu beschreiben. Aus Gründen der Übersichtlichkeit werden diese Anweisungen in einer separaten Datei abgelegt, die dann über den Parameter `include` in die Konfigurationsdatei `slapd.conf` eingebunden wird. Diese Datei trägt den Namen `slapd.at.conf`.

Die Deklarationen, die dort vorhanden sind, werden im Folgenden dargestellt und anschließend beschrieben.

Beispiel 4.8: Syntax der Attribute

```
root@host1:~ # cat /etc/openldap/slapd.at.conf
attribute       photo                                   bin
attribute       personalsignature                       bin
attribute       jpegphoto                               bin
attribute       audio                                   bin
attribute       labeledurl                              ces
attribute       ref                                     ces
attribute       userpassword                            ces
attribute       telephonenumber                         tel
attribute       facsimiletelephonenumber        fax     tel
attribute       pagertelephonenumber            pager   tel
attribute       homephone                               tel
attribute       mobiletelephonenumber           mobile  tel
attribute       aliasedObjectName                       dn
attribute       member                                  dn
attribute       owner                                   dn
attribute       seealso                                 dn
attribute       manager                                 dn
attribute       documentauthor                          dn
attribute       secretary                               dn
attribute       lastmodifiedby                          dn
attribute       associatedname                          dn
attribute       naminglink                              dn
attribute       reciprocalnaminglink                    dn
attribute       dn                                      dn
root@host1:~ #
```

In der ersten Spalte ist das Schlüsselwort `attribute` zu finden. Ihm folgt der Name des Attributs, dessen Syntax definiert werden soll. Es ist außerdem möglich, einen zweiten, alternativen Begriff anzugeben. So kann die `facsimiletelephonenumber` auch mit dem Wort `fax` angesprochen werden. In der letzten Spalte folgt dann eine der Zeichenketten

- `bin` (binär),
- `ces` (Groß- und Kleinschreibung wird unterschieden),
- `dn` (Distinguished Name) oder
- `tel` (Telefonnummer).

Falls es sich bei den Attributwerten um Texte handelt, deren Groß- und Kleinschreibung nicht unterschieden wird (`cis`), so müssen diese Eigenschaften nicht mit der Deklaration `attribute` erwähnt werden, da diese Syntax standardmäßig verwendet wird.

Im Beispiel bedeutet dies, dass bei `photo` die Werte binär angegeben werden. Das `userPassword`-Attribut ist ein Text, der zwischen Groß- und Kleinschreibung unterscheidet (case exact string). Hinter der `telephonenumber` befindet sich der Typ `tel`. Außerdem sind noch einige Attribute von der Art `dn` angegeben. So kann der Wert der Eigenschaft `manager` zum Beispiel den Distinguished Name des zuständigen Mitarbeiters haben (`cn=FlemmingJ, ou=Verkauf, o=Intern`).

Alle in dieser Datei nicht aufgeführten Attribute sind von der Syntax `cis` (case ignore string).

4.5.4 Der Dämon `slapd`

Das zentrale Element des Verzeichnisdienstes ist der LDAP-Server. Er trägt den Namen `slapd`. Die einfachste Möglichkeit, den Server zu starten, besteht darin, das vorhandene Startskript zu nutzen. Es befindet sich in dem Verzeichnis `/etc/rc.d/init.d/` und trägt bei einer SuSE-Distribution den Namen `ldap`. Durch die beiden folgenden Aufrufe kann der Server gestartet bzw. gestoppt werden.

Beispiel 4.9: Starten des LDAP-Servers

```
root@host1:~ # /etc/rc.d/init.d/ldap start
Starting ldap-server.                                    done
root@host1:~ #
```

Beispiel 4.10: Stoppen des LDAP-Servers

```
root@host1:~ # /etc/rc.d/init.d/ldap stop
Shutting down ldap-server.                               done
root@host1:~ #
```

Damit das Skript permanent beim Rechnerstart aktiviert wird, ist bei einer SuSE-Distribution die Variable `START_LDAP` mit dem Programm `yast` auf `yes` zu setzen.

Das Skript ruft den eigentlichen Server-Prozess auf. Es verwendet dazu in der Regel keine weiteren Optionen, wie in Beispiel 4.11 zu sehen ist.

Beispiel 4.11: Auszug aus dem Startskript

```
root@host1:~ # cat /etc/rc.d/init.d/ldap
…
case "$1" in
  start)
    echo -n "Starting ldap-server."
    /sbin/startproc /usr/lib/openldap/slapd || return=$rc_failed
    echo -e "$return"
…
root@host1:~ #
```

Der allgemeine Aufruf des LDAP-Servers aus `/usr/lib/openldap` erfolgt in der Form:

```
slapd [Optionen]
```

Dazu stehen die Optionen zur Verfügung, die in Tabelle 4.12 aufgelistet sind.

Option	Beschreibung
`-d <Level>`	Debug-Level
`-f <Datei>`	Konfigurationsdatei
`-l <Text>`	Syslog-User
`-p <Port>`	Server-Port
`-s <Level>`	Syslog-Level

Tabelle 4.12: Optionen des LDAP-Servers `slapd`

Wenn der LDAP-Dämon aufgerufen wird, begibt er sich automatisch in den Hintergrund, ohne die Textkonsole zu blockieren. Über den Parameter `-d` wird er angeregt, sich in den Debugging-Modus zu versetzen. Dabei ist über den Wert, der auf die Option folgt, anzugeben, welche Informationen auf der Textkonsole ausgegeben werden sollen. Es sind die Werte zu benutzen, die auch für den Wert `loglevel` in der Konfigurationsdatei `slapd.conf` möglich waren (siehe Seite 56). Eine Übersicht der möglichen Werte kann auch mit `-d ?` eingesehen werden. Der Dämon versetzt sich beim Debuggen nicht in den Hintergrund, sondern bleibt in der startenden Shell aktiv.

Eine ähnliche Funktion hat die Option `-s`. Ihr wird ebenfalls ein Wert übergeben, der den Grad der Protokollierung beschreibt. Die so definierten Meldungen werden dem

`syslogd` übergeben. Er schreibt sie per Standard in die Datei `/var/log/messages`. Der `syslogd` ist über die Datei `syslog.conf` im Verzeichnis `/etc/` zu konfigurieren. Dort kann festgelegt werden, was mit welchen Protokollinformationen geschehen soll. Die Informationen des LDAP-Servers werden über `LOCAL4` dem `syslogd` übergeben. Falls die Übergabe über `LOCAL0`, ..., `LOCAL7` geschehen soll, so kann dies über den Parameter `-l` definiert werden. Nähere Informationen zum `syslogd` sind in [4] nachzulesen.

Dem Dämon `slapd` muss ferner mitgeteilt werden, welche Konfigurationsdatei er verwenden soll. Fehlt diese Angabe über die Option `-f`, so verwendet er die Datei `/etc/openldap/slapd.conf`. Dieser Parameter ist hauptsächlich zum Testen wichtig, wenn es darum geht, neue Einstellungen nicht direkt in der Originalkonfiguration einzutragen.

Der LDAP-Server wartet auf Client-Anfragen am TCP-Port 389. Ein vom Standard abweichender Port kann über die Option `-p` definiert werden.

Beispiel 4.12: Starten von `slapd`

```
root@host1:~ # /usr/lib/openldap/slapd
root@host1:~ #
```

Beispiel 4.13: Starten von `slapd` mit der Option `-f`

```
root@host1:~ # /usr/lib/openldap/slapd \
# -f /etc/openldap/slapd.conf
root@host1:~ #
```

Beispiel 4.14: Starten von `slapd` mit der Option `-d ?`

```
root@host1:~ # /usr/lib/openldap/slapd -d ?
Debug levels:
        LDAP_DEBUG_TRACE        1
        LDAP_DEBUG_PACKETS      2
        LDAP_DEBUG_ARGS         4
        LDAP_DEBUG_CONNS        8
        LDAP_DEBUG_BER          16
        LDAP_DEBUG_FILTER       32
        LDAP_DEBUG_CONFIG       64
        LDAP_DEBUG_ACL          128
        LDAP_DEBUG_STATS        256
        LDAP_DEBUG_STATS2       512
        LDAP_DEBUG_SHELL        1024
        LDAP_DEBUG_PARSE        2048
        LDAP_DEBUG_ANY          65535
root@host1:~ #
```

Beispiel 4.15: Starten von `slapd` mit der Option `-d`

```
root@host1:~ # /usr/lib/openldap/slapd -d 63
slapd 1.2.11-Release (Sat Jul 29 13:23:02 GMT 2000)
        root@landman:/usr/src/packages/BUILD/openldap-…
slapd starting
listening for connections on 6, activity on:
before select active_threads 0
```

4.5.5 Das Datenbankformat LDBM

Die LDAP-Software der OpenLDAP-Organisation speichert alle Informationen des Verzeichnisdienstes in einer Datenbank ab, die hierarchisch organisiert ist. Die zugehörigen Daten werden dabei in Dateien des Formats LDBM (LDAP Database) abgelegt.

Bei LDBM handelt es sich um ein Binärformat. Somit sind die Daten nicht direkt einzusehen und können auch nicht direkt verändert werden. In diesem Abschnitt sollen jedoch drei Dateien vorgestellt werden. Sie befinden sich in dem Verzeichnis, das in der Datei `slapd.conf` mit dem Parameter `directory` angegeben wurde.

```
root@host1:/var/lib/ldap # ls -l dn.dbb id2entry.dbb \
# objectclass.dbb
-rw---   1   root   root   5120 Nov  9 13:25 dn.dbb
-rw---   1   root   root   5120 Nov 10 08:06 id2entry.dbb
-rw---   1   root   root   5120 Nov  9 13:25 objectclass.dbb
root@host1:/var/lib/ldap #
```

Datenbankdateien im LDBM-Format haben grundsätzlich die Endung `dbb`. Die obigen Einträge haben jeweils die folgenden Funktionen:

- In `dn.dbb` werden alle Distinguished Names des Verzeichnisdienstes festgehalten.
- Die verwendeten Objektklassen sind in der Datei `objectclass.dbb` zu finden.
- Außerdem sind in der Datei `id2entry` alle Einträge des Verzeichnisdienstes vorhanden.

Beachten Sie, dass ein Anwender, der direkten Zugriff auf die Datenbankdateien im entsprechenden Verzeichnis hat, alle Daten sowohl lesen als auch mitunter ändern kann. Dieses wird dadurch unterbunden, dass alle Dateien dem Systemadministrator `root` gehören und die Rechtestruktur 0600 verwendet wird. Dadurch kann nur `root` lesend und schreibend zugreifen. Diese Einschränkung der Rechte muss nicht ausdrücklich vergeben werden, da sie per Standard zur Anwendung kommt (siehe den Parameter `mode` auf Seite 61).

4.5.6 Das Datenbankformat LDIF

Neben dem Format LDBM existiert ein weiteres textbasiertes Format. Mit ihm ist es möglich, den gesamten Inhalt einer Datenbank mit einem Editor zu definieren. Dieser Mechanismus kommt bei folgendem Problem zur Anwendung: Der Systemadministrator möchte einen LDAP-Server auf seinem Rechner einrichten. Er hat die Software installiert und muss seine Firmenstruktur sowie alle Anwender in Form von Objekten repräsentieren. Gäbe es neben LDBM kein anderes Format, so müssten alle Objekte einzeln mit einem grafischen Programm hinzugefügt werden. Dieses wird jedoch sehr schnell unpraktisch und zu zeitaufwendig.

Mit dem LDAP Directory Interchange Format LDIF ist es möglich, alle Informationen, die zur Datenbank hinzugefügt werden sollen, in Form einer Textdatei zu definieren. Sie kann mit jedem beliebigen Editor erstellt und gepflegt werden. Auf diese Weise können die Daten mitunter auch aus anderen bereits vorhandenen Formen nach LDIF übertragen werden.

Bevor in diesem Buch eine LDIF-Datei, die zum Erstellen einer Datenbank verwendet wird, vorgestellt werden kann, müssen zunächst die grundlegenden syntaktischen Regeln beschrieben werden.

In einer LDIF-Datei können beliebig viele Objekte definiert werden. Jedes der Objekte ist nach dem folgenden Muster anzugeben:

```
[<Kennung>]
dn: <Distinguished Name>
<Attribut>: <Wert>
<Attribut>: <Wert>
<Attribut>: <Wert>
<…>     : <…>
```

Jedes Objekt wird in der Datenbank mit einer eindeutigen ID versehen. In der ersten Zeile kann diese Kennung angegeben werden, was jedoch allgemein unüblich ist. In der zweiten Zeile wird nun der Distinguished Name des Objekts angegeben. Ein Beispiel wäre `dn: o=Intern`. Anschließend folgen die Angaben zu den Attributen, wobei jedes Attribut in einer separaten Zeile aufzuführen ist (`objectclass: organization`). Falls die Werte eines Attributs aufgrund ihres Umfangs nicht in einer Zeile angegeben werden können, so ist es möglich, einen Zeilenumbruch einzufügen. Die folgenden Zeilen müssen dann mit einem Leerzeichen oder einem Tabulator beginnen:

```
<Attribut>: <Wert. . . . . .
 . . . . . .
 . . . . . .
 . . . .>
```

Um im LDIF-Format eine Organisation mit dem Namen `Intern` zu beschreiben, sind demnach lediglich folgende Zeilen notwendig:

```
dn: o=Intern
objectclass: organization
o: Intern
```

Werden in einer Datei wie im Regelfall mehrere Objekte definiert, so sind diese jeweils durch eine Leerzeile voneinander zu trennen.

Attributwerte bestehen meistens aus druckbaren Zeichenfolgen. Mitunter ist es jedoch notwendig, einen binären Wert angeben zu können. Dazu betrachten wir das Attribut `photo`, das die Syntax `bin` hat. Der Wert dieser Eigenschaft soll eine Grafik beschreiben. Es ist offensichtlich, dass die Angabe nicht in Textform vollzogen werden kann, da die Grafik an sich und nicht irgendein Pfad beschrieben werden muss. In einer LDIF-Datei können deshalb auch Werte angegeben werden, die nicht auf ASCII, sondern auf der BASE-64-Notation beruhen. Es gibt zwei Möglichkeiten, solche Werte zu kennzeichnen:

1. Zum einen wird ein solcher Wert dadurch markiert, dass hinter dem Namen des Attributs ein weiterer Doppelpunkt gesetzt wird:

   ```
   photo:: QmVpc3BpZWwK
   ```

2. Zum anderen wird der Wert ebenfalls als BASE-64 gedeutet, wenn er mit einem Leerzeichen beginnt:

   ```
   photo: " QmVpc3BpZWwK"
   ```

Die folgende Beispieldatei (siehe Beispiel 4.16) repräsentiert einen kompletten Verzeichnisbaum. Er wird im weiteren Verlauf dieses Kapitels in das Format LDBM konvertiert und bearbeitet. Der Directory Information Tree enthält eine Organisation und zwei organisatorische Einheiten, nämlich

- `Intern` als oberstes Element des Baumes,
- `Einkauf` als Eintrag unterhalb von `Intern` und
- ebenfalls als Eintrag unter `Intern` der `Verkauf`.

Ferner existieren im Einkauf und im Verkauf drei Anwender. Der Administrator wird durch das Objekt `Admin` dargestellt.

Beispiel 4.16: LDIF-Datei der Organisation `Intern`

```
root@host1:~ # cat intern.ldif
dn: o=Intern
objectclass: organization
o: Intern
```

```
dn: cn=Admin, o=Intern
objectclass: person
cn: Admin
sn: Admin
telephonenumber: 800
description: "LDAP Administrator"

dn: ou=Einkauf, o=Intern
objectclass: organizationalunit
ou: Einkauf
description: "Abteilung Einkauf"

dn: ou=Verkauf, o=Intern
objectclass: organizationalunit
ou: Verkauf
description: "Abteilung Verkauf"

dn: cn=MeierP, ou=Einkauf, o=Intern
objectclass: person
cn: MeierP
sn: Meier
userpassword: geheim
telephonenumber: 100
description: "Peter Meier"

dn: cn=MuellerC, ou=Einkauf, o=Intern
objectclass: person
cn: MuellerC
sn: Mueller
userpassword: geheim
telephonenumber: 101
description: "Christian Mueller"

dn: cn=SchulzeI, ou=Einkauf, o=Intern
objectclass: person
cn: SchulzeI
sn: Schulze
userpassword: geheim
telephonenumber: 102
description: "Ingrid Schulze"

dn: cn=JohansenL, ou=Verkauf, o=Intern
objectclass: person
cn: JohansenL
sn: Johansen
userpassword: geheim
telephonenumber: 201
description: "Lucas Johansen"
```

```
dn: cn=PaulsenI, ou=Verkauf, o=Intern
objectclass: person
cn: PaulsenI
sn: Paulsen
userpassword: geheim
telephonenumber: 202
description: "Ida Paulsen"

dn: cn=FlemmingJ, ou=Verkauf, o=Intern
objectclass: person
cn: FlemmingJ
sn: Flemming
userpassword: geheim
telephonenumber: 200
description: "Jan Flemming"

root@host1:~ #
```

Für jedes Objekt wird der Distinguished Name angegeben. Anschließend folgen die Attribute, zu denen auch `objectclass` gehört. Falls ein Objekt zu mehreren Objektklassen gehört, so ist dies in der Form

```
objectclass: <Objektklasse 1>
objectclass: <Objektklasse 2>
objectclass: <..>
```

anzugeben. Das gleiche Verfahren ist bei anderen Eigenschaften anzuwenden, die mehrere Werte aufweisen. Neben den notwendigen Attributen eines Objekts werden ferner optionale Eigenschaften angegeben. Hierbei handelt es sich generell um die Beschreibung (`description`). Bei den realen Anwendern werden außerdem eine Telefonnummer (`telephonenumber`) und ein Password (`userpassword`) definiert. Die Angabe des Passworts erfolgt im Klartext, da später die definierten Rechte im Verzeichnisbaum es jedem Anwender nur erlauben, sein eigenes Passwort einzusehen. Eine verschlüsselte Darstellung wie zum Beispiel in der Form

```
userpassword: {CRYPT}NGWZnzRgelHYE
```

ist jedoch ebenfalls möglich.

Insgesamt sind zwei Punkte bei der Erstellung einer Datei vom Format LDIF zu beachten:

1. Bei der Angabe von Attributnamen wird nicht zwischen Groß- und Kleinschreibung unterschieden (`userPassword` = `userpassword`).
2. Falls die Daten der LDIF-Datei zum Erstellen eines Verzeichnisbaums verwendet werden sollen, ist auf die Reihenfolge der Objekte in der Datei zu achten. Sie müssen in der Baumstruktur von der Wurzel her beschrieben werden. So ist zum

Beispiel ein Benutzerobjekt im Verzeichnisbaum erst zu erstellen, wenn zuvor der übergeordnete Container generiert worden ist.

4.5.7 Das Kommando `ldif2ldbm`

Die in der LDIF-Datei definierten Objekte sollen nun direkt im Verzeichnisbaum generiert werden. Dazu ist es nötig, sie in das Datenbankformat LDBM zu konvertieren. Das Kommando `ldif2ldbm` erledigt diese Funktion. Es ist allgemein in der Form

```
ldif2ldbm [Optionen]
```

zu starten. Über die Optionen wird definiert, welche LDIF-Datei konvertiert wird. Die der LDAP-Software zugrunde liegenden LDBM-Dateien im Verzeichnis `/var/lib/ldap/` werden anschließend erstellt. Falls sie bereits vorhanden waren, werden sie überschrieben. Nach dem Aufruf dieses Kommandos befinden sich ausschließlich die Objekte im Directory Information Tree, die mit der LDIF-Datei definiert wurden. Nach der Konvertierung kann der LDAP-Server aktiviert werden.

Option	Beschreibung
`-d <Level>`	Debug-Level
`-f <Datei>`	Konfigurationsdatei
`-i <Datei>`	LDIF-Datei
`-j <Anzahl>`	Anzahl paralleler Jobs
`-s <Verzeichnis>`	Pfad zu weiteren Tools

Tabelle 4.13: Optionen des Kommandos `ldif2ldbm`

Durch die Option `-d` wird das Kommando angeregt, während der Konvertierung Statusmeldungen auf der Konsole auszugeben. Der Wert des Debug-Levels hat die Bedeutung, die bereits vom LDAP-Server bekannt ist. Der Pfad zur Konfigurationsdatei des LDAP-Servers ist über den Parameter `-f` anzugeben. Aus ihr wird analysiert, in welchem Verzeichnis die LDBM-Dateien angelegt werden sollen. Standardmäßig wird als Konfigurationsdatei `slapd.conf` im Verzeichnis `/etc/openldap/` angenommen.

Der wichtigste und für die Funktionalität des Kommandos `ldif2ldbm` notwendige Parameter ist `-i <Datei>`. Über ihn wird die Eingabedatei im LDIF-Format angegeben, die die Basis der Konvertierung ist. Die Umwandlung der Objekte erfolgt anschließend durch die Benutzung weiterer Kommandos. Sie können parallel ausgeführt werden, was zwar die Geschwindigkeit der Konvertierung erhöht, jedoch auch den Speicher und die Festplatte stärker belastet. Der Grad der Parallelität wird mit dem Parameter `-j` definiert. Es werden so viele Jobs parallel gestartet, wie über `-j` angegeben wurde (Default: 1).

Die Option `-s` gibt an, an welcher Stelle im Dateisystem die Subkommandos zu finden sind, die von `ldif2ldbm` benutzt werden. Per Default wird das Verzeichnis `/usr/sbin/` angenommen. Tabelle 4.13 zeigt eine Übersicht der Optionen, die in den Beispielen 4.17 bis 4.19 gezeigt werden.

Beispiel 4.17: Starten von `ldif2ldbm` mit der Option `-i`

```
root@host1:~ # ldif2ldbm -i intern.ldif
root@host1:~ #
```

Beispiel 4.18: Starten von `ldif2ldbm` mit der Option `-d`

```
root@host1:~ # ldif2ldbm -i intern.ldif -d 1
…
=> index_change_values( "dn", 10, op=ADD )
=> ldbm_cache_open( "/var/lib/ldap/dn.dbb", 7, 600 )
<= ldbm_cache_open (cache 0)
index_change_values syntax 0x11 syntax bin 0x4
=> change_value( "=CN=FLEMMINGJ,OU=VERKAUF,O=INTERN", op=ADD )
<= change_value 0
…
=> index_change_values( "objectclass", 10, op=ADD )
=> ldbm_cache_open( "/var/lib/ldap/objectclass.dbb", 7, 600 )
<= ldbm_cache_open (cache 0)
index_change_values syntax 0x1 syntax bin 0x4
=> change_value( "=PERSON", op=ADD )
<= change_value 0
…
root@host1:~ #
```

Beispiel 4.19: Starten von `ldif2ldbm` mit der Option `-f`

```
root@host1:~ # ldif2ldbm -i intern.ldif \
# -f /etc/openldap/slapd.conf
root@host1:~ # /etc/rc.d/init.d/ldap start
Starting ldap-server.                                      done
root@host1:~ #
```

4.5.8 Das Kommando `ldbmcat`

Das Kommando `ldbmcat` hat genau die entgegengesetzte Funktionalität zu `ldif2ldbm`. Es konvertiert Dateien im LDBM-Format in das LDIF-Format. Auf diese Weise ist es möglich, alle Einträge der LDAP-Datenbank in eine Textdatei zu überführen. Somit lässt sich ein Replica-Server komplett mit den Daten des Masters updaten, indem die Ausgaben dieses Kommandos auf dem Replica-Server als Eingaben des Kommandos `ldif2ldbm` benutzt werden.

```
ldbmcat [Optionen] <LDBM-Datei>
```

Dem Kommando steht lediglich eine Option zur Verfügung, die Sie in Tabelle 4.14 sehen.

Option	Beschreibung
`-n`	Keine Angabe der IDs

Tabelle 4.14: Option des Kommandos `ldbmcat`

Standardmäßig werden in der Ausgabe des Kommandos die eindeutigen Kennungen der Objekte (ID) angegeben. Mit der Option `-n` werden sie nicht aufgeführt. Bei der grundsätzlich anzugebenden LDBM-Datei handelt es sich um `id2entry.dbb` aus dem Datenbankverzeichnis.

Beispiel 4.20: Starten von `ldbmcat`

```
root@host1:~ # ldbmcat /var/lib/ldap/id2entry.dbb
1
dn: o=Intern
objectclass: organization
o: Intern

2
dn: cn=Admin, o=Intern
objectclass: person
cn: Admin
sn: Admin
telephonenumber: 800
description: "LDAP Administrator"

…
root@host1:~ #
```

Die drei Punkte … deuten an, dass die Ausgabe hier gekürzt dargestellt ist.

Beispiel 4.21: Starten von `ldbmcat` mit der Option `-n`

```
root@host1:~ # ldbmcat -n /var/lib/ldap/id2entry.dbb \
# > fullupdate.ldif
root@host1:~ # scp fullupdate.ldif host2:/root/fullupdate.ldif
fullupdate.ldif | 1 KB | 1.3 kB/s | ETA: 00:00:00 | 100%

root@host1:~ # ssh host2
Last login: Fri Nov 10 09:58:19 2000 from host1.intern
Have a lot of fun…
You have mail.
root@host2:~ # ldif2ldbm -i fullupdate.ldif
```

```
root@host2:~ # /etc/rc.d/init.d/ldap start
Starting ldap-server.                                          done
root@host2:~ #
```

Durch die Verwendung der Option `-n` entspricht die Ausgabe exakt der Datei aus Abschnitt 4.5.6. Das Kopieren der Daten vom Master (`host1`) zur Replica (`host2`) erfolgt im Beispiel mit dem Befehl `scp` der Secure Shell. Das Einloggen auf der Replica wird ebenfalls mit der sicheren Übertragung der SSH realisiert. Die Datenübertragung und das Einloggen an der entfernten Station sind zwar auch mit den Programmen `ftp` und `telnet` möglich, diese sollten jedoch wegen der unverschlüsselten Kommunikation der Partner nicht verwendet werden. Alle Daten (auch die Passwörter) können von anderen Netzwerkteilnehmern ausspioniert werden.

4.5.9 Die Datei `slapd.replog`

Um eine inkrementelle Replizierung zu realisieren, sind in der Konfigurationsdatei des Master-Servers entsprechende `replica`-Einträge erforderlich. Sie beschreiben das Ziel, auf das die Daten des Masters dupliziert werden sollen. Es ist auch möglich, die Daten auf mehrere Server zu replizieren. Jeder Replica-Server ist dann in der Datei `/etc/openldap/slapd.conf` anzugeben.

Der LDAP-Master-Server protokolliert alle Aktionen am Verzeichnisdienst in der Datei, die über den Parameter `replogfile` in der Konfigurationsdatei definiert wurde. In dem Beispiel, das in diesem Buch verwendet wird, handelt es sich um die Datei `/var/run/slapd.replog`.

In diesem Abschnitt wird das Format der Datei `slapd.replog` beschrieben, bevor die inkrementelle Replizierung im nächsten Abschnitt erläutert wird.

Für jede Änderung an der Datenbank des Verzeichnisdienstes wird ein Eintrag in der Log-Datei erzeugt. Mehrere Einträge werden mit Leerzeilen voneinander getrennt.

```
replica: <Host:Port>
time: <Zeit>
dn: <Distinguished Name>
```

Die erste Zeile eines jeden Eintrags beschreibt den Host des Replica-Servers. Sie kann bei mehreren Replica-Datenbanken auch mehrfach auftreten. Der Server kann in Form seines DNS-Namens oder seiner IP-Adresse angegeben werden. Optional wird der TCP-Port aufgeführt, sofern dies in der Datei `slapd.conf` ebenfalls geschehen ist. Über die Zeitangabe wird jede Änderung eindeutig beschrieben. Sie wird, wie unter Linux üblich, in Sekunden seit dem 01.01.1970 00:00 Uhr angegeben. Falls mehrere Änderungen zum gleichen Zeitpunkt vollzogen werden, kann der Zeitangabe zur Unterscheidung noch eine optionale dezimale Erweiterung folgen. Schließlich bezeichnet der Distinguished Name das Objekt, an dem die Änderung durchgeführt wurde.

Nach diesen grundlegenden Angaben wird über die Zeile

```
changetype: <add|delete|modify|modrdn>
```

die Art der an dem Objekt durchgeführten Aktion beschrieben. Die möglichen Arten sind in Tabelle 4.15 aufgelistet.

Art	Beschreibung
`add`	Hinzufügen eines Objekts
`delete`	Löschen eines Objekts
`modify`	Ändern der Attribute eines Objekts
`modrdn`	Ändern des Objektnamens

Tabelle 4.15: Änderungsarten der Replizierung

Beim Löschen von Einträgen sind keine weiteren Details vorhanden, die aufgeführt werden können. Beim Hinzufügen werden jedoch alle dem Objekt zugehörigen Attribute aufgelistet. Dieses geschieht in der Form:

```
<Attributname>: <Wert>
<Attributname>: <Wert>
<Attributname>: <Wert>
...
```

Die Aktion `modify` beschreibt eine Änderung an den Attributen eines Objekts. Hierbei kann es sich um

- das Hinzufügen (`add`),
- das Löschen (`delete`) oder
- das Ersetzen (`replace`)

handeln.

So wird beim Hinzufügen eines Attributs ein Datensatz der Form

```
replica: <Host:Port>
time: <Zeit>
dn: <Distinguished Name>
changetype: modify
add: <Attributname>
<Attributname>: <Attributwert>
-
```

erstellt. Der Eintrag wird mit einem Trennzeichen in einer separaten Zeile abgeschlossen. Das Ersetzen eines Attributwertes erfolgt analog. Für das Löschen von Eigenschaften sei auf Beispiel 4.22 verwiesen.

Bei der Änderung `modrdn` wird der Name des Objekts im Verzeichnisdienst aktualisiert. In der Datei `slapd.replog` wird neben dem neuen Namen vermerkt, ob der alte gelöscht oder als Attribut gespeichert wurde.

```
replica: <Host:Port>
time: <Zeit>
dn: <Distinguished Name>
changetype: modrdn
newrdn: <Neuer Name>
deleteoldrdn: <0|1>
```

Das folgende Beispiel zeigt eine Log-Datei `slapd.replog`, in der nahezu alle der zuvor erwähnten Einträge vorhanden sind.

Beispiel 4.22: Log-Datei `slapd.replog` des Master-Servers

```
root@host1:/var/run # cat slapd.replog
replica: 192.168.17.2:389
time: 974096572
dn: CN=MEIERP,OU=EINKAUF,O=INTERN
changetype: delete

replica: 192.168.17.2:389
time: 974105017
dn: CN=FLEMMINGJ,OU=VERKAUF,O=INTERN
changetype: modify
delete: description
-

replica: 192.168.17.2:389
time: 974105040
dn: CN=FLEMMINGJ,OU=VERKAUF,O=INTERN
changetype: modify
add: description
description: Jan Flemming
-

replica: 192.168.17.2:389
time: 974105283
dn: cn=MeierP, ou=Einkauf, o=Intern
changetype: add
objectclass: person
cn: MeierP
sn: Meier
userpassword: geheim
telephonenumber: 100
description: "Peter Meier"

replica: 192.168.17.2:389
```

```
time: 974109468
dn: CN=FLEMMINGJ,OU=VERKAUF,O=INTERN
changetype: modrdn
newrdn: cn=FlemmingA
deleteoldrdn: 1

root@host1:/var/run #
```

Diese Datei protokolliert die folgenden Aktionen:

1. Das Objekt `MeierP` des Einkaufs wurde gelöscht.
2. Das Attribut `description` des Objekts `FlemmingJ` aus dem Verkauf wurde entfernt.
3. Das Attribut `description` des Objekts `FlemmingJ` aus dem Verkauf wurde mit dem Wert `Jan Flemming` hinzugefügt.
4. Das Objekt `MeierP` wurde im `Einkauf` mit mehreren Eigenschaften erstellt.
5. Der Relative Distinguished Name des Objekts `FlemmingJ` wurde in den Namen `FlemmingA` geändert. Der vorige Name wurde dabei nicht in Form eines Attributs gespeichert.

4.5.10 Der Dämon `slurpd`

Die Daten, die der LDAP-Master-Server in der Datei `slapd.replog` protokolliert, dienen einzig und allein dazu, an den oder die aufgeführten Replica-Server übertragen zu werden. Die OpenLDAP-Organisation stellt dafür ein Programm namens Standalone LDAP Update Replication Daemon (`slurpd`) zur Verfügung. Der Start dieses Dämons kann entweder ebenfalls mit dem Startskript `ldap` im Verzeichnis `/etc/rc.d/init.d/` vollzogen werden oder aber direkt in der Linux-Shell. Für das Startskript ist bei einer SuSE-Distribution die Variable `START_SLURPD` mit dem Programm `yast` auf `yes` zu setzen. Dadurch erfolgt die Aktivierung bei jedem Systemstart.

Beispiel 4.23: Auszug aus dem Startskript

```
root@host1:~ # cat /etc/rc.d/init.d/ldap
…
case "$1" in
 start)
  echo -n "Starting ldap-server."
  /sbin/startproc /usr/lib/openldap/slapd || return=$rc_failed
  echo -e "$return"

  if [ "$START_SLURPD" == "yes" ]; then
```

```
    echo -n "Starting slurpd."
   /sbin/startproc /usr/lib/openldap/slurpd || return=$rc_failed
   echo -e "$return"
  fi
  ;;
…
root@host1:~ #
```

Der Aufruf von `slurpd` erfolgt aus dem Startskript heraus ohne die Verwendung von Optionen. Es sind jedoch generell Einstellungen vorhanden, die den Replizierungsvorgang beeinflussen. Diese können benutzt werden, indem der Update Daemon nicht über das Startskript, sondern direkt auf der Konsole aus dem Verzeichnis `/usr/lib/openldap/` gestartet wird. Das Startskript kann aber auch dahingehend geändert werden, dass die Optionen dort zur Anwendung kommen.

```
slurpd [Optionen]
```

Option	Beschreibung
`-d <Level>`	Debug-Level
`-f <Datei>`	Konfigurationsdatei
`-r <Datei>`	Replog-Datei
`-o`	Einmalige Ausführung
`-t <Verzeichnis>`	Temporäres Verzeichnis

Tabelle 4.16: Optionen des LDAP Replication Daemons `slurpd`

Die Optionen `-d` und `-f` sind bereits bekannt. Sie beschreiben den Debug-Level bzw. den Pfad zur Konfigurationsdatei des LDAP-Servers (Standard: `/etc/openldap/slapd.conf`). Aus ihr liest der Dämonprozess die Informationen, an welcher Stelle im Dateisystem er die Replog-Datei findet. Eine alternative Datei kann über den Parameter `-r` angegeben werden. `slurpd` läuft grundsätzlich, sofern das Debugging nicht eingeschaltet ist, im Hintergrund ab und überträgt die inkrementellen Daten aus der Datei `slapd.replog` fortlaufend zum Replica-Server. Durch den Parameter `-o` wird `slurpd` jedoch angewiesen, im so genannten One-Shot-Modus nur einmalig die Replog-Datei zur Replica zu übertragen und sich dann zu beenden.

Der Dämonprozess arbeitet in folgenden Schritten:

1. In periodischen Abständen wird die Datei `slapd.replog` gelesen.
2. Sie wird dann kurzzeitig für weitere Änderungen gesperrt.
3. In dieser Zeit wird die Datei in ein temporäres Verzeichnis übertragen und an der Originalstelle entleert.
4. Die Sperrung wird anschließend wieder aufgehoben.

5. Abschließend werden die Daten der kopierten Datei an die angegebenen Replica-Server übertragen und dort sofort in deren Datenbanken eingepflegt.

Als temporäres Verzeichnis wird `/usr/tmp/` verwendet. Mit der Option `-t` kann jedoch eine andere Stelle im Dateisystem dafür definiert werden.

Man beendet den Dämon `slurpd` genau wie `slapd` entweder durch das Skript `/etc/rc.d/init.d/ldap stop` oder direkt durch Aufruf des Shell-Kommandos `killall slurpd` bzw. `killall slapd`.

Beispiel 4.24: Starten von `slurpd`

```
root@host1:~ # /usr/lib/openldap/slurpd
root@host1:~ #
```

Beispiel 4.25: Starten von `slurpd` mit der Option `-f`

```
root@host1:~ # /usr/lib/openldap/slurpd \
# -f /etc/openldap/slapd.conf
root@host1:~ #
```

Beispiel 4.26: Starten von `slurpd` mit der Option `-d`

```
root@host1:~ # /usr/lib/openldap/slurpd -d ?
Debug levels:
        LDAP_DEBUG_TRACE        1
        LDAP_DEBUG_PACKETS      2
        LDAP_DEBUG_ARGS         4
        LDAP_DEBUG_CONNS        8
        LDAP_DEBUG_BER          16
        LDAP_DEBUG_FILTER       32
        LDAP_DEBUG_CONFIG       64
        LDAP_DEBUG_ACL          128
        LDAP_DEBUG_ANY          65535

root@host1:~ # /usr/lib/openldap/slurpd -d 1
ldap_open
ldap_init
ldap_delayed_open
open_ldap_connection
ldap_connect_to_host: 192.168.17.2:389
sd 6 connected to: 192.168.17.2
ldap_open successful, ld_host is (null)
…
```

Beispiel 4.27: Beenden von `slurpd`

```
root@host1:~ # killall slurpd
root@host1:~ #
```

Die Übertragung der Daten vom Master zur Replica erfolgt unverschlüsselt und daher absolut unsicher. Sie kann jedoch mit der Secure Shell (SSH) auch gesichert stattfinden (siehe auch Abschnitt 5.3.9).

4.6 Der LDAP-Client

Der LDAP-Server ist mittlerweile installiert und funktionstüchtig. Auch die Replizierung ist konfiguriert und aktiv. In diesem Abschnitt geht es um den Zugriff auf die Einträge in der LDAP-Datenbank. Dazu werden einige Kommandos und Programme vorgestellt, die diesen Tatbestand erfüllen.

4.6.1 Die Datei `ldap.conf`

Der Zugriff mit einem LDAP-Client auf den LDAP-Server kann durch die Verwendung von diversen Kommandos in der Linux-Shell erfolgen. Dadurch ist es möglich, die gewünschten Aktionen in Form von Shell-Skripten ausführen zu lassen. Alle LDAP-Kommandos, die einen Client-Zugriff darstellen, basieren auf der Konfigurationsdatei `ldap.conf` im Verzeichnis `/etc/openldap/`. Es handelt sich um eine reine Textdatei, die mit jedem beliebigen Editor bearbeitet werden kann. Neben den möglichen Parametern können auch Kommentare eingetragen werden. Sie beginnen mit dem Zeichen # und fließen genauso wie alle Leerzeilen nicht in die Konfiguration ein.

Beispiel 4.28: Kommentare

```
root@host3:~ # cat /etc/openldap/ldap.conf
#
# Konfigurationsdatei des LDAP-Clients
#
# ldap.conf
#
# Erstellt von Jens Banning
#
root@host3:~ #
```

Die globale Konfiguration des LDAP-Clients erfolgt anhand von sechs Parametern.

Der Parameter `BASE`

Mit dem Parameter `BASE` wird der Distinguished Name des Objekts angegeben, der als Basis für alle Client-Aktionen verwendet werden soll:

```
BASE <Distinguished Name>
```

Es empfiehlt sich, den ersten Container unterhalb der Wurzel zu verwenden, z. B.:

```
BASE o=Intern
```

Der Parameter `HOST`

Die Position des LDAP-Servers im Netzwerk wird anhand seiner IP-Adresse oder seines DNS-Namens festgelegt. Der TCP-Port, der angesprochen werden soll, kann direkt mit angegeben werden:

```
HOST <Hostname:Port> <Hostname:Port> <…>
```

Falls aus Gründen der Ausfallsicherheit mehrere LDAP-Server angegeben werden sollen, so kann dies jeweils durch ein Leerzeichen getrennt hinter `HOST` geschehen:

```
HOST host1.intern
```

Der Parameter `PORT`

Falls über die `HOST`-Option der TCP-Port des Servers nicht definiert wurde, so kann dem Parameter `PORT` der nummerische Wert des per Standard zu verwendenden TCP-Kanals mitgeteilt werden:

```
PORT <TCP-Portnummer>
```

LDAP-Server reagieren in der Regel am Port 389:

```
PORT 389
```

Die Parameter `SIZELIMIT` **und** `TIMELIMIT`

Über diese beiden Parameter wird am Client eine Grenze bezüglich des Umfanges und der Zeit gesetzt, die bei der Ausführung von Anfragen zur Anwendung kommt.

```
SIZELIMIT <Wert>
TIMELIMIT <Wert>
```

Diese Parameter bestimmen, wie viel Zeit verwendet wird und wie viele Ergebnisse betrachtet werden, z. B.:

```
SIZELIMIT 12
TIMELIMIT 15
```

Der Parameter `DEREF`

Im Verzeichnisdienst ist es möglich, so genannte Alias-Objekte zu definieren. Es handelt sich dabei um Einträge, die lediglich auf Objekte zeigen, die sich an anderer Stelle im DIT befinden. Dem LDAP-Client muss mitgeteilt werden, ob er diese Alias-Objekte auflösen soll. Dazu wird der Parameter `DEREF` verwendet:

```
DEREF <always|finding|never|searching>
```

Dadurch wird definiert, dass Aliase immer, nur beim Ansprechen, nie oder nur beim Suchen aufgelöst werden. Der Standardwert lautet `never`:

```
DEREF never
```

Im Ganzen könnte die Konfigurationsdatei des LDAP-Clients demnach so aussehen, wie in Beispiel 4.29 gezeigt.

Beispiel 4.29: Konfigurationsdatei des Clients

```
root@host3:~ # cat /etc/openldap/ldap.conf
#
# Konfigurationsdatei des LDAP-Clients
#
# ldap.conf
#
# Erstellt von Jens Banning
#

BASE    o=Intern
HOST    host1.intern
PORT    389

SIZELIMIT       12
TIMELIMIT       15
DEREF           never

root@host3:~ #
```

Diese Konfigurationsdatei ist systemweit für alle Anwender auf dem Linux-Client maßgebend. Es ist jedoch auch möglich, die Einstellungen benutzerspezifisch festzulegen. Dazu muss man im Heimatverzeichnis eine Datei namens `.ldaprc` anlegen. In ihr sind exakt die gleichen Parameter möglich wie in der systemweit relevanten Datei. Es ist daher sinnvoll, `/etc/openldap/ldap.conf` in das Home-Directory des gewünschten Benutzers auf den Namen `.ldaprc` zu kopieren. Anschließend können die Einstellungen dort angepasst werden.

4.6.2 Die Datei `ldapfilter.conf`

Am LDAP-Client existiert eine Reihe von Dateien, die ebenfalls Einfluss auf die Funktionalität von LDAP-Aktionen haben. Die Datei `ldapfilter.conf` steuert das Verhalten von Suchfunktionen. Sie wird in einigen Anwendungen verwendet, um den Suchbegriff, den der Anwender eingegeben hat, so zu deuten, dass die Suche an der LDAP-Datenbank ausgeführt werden kann. Die Datei hat demnach die Aufgaben,

1. den Suchbegriff zu analysieren und
2. ihn zu konvertieren.

Bei der genannten Datei handelt es sich wie bei allen anderen LDAP-Konfigurationsmöglichkeiten um eine Textdatei. In ihr ist es möglich, neben den eigentlichen Einstellungen aus Gründen der Übersichtlichkeit auch Leerzeilen und Kommentare einzufügen. Zur Kommentierung werden Zeilen verwendet, die mit dem Zeichen # beginnen.

Die Suchfilter in dieser Datei bilden hauptsächlich die Grundlage für die LDAP-Suche mit den in Abschnitt 4.7 auf Seite 141 beschriebenen Gateways. Da die Einstellungen in dieser Datei jedoch auch von anderen Anwendungen benutzt werden können, ist es notwendig, sie an dieser Stelle vorzustellen.

Die Einträge in der Datei `/etc/openldap/ldapfilter.conf` sind in Bereiche eingeteilt. Jeder Bereich beschreibt im Prinzip eine Anwendung, für die die aufgeführten Filter gelten. Die Bereiche können mit einer Leerzeile voneinander getrennt werden. Im Folgenden werden Einstellungen für die Anwendung `finger` (siehe Abschnitt 4.7) beschrieben. Falls die Datei für eigene Anwendungen genutzt werden soll, so kann deren Name zur Kennzeichnung des Bereichs verwendet werden. Falls die Daten, die in dieser Datei abgelegt werden, Leerzeichen enthalten, ist es möglich, den entsprechenden Ausdruck mit Anführungszeichen (") zu umschließen, um ihn somit als Ganzes darzustellen.

Die erste Zeile hat allgemein die Form:

```
<Bereich>
```

Für den Finger-Dienst würde sie so aussehen:

```
finger
```

In den nächsten Zeilen werden nun die eigentlichen Filterdefinitionen festgehalten. Sie bestehen aus vier oder fünf Spalten pro Zeile, deren Inhalt in Tabelle 4.17 angegeben ist.

Zunächst wird geprüft, ob die vom Anwender in Auftrag gegebene Suche dem in der ersten Spalte definierten Muster entspricht. Anschließend werden die einzelnen Wörter des Suchbegriffs bestimmt. Dazu wird er anhand der in der zweiten Spalte festgelegten Trennzeichen aufgeteilt. Mit den so erlangten Wörtern wird anschließend die Suchvorgabe definiert, die direkt per LDAP aufzulösen ist. Die Ergebnisse werden dem Anwender unter Zuhilfenahme der vierten Spalte (des Ergebnistextes) präsentiert. In der letzten Spalte kann angegeben werden, auf welchen Bereich im Verzeichnisbaum sich die Suche bezieht. Es sind die Werte `base`, `onelevel` und `subtree` möglich, um eine Suche auf der Basis, dem angesprochenen Container oder dem ganzen Subtree zu beschreiben. Die Angabe des Suchbereichs kann in der Datei `ldapfilter.conf` entfallen. Es wird dann `subtree` angenommen.

Spalte	Beschreibung
1	Suchmuster
2	Trennzeichen
3	Suchvorgabe
4	Ergebnistext
5	Suchbereich

Tabelle 4.17: Spalten der Datei `ldapfilter.conf`

Der Zusammenhang zwischen den Werten einer Filterdefinition lässt sich wie folgt beschreiben:

Der Suchtext des Anwenders wird anhand des Suchmusters und der Trennzeichen in Wörter aufgeteilt, um somit in dem Suchbereich die erstellte Suchvorgabe auszuführen. Das Ergebnis wird dem Anwender mit dem Ergebnistext präsentiert.

Die folgende Zeile ist per Default in der Filterdatei vorhanden:

```
"="        " "        "%v"        "arbitrary filter"
```

Sie bedeutet, dass eine Anfrage, die das Zeichen = enthält, in ihre mit einem Leerzeichen getrennten Bestandteile zerlegt wird. Anschließend wird die in der dritten Spalte definierte Suchvorlage ausgeführt. Dabei hat die Variable `%v` die folgende Bedeutung:

- `%v` beschreibt den kompletten vom Anwender eingegebenen Suchtext.
- `%v$` beschreibt das erste Wort des Suchtextes.
- `%v<x>` beschreibt das x-te Wort des Suchtextes.
- Um mehrere Wörter anzusprechen, kann das Konstrukt `%v<x>-<y>` verwendet werden. Es beschreibt die Wörter `x` bis `y`.
- Fehlt in der vorigen Angabe die Definition von `y`, so sind die Wörter von `x` bis zum Ende des Suchtextes gemeint (`%v<x>-`).

Das Suchergebnis wird dem Anwender mit den Worten

One `arbitrary filter` match was found for …

bzw.

Two `arbitrary filter` matches were found for …

usw. repräsentiert.

Die eben vorgestellte Filterdefinition kommt zum Beispiel zur Anwendung bei einer Anfrage in der Form:

Suche `cn=Admin`.

Genau diese Anfrage würde auch per LDAP ausgeführt (dritte Spalte: `%v`) und das Ergebnis in folgender Form dargestellt:

One `arbitrary filter` match was found for `cn=Admin`.

Ebenfalls ist in der Filterdatei ein Eintrag der Form

```
"@"        " "        "(mail=%v)"        "email address"
```

vorhanden. Er bedeutet, dass bei einer Anfrage, die das Zeichen `@` enthält, an der Datenbank eine Anfrage auf das Attribut `mail` ausgeführt wird. Der Ergebnistext lautet `email address`.

Beispiel 4.30 zeigt lediglich einen Auszug der per Default definierten Filter.

Beispiel 4.30: Filterdatei des Clients

```
root@host3:~ # cat /etc/openldap/ldapfilter.conf
…
finger
  "^$"           ""      "(objectclass=*)"  "default filter"
  "="            " "     "%v"               "arbitrary filter"
  "@"            " "     "(mail=%v)"        "email address"
                         "(mail=%v*)"       "start of email add."
  "^.[\. _].*"   ". _"   "(cn=%v1* %v2-)"   "first initial"
  ".*[. _].$"    ". _"   "(cn=%v1-*)"       "last initial"
…
root@host3:~ #
```

4.6.3 Die Datei `ldaptemplates.conf`

Die in der Datei `ldapfilter.conf` definierten Ergebnistexte beziehen sich lediglich auf die Anzahl der gefundenen Daten. In welcher Form die eigentlichen Ergebnisdaten präsentiert werden, wird mit den Definitionen in der Datei `ldaptemplates.conf` festgelegt. Die Syntax der möglichen Vorgaben wird in diesem Abschnitt beschrieben. Auch diese Datei ist textbasiert, und Kommentare können ebenfalls mit einem # eingeleitet werden.

Die Datei beginnt mit einer Zeile, die die Versionsnummer beschreibt:

```
Version 1
```

Anschließend folgt für jedes gewünschte Objekt eine Umgebung, in der die Vorgaben definiert werden. Sie beginnt mit zwei umgangssprachlichen Namen für das Objekt, wobei der erste Name die Einzahl und der zweite die Mehrzahl beschreibt, z. B.:

```
"Person"
"People"
```

Außerdem kann ein Symbol angegeben werden. Anhand dieses Eintrages können zum Beispiel grafische Anwendungen entscheiden, in welcher Form sie das Objekt darstellen.

```
"person icon"
```

Als Nächstes kommen die Optionen des Templates. Es sind die Werte `addable` und `modrdn` möglich. Die Option `altview` dient zum Verzweigen von Templates und wird in der Regel nicht verwendet.

```
addable
```

Falls keine Optionen angegeben werden sollen, so kann dies durch die Zeichen `""` dargestellt werden.

Weiterhin werden in der Datei die Objektklassen aufgeführt, für die das Template gelten soll. Es können mehrere Klassen in separaten Zeilen aufgelistet werden. Die Auflistung ist mit einem `END` abzuschließen:

```
person
END
```

Falls Änderungen an einem Objekt vollzogen werden, so muss sich der Anwender mit dem folgenden Attribut authentifizieren. Die Zeichen `""` bedeuten, dass es angemessen ist, sich mit dem Eintrag an sich auszuweisen.

```
""
```

Der Name eines neu zu erstellenden Objekts wird anhand des Attributs

```
cn
```

generiert. Neue Einträge werden standardmäßig unter

```
"o=Intern"
```

angelegt. Für die Attribute neuer Objekte können im Weiteren Defaultwerte festgelegt werden. Die Auflistung der Eigenschaften ist mit einem `END` abzuschließen:

```
constant      description      Benutzer
addersdn      seealso
END
```

Mit dem Wort `constant` wird ein konstanter Wert festgelegt. Mit `addersdn` kann einer Eigenschaft als Wert der Objektersteller übergeben werden.

Am Ende einer Template-Definition werden die Texte aufgeführt, die dem Anwender mitgeteilt werden sollen. Sie sind in der Form

```
item <Typ> <Ergebnistext> <Attributname>
```

anzugeben. Welche Typen im Einzelnen möglich sind, ist zu Beginn der Datei als Kommentar beschrieben. So beschreibt zum Beispiel `cis` einen Case Ignore String. Der Typ kann ferner mit einer Option ergänzt werden, zum Beispiel `sort`. Die mit `item` getätigten Zuweisungen sind mit einem `END` abzuschließen, womit auch das Template beendet ist.

```
item cis         "Work Phone"            telephoneNumber
item cis         "Fax Number"            facsimileTelephoneNumber
item cis         "Pager Number"          pager
item cis,sort    "Title"                 title
item cis         "Home Phone"            homePhone
item cis         "Favorite Beverage"     drink
item cis         "Notice"                notice
END
```

In der Datei `ldaptemplate.conf` im Verzeichnis `/etc/openldap/` können beliebig viele Templates aufgeführt werden.

4.6.4 Die Datei `ldapsearchprefs.conf`

Bei der letzten Konfigurationsdatei handelt es sich um eine Definition von Suchvorgaben. Sie können von jeder Anwendung, sofern dies programmiert wurde, verwendet werden. In den Vorgaben wird genau definiert,

- bei welchem Objekt
- nach welchem Kriterium

gesucht werden kann. Die Einstellung der Vorgaben erfolgt in der Textdatei `ldapsearchprefs.conf`, die sich ebenfalls im Verzeichnis `/etc/openldap/` befindet. Kommentare können dabei in der bereits bekannten Weise verwendet werden.

Auch diese Datei ist nach der Installation vorhanden und braucht in der Regel nicht geändert werden. Sie beginnt mit einer Zeile, die die Versionsnummer beschreibt:

```
Version 1
```

Anschließend folgt ein umgangssprachlicher Name für die Objekte, zu denen Suchvorgaben definiert werden, z. B.:

```
People
```

Die nächste Zeile kann eine Option enthalten. Zurzeit ist jedoch nur `internal` möglich. Sie bedeutet, dass die Objekte dem Anwender nicht direkt präsentiert werden. Alternativ kann die Zeichenkette `""` verwendet werden:

```
""
```

Welchen Text ein Anwendungsprogramm benutzen kann, mit dem es den Anwender zur Eingabe eines Suchbegriffs auffordert, und welchen Zusatz die Suche generell mit sich bringt, ist in den Zeilen

```
"Search For:"
"(&(objectClass=person))"
```

vermerkt. Ferner erfolgt ein Verweis zu einem Filterbereich der Datei `ldapfilter.conf`. Hier wird der Suchbegriff näher analysiert.

```
"xax500"
```

Die folgenden drei Einträge definieren, was bei einer Suche mit mehreren Ergebnissen geschehen soll. Dem Anwender wird dann ein bestimmtes Attribut der gefundenen Objekte mit einer Beschreibung mitgeteilt. Ferner wird festgehalten, welchen Umfang Client-Anfragen generell haben, z. B.:

```
title
"Title"
subtree
```

Jetzt werden die Zeilen aufgeführt, die genau beschreiben, nach welchen Attributen mit welchem Kriterium gesucht werden kann. Dazu werden auch Textpassagen aufgeführt, die das Anwendungsprogramm nutzen kann:

```
"Common Name"                   cn                  11111  ""  ""
"Surname"                       sn                  11111  ""  ""
"Business Phone"                "telephoneNumber"   11101  ""  ""
"E-Mail Address"                "mail"              11111  ""  ""
"Uniqname"                      "uid"               11111  ""  ""
"Title"                         title               11111  "" ""
END
# Match types
"exactly matches"               "(%a=%v))"
"approximately matches"         "(%a~=%v))"
"starts with"                   "(%a=%v*))"
```

```
"ends with"                    "(%a=*%v))"
"contains"                     "(%a=*%v*))"
END
```

Wählt der Anwender zum Beispiel `Common Name` aus, so erfolgt die Suche nach dem Attribut `cn`. Die Bitmap `11111` bedeutet, dass alle fünf Kriterien, die unter `# Match types` aufgeführt sind, erlaubt sind. So ist es bei der Telefonnummer nicht möglich, eine `"ends with"`-Anfrage zu stellen.

Es können mehrere Suchvorgaben in der Datei `ldapsearchprefs.conf` angegeben werden. Auch diese Datei wird hauptsächlich von den LDAP-Gateways verwendet.

4.6.5 Die Datei `ldapfriendly`

Die letzte Datei in diesem Zusammenhang ist `ldapfriendly`. Sie befindet sich im Verzeichnis `/etc/openldap/` oder `/etc/web500/`. Der Ort, an dem diese Datei abgelegt ist, ist distributionsabhängig.

Auch wenn die Einstellungen in der Datei nur von relativ wenigen Funktionen verwendet werden, so soll sie doch an dieser Stelle vorgestellt werden, da sie sich als sehr nützliche Informationsquelle erweist.

In `ldapfriendly` sind jeweils Zeilen aus zwei Spalten vorhanden. Die erste Spalte beinhaltet dabei einen zweistelligen Ländercode und die zweite das zugehörige Land (siehe Tabelle 4.18).

Spalte	Beschreibung
1	Ländercode
2	Land

Tabelle 4.18: Spalten der Datei `ldapfriendly`

Diese Datei kann zum Beispiel beim Erstellen eines Country-Objekts im Verzeichnisdienst verwendet werden. Auch ist es möglich, über diese Datei als Beschreibung (`description`) eines Ländercontainers den Namen des Landes einzutragen.

Beispiel 4.31: Datei `ldapfriendly`

```
root@host3:~ # cat /etc/web500/ldapfriendly
AR      Argentina
AU      Australia
AT      Austria
BE      Belgium
BR      Brazil
```

```
CA      Canada
CZ      Czech Republic
HR      Croatia
DK      Denmark
EE      Estonia
FI      Finland
FR      France
DE      Germany
GB      Great Britain
GR      Greece
HK      Hong Kong
HU      Hungary
IS      Iceland
IN      India
IE      Ireland
IL      Israel
IT      Italy
JP      Japan
…
root@host3:~ #
```

4.6.6 Das Kommando `ldapsearch`

Zum Zugriff auf die Daten des Directory Information Tree existieren diverse Kommandos. Mit ihnen ist es möglich, die gewünschten Aktionen direkt in der Linux-Shell auszuführen. Der entscheidene Vorteil dieses Mechanismus besteht darin, dass keine grafische Oberfläche benötigt wird. Ferner können mehrere Kommandoaufrufe in Shell-Skripten vereint werden.

Die grundlegendste Funktion, die auf Verzeichnisdiensten ausgeführt werden kann, ist das Suchen. Sie wird mit dem Kommando `ldapsearch` realisiert. Der allgemeine Aufruf erfolgt dabei gemäß der Syntax:

```
ldapsearch [Optionen] <Kriterium> [Attribute]
```

Die Suche erfolgt unter Verwendung diverser Optionen (siehe Tabelle 4.19) anhand des übergebenen Suchkriteriums.

Falls die Suche lediglich die Namen der Attribute, jedoch nicht deren Werte liefern soll, so kann dieses dem Kommando mit der Option `-A` mitgeteilt werden. Mit dem Parameter `-B` werden im Ergebnis auch Binärwerte angezeigt, und `-L` generiert die Ausgabe im LDIF-Format. Mit dem Parameter `-R` kann die automatische Verfolgung von Referrals unterdrückt werden, und durch `-u` wird eine zusätzliche Zeile ausgegeben, die eine benutzerfreundliche Angabe des Distinguished Name enthält. In der Ausgabe sind Attributname und -wert durch das Zeichen = getrennt. Mit der Option `-F` kann ein anderes Trennzeichen definiert werden. Falls die Ausgaben alphabetisch

nach dem Wert eines Attributs sortiert werden sollen, so ist die gewünschte Eigenschaft der Option `-S` zu übermitteln.

Option	Beschreibung
`-A`	Nur Attributnamen anzeigen
`-B`	Auch Binärwerte anzeigen
`-L`	Ausgabe im LDIF-Format
`-R`	Referrals nicht verfolgen
`-u`	Benutzerfreundliche Angabe des DN
`-F <Zeichen>`	Trennzeichen zwischen Attribut und -wert
`-S <Attribut>`	Sortierung nach Attribut
`-t`	Ergebnisse in Dateien
`-f <Datei>`	Verwenden einer Datei
`-h <Host>`	Hostname des LDAP-Servers
`-p <Port>`	Portnummer des LDAP-Servers
`-b <Basis>`	Basis der Suche
`-s <Wert>`	Reichweite der Suche
`-D <DN>`	Verbinden mit DN am DIT
`-w <Passwort>`	Zugehöriges Passwort
`-W`	Passwort abfragen
`-a <Wert>`	Auflösung von Alias-Definitionen
`-l <Wert>`	Zeit-Begrenzung
`-z <Wert>`	Begrenzung der Ergebnisse
`-v`	Zusätzliche Ausgaben
`-d <Level>`	Debug-Level

Tabelle 4.19: Optionen des Kommandos `ldapsearch`

Attribute, die binäre Werte (zum Beispiel Grafiken) enthalten, sollten nicht auf der Textkonsole ausgegeben werden. In einem solchen Fall kann der Parameter `-t` verwendet werden. Er speichert jeden Attributwert in einer temporären Datei ab. Ferner können die Aktionen, die dem Kommando übergeben werden müssen, auch aus einer Datei gelesen werden. Die nächsten beiden Optionen beschreiben den Hostnamen des LDAP-Servers (`-h`) sowie dessen TCP-Port (`-p`). Der Container, an dem die Suche beginnen soll, wird mit `-b`, der Suchbereich (`base`, `one`, default: `sub`) mit `-s` definiert.

Das Beglaubigen am Verzeichnisdienst geschieht durch Angabe des entsprechenden Objekts (`-D`) sowie dessen Passworts (`-w`). Das Kennwort kann auch interaktiv abgefragt werden (`-W`). Ob Alias-Definitionen aufgelöst werden sollen, wird mit `-a` definiert. Es sind die Werte `never`, `always`, `search` und `find` möglich, wobei `never` als Standard angenommen wird.

Wie viel Zeit für die Suchfunktion verwendet wird und wie viele Ergebnisse maximal geliefert werden, wird mit den Optionen `-l` und `-z` eingestellt.

Über `-d` und `-v` kann die Ausgabe weiter verfeinert werden.

Die über Optionen definierten Gegebenheiten überschreiben jene aus der Datei `ldap.conf`.

Das Suchkriterium ist nach der Syntax anzugeben, die im RFC 1558 (siehe [18]) definiert ist. In der einfachsten Form kann direkt nach einem Common Name gesucht werden:

```
"cn=FlemmingJ"
```

Ferner ist es auch möglich, zur Ersetzung von Zeichen das Symbol * zu verwenden. Es steht für beliebig viele Zeichen (auch keines). Beispiel:

```
"cn=Flem*"
```

Neben dieser trivialen Suche, ist es möglich, mehrere Teilkriterien zu einer Gesamtheit zu kombinieren (siehe Tabelle 4.20).

Operator	Beschreibung
&	Und-Verknüpfung
|	Oder-Verknüpfung
!	Negierung

Tabelle 4.20: Kombination von Suchkriterien

Eine Und-Verknüpfung erfolgt demnach mit dem Zeichen &, eine Oder-Verknüpfung mit | und eine Negierung mit !. Die Verknüpfungen erfolgen dabei grundsätzlich nach der Syntax:

```
"(<Operator>(<Teilkriterium 1>)(<Teilkriterium 2>)(…))"
```

Eine Und-Verknüpfung wird demnach zum Beispiel als

```
"(&(cn=FlemmingJ)(objectclass=person))"
```

dargestellt und eine Kombination mehrerer Verknüpfungen als:

```
"(&(objectClass=Person)(|(cn=Flem*)(cn=Paul*)))"
```

Nachdem nun Optionen und Suchkriterien beschrieben wurden, verbleiben noch die optional verwendbaren Attribute. Werden sie angegeben, so werden lediglich die aufgeführten Eigenschaften im Suchergebnis angezeigt. Anderenfalls erfolgt eine Angabe aller Attribute.

Beispiel 4.32: Aufruf von `ldapsearch` mit der Option `-z`

```
user1@host3:~ # ldapsearch -z 2 "objectclass=*"
o=Intern
objectclass=organization
o=Intern

ou=Einkauf, o=Intern
objectclass=organizationalUnit
ou=Einkauf
description="Abteilung Einkauf"
ldap_search: Sizelimit exceeded
user1@host3:~ #
```

Beispiel 4.33: Aufruf von `ldapsearch` mit der Option `-t`

```
user1@host3:~ # ldapsearch -t "ou=Ein*"
ou=Einkauf, o=Intern
objectclass=/tmp/ldapsearch-objectclass-sHEpnW
ou=/tmp/ldapsearch-ou-YT2UuI
description=/tmp/ldapsearch-description-MH9PDu
user1@host3:~ # cat /tmp/ldapsearch-description-MH9PDu
"Abteilung Einkauf"
user1@host3:~ #
```

Beispiel 4.34: Aufruf von `ldapsearch` mit der Option `-F`

```
user1@host3:~ # ldapsearch -F "#" "cn=FlemmingJ" cn userpassword
cn=FlemmingJ, ou=Verkauf, o=Intern
cn#FlemmingJ
user1@host3:~ #
```

Beispiel 4.35: Aufruf von `ldapsearch` mit den Optionen `-D -w`

```
user1@host3:~ # ldapsearch -D "cn=FlemmingJ, ou=Verkauf, \
# o=Intern" -w geheim "cn=FlemmingJ" cn userpassword
cn=FlemmingJ, ou=Verkauf, o=Intern
cn=FlemmingJ
userpassword=geheim
user1@host3:~ #
```

Beispiel 4.36: Aufruf von `ldapsearch` mit der Option `-f`

```
user1@host3:~ # cat filter.txt
(&(objectclass=per*)(cn=Flem*))
user1@host3:~ # ldapsearch -f filter.txt "%s" description
cn=FlemmingJ, ou=Verkauf, o=Intern
description="Jan Flemming"
user1@host3:~ #
```

Im letzten Beispiel wurde im Kriterium die Zeichenkette `%s` verwendet. Sie steht als Platzhalter für die Zeile aus der über den Parameter `-f` angegebenen Datei.

4.6.7 Das Kommando `ldapadd`

Das Hinzufügen von Objekten erfolgt mit dem Kommando `ldapadd`. Es wird allgemein mit

```
ldapadd [Optionen]
```

aufgerufen. In den Optionen wird dann unter anderem festgehalten, in welcher Datei sich die zu ergänzenden Objekte befinden (siehe Tabelle 4.21).

Option	Beschreibung
`-f <Datei>`	Verwenden einer Datei
`-h <Host>`	Hostname des LDAP-Servers
`-p <Port>`	Portnummer des LDAP-Servers
`-D <DN>`	Verbinden mit DN am DIT
`-w <Passwort>`	Zugehöriges Passwort
`-W`	Passwort abfragen
`-b`	Deute / in Werten als Pfad
`-c`	Bei Fehlern nicht abbrechen
`-v`	Zusätzliche Ausgaben
`-d <Level>`	Debug-Level

Tabelle 4.21: Optionen des Kommandos `ldapadd`

Der Wert eines Attributs wird in der Regel aus dem Text gedeutet. Falls der Anwender jedoch einen Wert nicht direkt angeben möchte, sondern dazu eine Datei heranziehen will, so kann er als Attributwert den Pfad zu der Datei angeben. Damit dieser Pfad entsprechend aufgelöst wird, muss die Option `-b` verwendet werden. Der Parameter `-c` gibt an, dass das Kommando bei Fehlern und Warnungen nicht abbricht, sondern fortfährt. Welche Objekte und Attribute zu dem Directory Information Tree

hinzugefügt werden sollen, wird über eine Datei definiert, die dem Kommando mit der Option `-f` mitgeteilt wird. Alle anderen Parameter haben die gleiche Bedeutung wie beim Kommando `ldapsearch` in der Tabelle 4.19 auf Seite 107.

Welche Objekte mit welchen Attributen hinzugefügt werden, wird vom Anwender in einer Textdatei festgehalten. Sie muss dem Format LDIF genügen.

Beispiel 4.37: Aufruf von `ldapadd` mit der Option `-f`

```
user1@host3:~ # cat neu.ldif
dn: cn=SchreinerW, ou=Verkauf, o=Intern
objectclass: person
cn: SchreinerW
sn: Schreiner
userpassword: geheim
telephonenumber: 251
description: "Werner Schreiner"
user1@host3:~ # ldapadd -f neu.ldif
adding new entry cn=SchreinerW, ou=Verkauf, o=Intern
ldap_add: Insufficient access

user1@host3:~ #
```

Beispiel 4.38: Aufruf von `ldapadd` mit den Optionen `-D -w -f`

```
user1@host3:~ # cat neu.ldif
dn: cn=SchreinerW, ou=Verkauf, o=Intern
objectclass: person
cn: SchreinerW
sn: Schreiner
userpassword: geheim
telephonenumber: 251
description: "Werner Schreiner"
user1@host3:~ # ldapadd -D "cn=Admin, o=Intern" -w linux \
# -f neu.ldif
adding new entry cn=SchreinerW, ou=Verkauf, o=Intern

user1@host3:~ #
```

Beispiel 4.39: Aufruf von `ldapadd` mit der Option `-p`

```
user1@host3:~ # cat neumitpfad.ldif
dn: cn=SchillerU, ou=Verkauf, o=Intern
objectclass: person
cn: SchillerU
sn: Schiller
userpassword: geheim
telephonenumber: 252
description: /home/user1/beschreibung
```

```
user1@host3:~ # cat beschreibung
"Udo Schiller"
user1@host3:~ # ldapadd -b -D "cn=Admin, o=Intern" -w linux \
# -f neu.ldif
adding new entry cn=SchillerU, ou=Verkauf, o=Intern

user1@host3:~ # ldapsearch "cn=Schill*"
cn=SchillerU, ou=Verkauf, o=Intern
objectclass=person
cn=SchillerU
sn=Schiller
telephonenumber=252
description="Udo Schiller"
user1@host3:~ #
```

4.6.8 Das Kommando `ldapmodify`

Neben dem Hinzufügen von Einträgen zu der LDAP-Datenbank besteht des Weiteren die Notwendigkeit, bestehende Objekte und Attribute auch ändern zu können. Dazu benutzt man das Kommando `ldapmodify`:

```
ldapmodify [Optionen]
```

Die möglichen Optionen (siehe Tabelle 4.22) entsprechen denen der Prozedur zum Hinzufügen (siehe Tabelle 4.21 auf Seite 110). Zusätzlich existieren lediglich zwei Parameter: Mit `-a` wird `ldapmodify` zum Hinzufügen von Objekten verwendet. Es hat dann exakt die gleiche Funktionalität wie `ldapadd`. Über den Parameter `-r` wird definiert, dass bereits vorhandene Werte beim Ändern per Default mit den neuen Vorgaben überschrieben werden.

Die dem Kommando zu übergebende Datei (`-f`), in der Änderungen aufgeführt sind, die durchgeführt werden sollen, wird im Folgenden bezüglich Ihres Formates beschrieben. Es entspricht prinzipiell dem Format der Datei `slapd.replog`, das in Abschnitt 4.5.9 auf Seite 90 bereits beschrieben wurde. Jedoch gelten für die Nutzung als Eingabedatei des Kommandos `ldapmodify` einige Ausnahmen:

- Zeilen, die mit dem Wort `replica` beginnen, werden dahingehend geprüft, ob es sich bei dem dort angegebenen Server um den handelt, auf dem die Änderungen durchgeführt werden. Anderenfalls wird die gewünschte Aktion nicht durchgeführt. `replica`-Zeilen sind optional.
- Die Angabe der Zeit mittels `time` entfällt.
- Falls in der Datei der `changetype` nicht angegeben ist, so wird bei Verwendung des Parameters `-a` der Befehl `add`, ansonsten `modify` verwendet.

- Falls als Änderungstyp `modify` verwendet wird, ohne die Art der Änderung näher anzugeben, so wird bei Verwendung der Option `-r` der Befehl `replace`, anderenfalls `add` für die Attribute angenommen.

Option	Beschreibung
`-f <Datei>`	Verwenden einer Datei
`-h <Host>`	Hostname des LDAP-Servers
`-p <Port>`	Portnummer des LDAP-Servers
`-D <DN>`	Verbinden mit DN am DIT
`-w <Passwort>`	Zugehöriges Passwort
`-W`	Passwort abfragen
`-b`	Deute / in Werten als Pfad
`-c`	Bei Fehlern nicht abbrechen
`-v`	Zusätzliche Ausgaben
`-d <Level>`	Debug-Level
`-a`	Hinzufügen
`-r`	Vorhandene Werte ersetzen

Tabelle 4.22: Optionen des Kommandos `ldapmodify`

Die obigen Regeln beschreiben somit, welche Defaultaktionen durchgeführt werden, sofern diese nicht explizit angegeben wurden. Es empfiehlt sich daher, die Datei möglichst so zu beschreiben, dass keine der Standardregeln angewendet werden müssen, da sie mitunter nicht jedem Administrator auf Anhieb geläufig sind.

Beispiel 4.40: Aufruf von `ldapmodify` mit den Optionen `-D -w -f`

```
user1@host3:~ # cat aenderung.txt
dn: cn=SchreinerW, ou=Verkauf, o=Intern
changetype: modify
replace: description
description: "Wilfried Schreiner"
-

dn: cn=SchillerU, ou=Verkauf, o=Intern
changetype: modify
delete: telephonenumber
-
replace: userpassword
userpassword: secret
-
replace: description
```

```
description: "Ulrich Schiller"

user1@host3:~ # ldapmodify -D "cn=Admin, o=Intern" \
# -w linux -f aenderung.txt
modifying entry cn=SchreinerW, ou=Verkauf, o=Intern

modifying entry cn=SchillerU, ou=Verkauf, o=Intern

user1@host3:~ #
```

Zum Hinzufügen, Ändern und Löschen von Attributen kann das Kommando `ldapmodify` verwendet werden. Das Hinzufügen von Objekten ist jedoch nur mit `ldapadd` oder `ldapmodify -a` möglich.

4.6.9 Das Kommando `ldapmodrdn`

Die bisher vorgestellten Funktionen geben den Anwender die Möglichkeit, Objekte im Verzeichnisdienst zu ändern und zu erstellen. Das Umbenennen des Relative Distinguished Name eines Objekts kann mit dem Kommando `ldapmodrdn` durchgeführt werden:

```
ldapmodrdn [Optionen] [<DN> <RDN>]
```

Option	Beschreibung
`-f <Datei>`	Verwenden einer Datei
`-h <Host>`	Hostname des LDAP-Servers
`-p <Port>`	Portnummer des LDAP-Servers
`-D <DN>`	Verbinden mit DN am DIT
`-w <Passwort>`	Zugehöriges Passwort
`-W`	Passwort abfragen
`-v`	Zusätzliche Ausgaben
`-d <Level>`	Debug-Level
`-r`	Alten RDN löschen

Tabelle 4.23: Optionen des Kommandos `ldapmodrdn`

Durch Verwendung des Parameters `-r` wird das Kommando `ldapmodrdn` angeregt, den Objektnamen, der vor der Änderung aktuell war, zu löschen. Ohne diese Option würde der alte Name als Wert des Attributs `cn` erhalten bleiben. Die anderen in der Tabelle 4.23 aufgeführten Parameter haben die gleichen Optionen wie bei den bereits genannten Kommandos (siehe Abschnitt 4.6.6 auf Seite 106).

Falls ein Parameter angegeben wird, dessen Funktionalität auch über die Datei `/etc/openldap/ldap.conf` definiert wurde, werden die Einstellungen aus der Konfigurationsdatei missachtet.

Beispiel 4.41: Aufruf von `ldapmodrdn` mit den Optionen `-D -w`

```
user1@host3:~ # ldapsearch "cn=SchillerU" cn
cn=SchillerU, ou=Verkauf, o=Intern
cn=SchillerU
user1@host3:~ # ldapmodrdn -D "cn=Admin", o=Intern" -w linux \
# "cn=SchillerU, ou=Verkauf, o=Intern" "cn=UlrichS"
user1@host3:~ # ldapsearch "cn=UlrichS" cn
cn=UlrichS, ou=Verkauf, o=Intern
cn=SchillerU
cn=UlrichS
user1@host3:~ #
```

Beispiel 4.42: Aufruf von `ldapmodrdn` mit `-D -w -f -r`

```
user1@host3:~ # cat neuername.txt
cn=SchreinerW, ou=Verkauf, o=Intern
cn=WilfriedS
user1@host3:~ # ldapmodrdn -r -D "cn=Admin", o=Intern" \
# -w linux -f neuername.txt
user1@host3:~ # ldapsearch "cn=WilfriedS" cn
cn=WilfriedS, ou=Verkauf, o=Intern
cn=WilfriedS
user1@host3:~ #
```

4.6.10 Das Kommando `ldapdelete`

Das vorletzte in diesem Zusammenhang vorhandene Kommando dient zum Löschen von Objekten in der LDAP-Datenbank. Es trägt den Namen `ldapdelete` und wird nach der Syntax

```
ldapdelete [Optionen] [Distinguished Names]
```

aufgerufen. Die Bedeutung der Optionen, die dabei zur Verfügung stehen (siehe Tabelle 4.24), wurde bereits in Abschnitt 4.6.7 auf Seite 110 im Zusammenhang mit dem Kommando `ldapadd` beschrieben.

Auf diese Weise ist es möglich, sich mit der Datenbank zu verbinden, um dort ein oder mehrere Objekte zu entfernen. Dem Kommando können beliebig viele Objekte mitgeteilt werden, die gelöscht werden sollen. Auch ist es möglich, durch einen Filterausdruck gleichzeitig mehrere Einträge anzusprechen.

Option	Beschreibung
`-f <Datei>`	Verwenden einer Datei
`-h <Host>`	Hostname des LDAP-Servers
`-p <Port>`	Portnummer des LDAP-Servers
`-D <DN>`	Verbinden mit DN am DIT
`-w <Passwort>`	Zugehöriges Passwort
`-W`	Passwort abfragen
`-c`	Bei Fehlern nicht abbrechen
`-v`	Zusätzliche Ausgaben
`-d <Level>`	Debug-Level

Tabelle 4.24: Optionen des Kommandos `ldapdelete`

Beispiel 4.43: Aufruf von `ldapdelete` mit den Optionen `-D -w`

```
user1@host3:~ # ldapsearch "cn=WilfriedS"
cn=WilfriedS, ou=Verkauf, o=Intern
objectclass=person
cn=WilfriedS
sn=Schreiner
telephonenumber=241
description="Wilfried Schreiner"
user1@host3:~ # ldapdelete -D "cn=Admin, o=Intern" -w linux \
# "cn=WilfriedS, ou=Verkauf, o=Intern"
user1@host3:~ # ldapsearch "cn=WilfriedS"
user1@host3:~ #
```

Beispiel 4.44: Aufruf von `ldapdelete` mit den Optionen `-D -w -f`

```
user1@host3:~ # ldapsearch "cn=UlrichS"
cn=UlrichS, ou=Verkauf, o=Intern
objectclass=person
cn=UlrichS
cn=SchillerU
sn=Schiller
description="Ulrich Schiller"
user1@host3:~ # cat loeschen.txt
cn=UlrichS, ou=Verkauf, o=Intern
user1@host3:~ # ldapdelete -D "cn=Admin, o=Intern" -w linux \
# -f loeschen.txt "%s"
user1@host3:~ # ldapsearch "cn=UlrichS"
user1@host3:~ #
```

In Beispiel 4.44 wurde eine Datei verwendet, in der ein Objekt aufgeführt ist. Um den Inhalt der Datei zum Löschen benutzen zu können, muss deren Pfad dem Kommando `ldapdelete` über die Option `-f` mitgeteilt werden. Bei der Angabe des Distinguished Name des zu entfernenden Objekts kann anschließend der Platzhalter `%s` aufgeführt werden. Er steht für den Inhalt einer Dateizeile.

4.6.11 Das Kommando `ldappasswd`

In Abschnitt 5.3 auf Seite 203 wird die zentrale Verwaltung von Benutzerpasswörtern im Netzwerk per LDAP dargestellt. Dazu ist es notwendig, dass den Anwendern ein einfaches Verfahren zur Verfügung gestellt wird, mit dem sie ihr Kennwort ändern können. Ferner ist die Passwortverwaltung auch für den Administrator ein äußerst wichtiger Bestandteil seiner Arbeit. So ist es jedem LDAP-Client möglich, mit dem Kommando `ldappasswd` das Kennwort-Attribut eines Objekts zu verändern. Die Voraussetzung dafür ist, wie bei allen bisher erwähnten Aktionen, dass derjenige, der die Aktion durchführt, die notwendigen Rechte im Directory Information Tree besitzt. Für den Administrator besteht mit diesem Kommando auch die Möglichkeit, mehrere Passwörter gleichzeitig zu setzen oder Kennwörter automatisch generieren zu lassen.

```
ldappasswd [Optionen] [Suchfilter]
```

Durch die Verwendung einer oder mehrerer Optionen aus Tabelle 4.25 kann das Kommando in seiner Funktionalität beeinflusst werden. Das neue Passwort für das über die Parameter angegebene Objekt wird dann interaktiv abgefragt.

Beim Aufruf des Kommandos `ldappasswd` kann diesem mitgeteilt werden, wie der Name des Attributs lautet, in dem das Benutzerkennwort abgespeichert ist. Fehlt die Angabe dieses Parameters, so wird `-a userpassword` verwendet. Das Kennwort, das mit dieser Funktion neu gesetzt werden soll, kann auf verschiedene Weise im LDAP-Verzeichnisdienst eingetragen werden. Neben einer Eintragung im Klartext, ist es ferner möglich, Verschlüsselungen zu verwenden, die nach den Hashing-Verfahren

- `crypt`,
- `md5`,
- `smd5`,
- `sha` oder
- `ssha`

generiert werden. Die Angabe des Hashing-Verfahrens erfolgt über den Parameter `-H`. Ihm wird das Kürzel der Codierung mitgeteilt. Per Default wird davon ausgegangen, dass die Kennwörter wie unter Linux üblich nach `crypt` codiert werden.

Option	Beschreibung
`-a <Attribut>`	Name des Passwort-Attributs
`-H <Wert>`	Verschlüsselung des Kennworts
`-g <Anzahl>`	Passwortlänge bei automatischer Generierung
`-b <Basis>`	Startpunkt der Suche
`-s <Reichweite>`	Reichweite der Suche
`-h <Host>`	Hostname des LDAP-Servers
`-p <Port>`	Portnummer des LDAP-Servers
`-D <DN>`	Verbinden mit DN am DIT
`-w <Passwort>`	Zugehöriges Passwort
`-W`	Passwort abfragen
`-t <DN>`	Zielobjekt
`-v`	Zusätzliche Ausgaben (auch `-vvv`)
`-d <Level>`	Debug-Level
`-l <Wert>`	Zeitbegrenzung
`-s <Wert>`	Begrenzung der Anzahl

Tabelle 4.25: Optionen des Kommandos `ldappasswd`

Das Kommando `ldappasswd` kann vom Systemadministrator auch dazu verwendet werden, Passwörter automatisch generieren zu lassen. Falls das Mittel der Passwortgenerierung benutzt wird, muss man dem Kommando über die Option `-g` die gewünschte Länge der Kennwörter mitteilen. Bei der Generierung besteht ferner die Möglichkeit, die automatisch erstellten Einträge parallel auf der Konsole ausgeben zu lassen. Dies ist durch die Verwendung des Parameters `-vvv` möglich. Mit `ldappasswd` ist es nicht nur möglich, das Passwort-Attribut eines Objekts zu ändern, sondern es können auch direkt mehrere Objekte im Verzeichnisdienst angesprochen werden. Dies wird dadurch realisiert, dass dem Kommando optional ein `[Suchfilter]` übergeben wird. Falls er zum Beispiel aus der Zeichenkette `"objectclass=person"` besteht, so werden alle Benutzerobjekte angesprochen. Die Suche nach `person`-Objekten beginnt in diesem Fall an dem über die Option `-b` angegebenen Distinguished Name eines Containerobjekts. Die Reichweite der Suche wird mit dem Parameter `-s` festgelegt. Ihm wird einer der Werte

- `base` (Suche auf der Basis),
- `one` (Suche im angesprochenen Container) oder

- `sub` (Suche im gesamten Teilbaum)

übergeben, wobei die Einstellung `sub` als Default vorhanden ist.

Alle in der Konfigurationsdatei `/etc/openldap/ldap.conf` des LDAP-Clients festgelegten Informationen werden für die LDAP-Kommandos zur Realisierung ihrer Funktionalität herangezogen. So sind in der Datei auch die Einstellungen des LDAP-Servers und TCP-Ports vorhanden. Falls in den Kommandos jedoch abweichende Parameter für den Host und den Port verwendet werden sollen, so geschieht dies durch Angabe der Optionen `-h` und `-p`. Die an sie übergebenen Werte überschreiben die Einstellungen der Konfigurationsdatei.

Zum Ändern von Kennwörtern im Verzeichnisdienst ist es notwendig, dass dem Kommando `ldappasswd` mitgeteilt wird, mit welcher Kennung es sich am Verzeichnisdienst anmelden soll und welches Objekt es bearbeiten soll. Der Distinguished Name des Objekts, mit dem die Anmeldung erfolgen soll, wird über die Option `-D` angegeben, sein Passwort über `-w`. Falls die Abfrage des Kennwortes interaktiv erfolgen soll, kann stattdessen `-W` verwendet werden. Anschließend muss noch definiert werden, an welchem Zielobjekt das Attribut `userpassword` geändert werden soll. Der DN des Objekts wird dem Kommando über `-t` mitgeteilt.

Um während der Passwortänderung detaillierte Ausgaben auf der Textkonsole zu bekommen, können zwei Parameter definiert werden. Während die Option `-d` den Debug-Level festlegt, führt `-v` allgemein zu mehr Ausgaben. Eine Vermehrung dieser Ausgaben kann mit `-vvv` angeregt werden.

Falls dem Kommando ein Suchfilter übergeben wird, um gleichzeitig die Kennwörter bei mehreren Objekten zu ändern, so kann die Zeit, die für diese Suche in Anspruch genommen wird, genauso bestimmt werden wie die maximale Anzahl der Ergebnisse. Die mit den Parametern `-l` und `-z` definierten Werte überschreiben die Definitionen von `TIMELIMIT` und `SIZELIMIT` der Datei `ldap.conf`.

Beispiel 4.45: Aufruf von `ldappasswd` mit den Optionen `-D -w -t`

```
user1@host3:~ # ldappasswd -D "cn=Admin, o=Intern" -w linux \
# -t "cn=FlemmingJ ,ou=Verkauf, o=Intern"
New password:
Re-enter new password:
user1@host3:~ #
```

Beispiel 4.46: Aufruf von `ldappasswd` mit den Optionen `-D -w -g`

```
user1@host3:~ # ldappasswd -D "cn=Admin, o=Intern" -w linux \
# -t "cn=FlemmingJ ,ou=Verkauf, o=Intern" -g 6 -vvv
New password:
Re-enter new password:
cn=FlemmingJ, ou=Verkauf, o=Intern:{crypt}V4dopeTZsXpLc:gUhlJ4@
user1@host3:~ #
```

In Beispiel 4.46 sind die Fragen nach einem neuen Passwort lediglich mit **Enter** zu bestätigen, da das Kennwort automatisch generiert wird.

Beispiel 4.47: Aufruf von `ldappasswd` mit `-D -w -b -H`

```
user1@host3:~ # ldappasswd -D "cn=Admin, o=Intern" -w linux \
# -b "ou=Verkauf, o=Intern" -vvv -H none \
# "objectclass=person"
New password:
Re-enter new password:
cn=JohansenL, ou=Verkauf, o=Intern:neuespw:neuespw
cn=PaulsenI, ou=Verkauf, o=Intern:neuespw:neuespw
cn=FlemmingJ, ou=Verkauf, o=Intern:neuespw:neuespw
user1@host3:~ #
```

Das neu eingegebene Passwort wurde für alle Benutzer des Verkaufs im Klartext gesetzt.

4.6.12 Das Programm `ud`

Neben den vorgestellten Kommandos existiert ein LDAP-Client, der dem Anwender die Möglichkeit gibt, interaktiv Änderungen am Verzeichnisdienst vorzunehmen. Das Programm `ud` steht für User Directory und beschränkt sich auf zwei Objektklassen, die mit ihm verwaltet werden können:

- Benutzerobjekte
- Gruppenobjekte

Benutzerobjekte wurden in diesem Buch bereits behandelt. Ein Objekt der Klasse Gruppe besteht aus einer Menge von Benutzern. So wäre es möglich, eine Gruppe namens `Textverarbeitung` zu erstellen. Sie hat eine Eigenschaft `members`, in der alle Benutzer aufgeführt werden, die Mitglied dieser Gruppe sind. Die so definierten Objekte und Eigenschaften können später zum Beispiel dazu herangezogen werden, den Mitgliedern die nötigen Dateisystemrechte zum Starten der Textverarbeitung zu geben.

Das Programm `ud` arbeitet in der Textkonsole und ist allgemein in der Form

```
ud [Optionen]
```

aufzurufen. Die Anwendung wird anschließend unter Verwendung der übergebenen Parameter gestartet (siehe Tabelle 4.26).

Die sicher wichtigste Einstellung, die dem Programm `ud` mitgeteilt werden muss, ist der Hostname oder die IP-Adresse des LDAP-Servers, auf dem dieses Programm die LDAP-Datenbank findet. Der Option `-s` kann außer dem Namen oder der IP-Adresse

auch die Angabe eines TCP-Ports folgen. Er ist mit einem Doppelpunkt getrennt hinter dem Host anzugeben:

```
192.168.17.1:389
```

Fehlt die Angabe des Ports, so wird `389` angenommen.

Option	Beschreibung
`-s <Host>`	Hostname des LDAP-Servers
`-f <Datei>`	Pfad zur Konfigurationsdatei
`-v`	Zusätzliche Ausgaben
`-l <Level>`	LDAP-Debug-Level
`-d <Level>`	`ud`-Debug-Level
`-D`	Ausgabe der `ud`-Debug-Level

Tabelle 4.26: Optionen des Programms `ud`

Das Programm `ud` kann nicht nur mit Optionen beim Aufruf, sondern auch von einer Konfigurationsdatei entsprechend eingerichtet werden. Sie wird in Abschnitt 4.6.13 auf Seite 125 beschrieben. Der Pfad zu ihr kann mit dem Parameter `-f` definiert werden, sofern er von `/etc/openldap/ud.conf` abweicht.

Die verbleibenden vier Parameter `-v`, `-l`, `-d` und `-D` dienen dazu, detaillierte Ausgaben und Meldungen während des Programmlaufs zu erhalten. Dabei führt `-v` allgemein dazu, dass das Programm etwas mehr Meldungen ausgibt. Das Debuggen von LDAP-Aktionen und von Aktionen des Programms `ud` wird über die Parameter `-l` und `-d` eingestellt. Ihnen folgt ein Wert, der beschreibt, wie viele und welche Tatbestände debuggt werden sollen. Die möglichen Werte für `-d` können durch Verwendung der Option `-D` eingesehen werden.

Beispiel 4.48: Aufruf von `ud` mit der Option `-D`

```
user1@host3:~ # ud -D

  Debug flag values

    1  function trace
    2  find() information
    4  group information
    8  mod() information
   16  parsing information
   32  output information
   64  authentication information
  128  initialization information
```

```
  These are masks, and may be added to form multiple
  debug levels.  For example, '-d 35' would perform
  a function trace, print out information about the
  find() function, and would print out information
  about the output routines too.

user1@host3:~ #
```

Abbildung 4.7 zeigt den Programmstart mit der Option `-s`, wie bei:

```
user1@host3:~ # ud -s host1
```

Abbildung 4.7: Start des Programmes ud

Nach der Ausgabe der Begrüßungsmeldung hat der Anwender die Möglichkeit, verschiedene Kommandos hinter dem ud-Prompt * einzugeben. In Abbildung 4.7 wurde die Zeichenkette `status` eingegeben. Sie liefert eine Ausgabe, in der man die zurzeit von ud verwendeten Werte sehen kann. Dazu gehören der Name des Servers (`Current server`), der Startpunkt im Verzeichnisdienst für Suchaktionen (`Search base is`), das Containerobjekt, unter dem der Anwender, sofern er dazu berechtigt ist, Gruppen anlegen kann (`Group base is`), und der Name des Objekts, mit dem die Beglaubigung am Verzeichnisdienst erfolgte (`Bound as Nobody`). Falls wie in diesem Beispiel noch keine Anmeldung am Verzeichnisdienst erfolgt ist, so hat der ud-Anwender lediglich die Rechte eines Gastbenutzers. Die letzten beiden Zeilen geben an, ob das Programm detaillierte Ausgaben liefern soll (`Verbose mode is enabled`) und ob Aliase im Verzeichnisdienst aufzulösen sind (`Aliases are being dereferenced`).

Welche Kommandos der Anwender im User Directory-Programm generell nutzen kann, wird ihm nach der Eingabe von `h` oder `?` angezeigt (siehe Abbildung 4.8).

```
  ?                   To print this list.
  bind [who]          To bind (authenticate) to the directory.
  cb [where]          To change the search base.
  change [entry]      To change information associated with an entry.
  create [group]      To create a new group entry.
  dereference         To toggle dereferencing of aliases.
  vedit [entry]       To edit a complete Directory entry using your editor.
  find [entry]        To find an entry in the directory.
  groupbase [where]   To change the group base.
  help [command]      To display detailed help for a particular command.
  join [group]        To subscribe to a group.
  list [who]          To list the groups owned by someone.
  memberships [who]   To list out the groups in which someone is a member.
  purge [group]       To remove obsolete entries from a group.
  quit                To terminate the program.
  remove [group]      To remove a group entry.
  resign [group]      To unsubscribe from a group.
  status              To display directory connection status.
  tidy                To unsubscribe from groups that no longer exist.
  verbose             To toggle the verbose switch.

  Type "help <command-name>" to get help about a particular command.
  Type "help options" to get help about options in brackets above.
* []
```

Abbildung 4.8: Kommandoübersicht des Programms `ud`

Die Anmeldung am Verzeichnisdienst erfolgt mit `bind`, die Suchbasis kann mit `cb` eingestellt werden. Das Ändern eines Objekts ist mit `change`, das Erstellen einer Gruppe mit `create` möglich. Ob Aliase aufgelöst werden sollen, kann mit `dereference` eingestellt werden. Benutzerobjekte können auch mit der Hilfe des Standardeditors bearbeitet werden. Er wird gestartet, nachdem das Kommando `vedit` verwendet wurde. Ferner ist es möglich, im Verzeichnisdienst zu suchen (`find`), den Container für die Gruppenerstellung einzustellen (`groupbase`), sich selbst in eine Gruppe einzutragen (`join`) oder alle Gruppen aufzulisten (`list`), von denen ein bestimmtes Objekt als Gruppeneigentümer geführt wird. Um alle Gruppen angezeigt zu bekommen, in denen ein bestimmter Benutzer Mitglied ist, kann `memberships` verwendet werden. Falls in einer Gruppe Mitglieder aufgeführt sind, die im Verzeichnisdienst gar nicht mehr existieren, so können sie mit `purge` ausgetragen werden.

Das Beenden des Programms erfolgt mit `quit`, das Löschen einer Gruppe mit `remove` und das Entfernen des eigenen Benutzerobjektes aus einer Gruppe mit `resign`. Neben der bereits bekannten Status-Funktionalität (`status`) kann ferner mit `verbose` die Anzahl der `ud`-Ausgaben beeinflusst werden. Mit `tidy` ist es möglich, sich aus Gruppen auszutragen, die nicht mehr existieren.

Falls zu einem Kommando eine etwas ausführlichere Hilfe gewünscht wird, so kann sie mit

```
* help [Kommando]
```

eingesehen werden.

In Abbildung 4.9 wurde die Basis für Suchaktionen auf den Container `Verkauf` gesetzt. Anschließend erfolgte eine Suche nach dem Objekt `FlemmingJ`. Die Beglaubigung am Verzeichnisdienst wurde ebenfalls mit dem Benutzer `FlemmingJ` und dem Passwort `geheim` vollzogen. Somit kann der Anwender nun alle Aktionen am Directory Information Tree durchführen, für die `FlemmingJ` die notwendigen Rechte besitzt.

```
LDAP UserDirectory 1.2.11-Release (Sat Jul 29 13:21:56 GMT 2000)

  Enter a command.  If you need help, type 'h' or '?' and hit RETURN.

* cb ou=Verkauf, o=Intern
  Search base is now Verkauf, Intern
* find FlemmingJ
  Found one exact match for "FlemmingJ"
 "FlemmingJ"
  Business phone:      200
* bind FlemmingJ
  Authenticating to the directory as "FlemmingJ"...
  Enter your LDAP password:
  Authentication successful.

*
```

Abbildung 4.9: Beglaubigung im Programm `ud`

Mit dem Kommando

```
* vedit FlemmingJ
```

wird der über die Umgebungsvariable `EDITOR` definierte Standardeditor aktiviert. In ihm kann das angegebene Objekt dann entsprechend verändert werden (siehe Abbildung 4.10).

Zeilen, die mit den Zeichen *##* beginnen, markieren einen Kommentar. Zu dem Objekt wird in dieser Datei eine Auswahl von Attributen aufgeführt. Sie sind generell in der Form

```
## Directory entry of FlemmingJ
##
## Syntax is:
## <attribute-name>
##      <TAB> <value 1>
##      <TAB>   :  :
##      <TAB> <value N>
## Lines beginning with a hash mark are comments.
##
title
postalAddress
telephoneNumber
        200
mail
homePhone
homePostalAddress
facsimileTelephoneNumber
pager
labeledURL
onVacation
vacationMessage
drink

"/tmp/file4cNOG6" 22L, 328C                          1,1           All
```

Abbildung 4.10: Editieren eines Benutzerobjekts mit `ud`

```
<Attribut>
        <Wert>
        <Wert>
        <…>
```

vorhanden, wobei die Attributwerte mit der **TAB**-Taste eingerückt sind. Die bereits vorhandenen Eigenschaften können nun beliebig verändert werden. Falls ein Attribut angesprochen werden soll, das standardmäßig nicht angezeigt wird, so kann es einfach per Hand hinzugefügt werden.

Nachdem alle gewünschten Änderungen vollzogen wurden, kann man den Editor verlassen. Je nachdem, um welchen Editor es sich handelt, sind die Tastenkombinationen zu benutzen, die der Aktion ‚Speichern und Beenden' entsprechen. Falls es sich um den Editor `vi` handelt, so ist im Kommandozeilenmodus `:wq` einzugeben und mit **Enter** zu bestätigen.

4.6.13 Die Konfigurationsdatei `ud.conf`

Dem Programm `ud` können beim Aufruf einige Eigenschaften mit den Optionen mitgeteilt werden. Sie beschreiben zum Beispiel den DNS-Namen oder die IP-Adresse des Servers und werden dann beim Start verwendet. Es besteht jedoch auch die Möglichkeit, einige der Einstellungen in einer Konfigurationsdatei festzuhalten. Bei dieser Datei handelt es sich um ein ASCII-basiertes Dokument, das per Default im Verzeichnis `/etc/openlap/` anzulegen ist. Sie trägt den Namen `ud.conf`. Ist sie vorhanden,

kommen ihre Einstellungen grundsätzlich zur Anwendung. Falls eine andere Datei verwendet werden soll, kann deren Pfad dem Programm mit der Option `-f` mitgeteilt werden.

Syntaktisch besteht die Konfigurationsdatei aus mehreren Zeilen, die jeweils eine Einstellung beschreiben. Leerzeilen und Kommentare sind nicht möglich (siehe Tabelle 4.27).

Parameter	Beschreibung
`HOST`	Hostname des LDAP-Servers
`BASE`	Basis der Suchaktionen
`GROUPBASE`	Basis für die Erstellung von Gruppen
`SEARCH`	Definition von Suchalgorithmen

Tabelle 4.27: Parameter der Datei `ud.conf`

Der Parameter `HOST`

Bei dem ersten Parameter handelt es sich um die Angabe des LDAP-Servers. Erfolgt in der Konfigurationsdatei eine Anweisung nach dem Muster

```
HOST <Hostname>
```

so wird sie beim Programmstart verwendet. Die Option `-s` ist dann nicht mehr notwendig:

```
HOST host1
```

Wird beim Start von `ud` ebenfalls der LDAP-Server über `-s` definiert, so wird der Wert der Konfigurationsdatei missachtet.

Der Parameter `BASE`

Suchaktionen im Verzeichnisdienst beginnen an dem Container, der als so genannte `Search Base` definiert wurde. Dieser Container kann interaktiv im Programm `ud` mit dem Kommando `cb` eingestellt werden. Er ist per Default auf die Wurzel der LDAP-Datenbank fixiert. Um den Standard zu ändern, kann in der Datei `ud.conf` der Parameter `BASE` verwendet werden:

```
BASE <Distinguished Name>
```

Ihm ist der Distinguished Name des gewünschten Containerobjekts zu übergeben, z. B.:

```
BASE ou=Verkauf, o=Intern
```

Der Parameter GROUPBASE

Falls das User Directory-Programm dazu benutzt wird, Gruppen im Verzeichnisdienst zu erstellen, kann der Container, unter dem neu anzulegende Gruppen angeordnet werden sollen, mit dem Kommando `changegroup` im Programm `ud` angegeben werden. Falls jedoch generell beim Start des Programms ein Standardwert benutzt werden soll, so kann er in der Konfigurationsdatei mit dem Parameter `GROUPBASE` eingestellt werden:

```
GROUPBASE <Distinguished Name>
```

Auch ihm wird der DN des gewünschten Objekts übergeben, z. B.:

```
GROUPBASE ou=Verkauf, o=Intern
```

Der Parameter SEARCH

Falls mit dem Programm `ud` Suchaktionen durchgeführt werden sollen, kann in der Konfigurationsdatei angegeben werden, wie der vom Anwender eingegebene Suchbegriff gedeutet werden soll und mit welchem Algorithmus die Suche letztendlich am Verzeichnisdienst auszuführen ist. Es ist ferner möglich, mehrere Suchverfahren zu definieren. Sie werden der Reihe nach so lange angewendet, bis eine geeignete Antwort gefunden wird. Die Definition eines Verfahrens erfolgt mit dem `SEARCH`-Parameter:

```
SEARCH <Suchverfahren>
```

Ein Suchverfahren wird durch drei verschiedene Aspekte repräsentiert:

- Als Erstes ist ein Attribut anzugeben, auf das sich die Suche beziehen soll. So wäre beispielsweise das Attribut `cn` denkbar.
- Des Weiteren wird festgehalten, wie der Vergleich des vom Anwender definierten Suchbegriffs mit dem zuvor aufgeführten Attribut vonstatten gehen soll. Die Prüfung auf Gleichheit wird entsprechend mit dem Symbol = dargestellt.
- Als Drittes muss der Suchbegriff des Anwenders repräsentiert werden. Er kann durch `$0` angegeben werden.

Weitere Einzelheiten zum Parameter `SEARCH` sind in der Manual Page `man 5 ud.conf` vorhanden. In der Regel ist es jedoch nicht notwendig, diesen Parameter zu definieren.

Beispiel 4.49: Konfigurationsdatei des Programms ud

```
root@host3:~ # cat /etc/openldap/ud.conf
HOST            host1
BASE            ou=Verkauf, o=Intern
GROUPBASE       ou=Verkauf, o=Intern
root@host3:~ #
```

Die Einstellungen in der Datei `/etc/openldap/ud.conf` gelten für alle Benutzer systemweit auf einem Linux-Rechner. Es ist jedoch auch möglich, die Konfigurationen, die global vorgenommen werden können, benutzerspezifisch abzuspeichern. Dazu kann im Heimatverzeichnis des Anwenders die Datei `.udrc` angelegt werden. Beim Start des Programms `ud` werden die Einstellungen in der folgenden Reihenfolge gesucht.

1. Zunächst wird geprüft, welche Optionen beim Kommandoaufruf angegeben wurden. Sie haben grundsätzlich die höchste Priorität.
2. Anschließend wird die Datei `.udrc` im Heimatverzeichnis des Benutzers gesucht, der das Programm `ud` gestartet hat. Existiert sie, so kommen ihre Einstellungen zur Anwendung.
3. Zum Schluss werden die Konfigurationen der Datei `ud.conf` geprüft.

4.6.14 Das Programm `kldap`

Damit Software-Pakete und Dienste von einem großen Personenkreis akzeptiert werden, ist es unumgänglich, auch grafische Anwendungen zur Verfügung zu stellen. Das folgende Programm befindet sich zwar noch in der Entwicklungsphase, ist jedoch im täglichen Umgang mit Verzeichnisdiensten hilfreich und unverzichtbar. Das Programm trägt den Namen `kldap` und wird unter der grafischen Oberfläche KDE gestartet.

`kldap` ist in der Lage, mit einer leicht zu bedienenden Oberfläche die folgenden Aktionen durchzuführen:

- Durchsuchen des Verzeichnisdienstes
- Beglaubigung am Verzeichnisdienst
- Erstellen von Objekten
- Erstellen von Attributen
- Ändern von Attributen
- Löschen von Objekten und Attributen
- Suchen im Verzeichnisdienst

Das Programm `kldap` wird in der grafischen Oberfläche ohne die weitere Angabe von Optionen verwendet.

```
user1@host3:~ # kldap &
user1@host3:~ #
```

Nach dem Start (siehe Abbildung 4.11) hat der Anwender die Möglichkeit, aus dem Menü des Programms eine der gewünschten Funktionen auszuwählen. Die passenden Menüpunkte sind in Tabelle 4.28 aufgeführt.

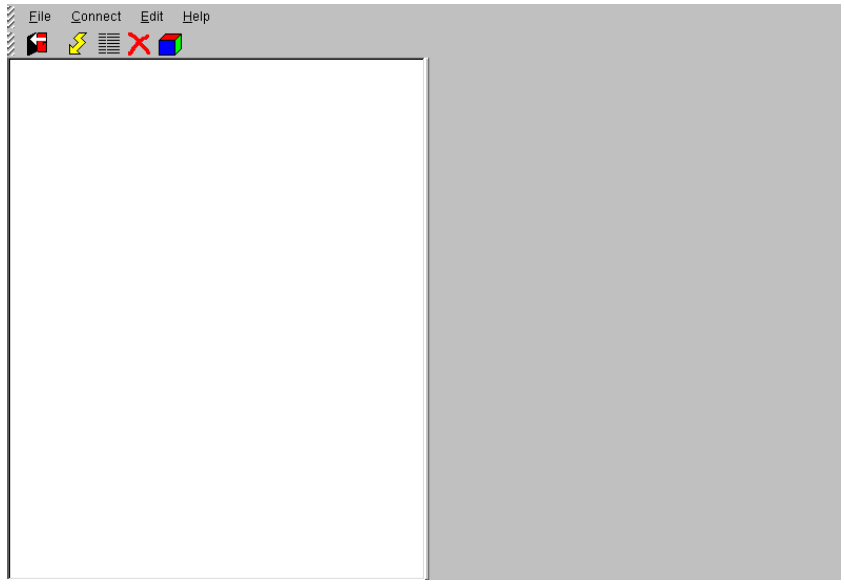

Abbildung 4.11: `kldap`: Grafische Oberfläche

Der Menüpunkt FILE hat derzeit lediglich einen Unterpunkt. Er trägt den Namen EXIT und dient zum Beenden von `kldap`. Auch unter dem Menüpunkt CONNECT existiert zurzeit nur eine einzige Auswahlmöglichkeit namens CONNECT. Mit ihr wird die Verbindung zu einem LDAP-Server hergestellt.

Das Suchen, Hinzufügen und Löschen von Objekten sowie die Änderung von Attributen erfolgt mit der Auswahl der Punkte SEARCH, ADD..., DELETE und EDIT ATTRIBUTES..., die sich im EDIT-Menü befinden. Außerdem existieren unter HELP Hilfen zum Programm. In ihnen ist auch der Autor von `kldap`, Oliver Jaun, erwähnt.

Alternativ können einige der über das Menü erreichbaren Funktionen (siehe Tabelle 4.28) auch durch Anklicken des zugehörigen Icons ausgeführt werden. Auf diesem Wege ist es möglich, das Programm zu beenden, eine Verbindung zum LDAP-Server herzustellen, die Attribute einzusehen, ein Objekt zu löschen oder ein neues zu erstellen.

Bevor die zuvor genannten Möglichkeiten am Verzeichnisdienst genutzt werden können, muss zunächst definiert werden, mit welchen Werten die Anmeldung am LDAP-Server durchzuführen ist. Nach der Anwahl von CONNECT oder durch Anklicken des

entsprechenden Symbols wird in der rechten Bildschirmhälfte eine Eingabemaske angezeigt, die die Informationen abfragt (siehe Abbildung 4.12).

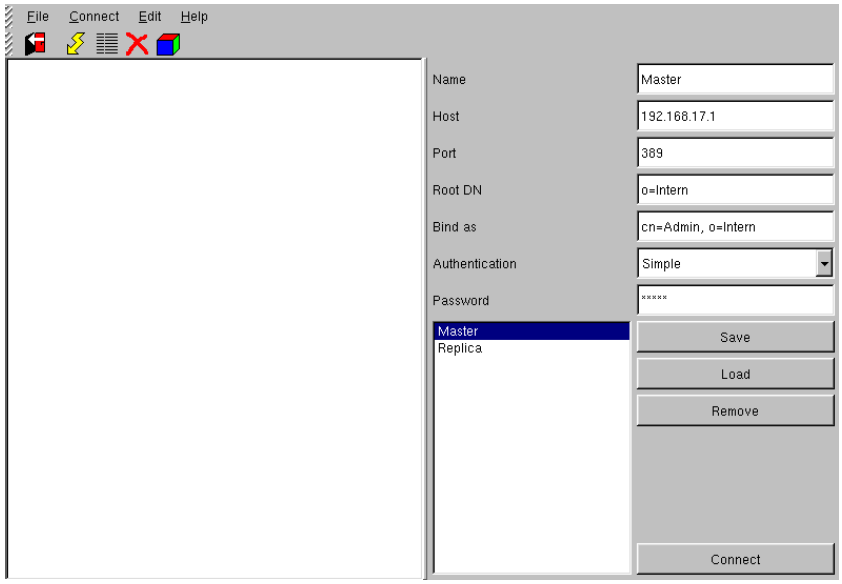

Abbildung 4.12: `kldap`: Verbindung zum LDAP-Server

Da es mit dem Bildschirmdialog möglich ist, die Einstellungen der LDAP-Server abzuspeichern, wird zunächst nach einem beliebigen Namen gefragt (`Name`), der die Server-Definition repräsentiert. Der DNS-Name oder die IP-Adresse des Servers sind anschließend im Feld `Host` anzugeben. Die Eingabe des TCP-Ports erfolgt in `Port`.

Hauptmenü	Menüpunkt	Beschreibung
File	Exit	Programmende
Connect	Connect	Verbinden zum LDAP-Server
Edit	Edit attributes...	Attribute verändern
Edit	Delete	Objekt löschen
Edit	Add...	Objekt hinzufügen
Edit	Search	Objekte suchen
Help	Contents	Hilfe zum Programm
Help	About kldap	Name des Autors von kldap
Help	About KDE	Namen der Autoren von KDE

Tabelle 4.28: Menüpunkte des Programms `kldap`

Als Nächstes wird der Context definiert, der als Wurzel in der Anzeige von `kldap` dienen soll (`Root DN`). Er wird als Distinguished Name des entsprechenden Objekts aufgeführt. Falls die ganze Baumstruktur angezeigt werden soll, ist an dieser Stelle der oberste Container anzugeben. Anschließend muss `kldap` mitgeteilt werden, mit welchen Werten die Beglaubigung am LDAP-Server durchzuführen ist. Dazu werden der DN des Objekts (`Bind as`), die Art der Beglaubigung (`Authentication`) und das zugehörige Kennwort (`Password`) angegeben. Für `Authentication` kann zurzeit lediglich `Simple` ausgewählt werden.

Die so eingegebenen Werte könnten nun unter dem definierten Namen abgespeichert werden (Save). Dadurch können sie bei späteren Sitzungen mittels Load wieder geladen werden. Gespeicherte Konfigurationen können mit Remove wieder gelöscht werden. Um die Verbindung mit den eingegebenen oder ausgewählten Werten herzustellen, ist der Button Connect anzuwählen. War die Anmeldung am LDAP-Server erfolgreich, so ist in der linken Bildschirmhälfte das im Feld `Root DN` angegebene Objekt sichtbar.

Auf der linken Bildschirmhälfte kann sich der Anwender nun mit der Maus im Verzeichnisdienst bewegen (siehe Abbildung 4.13).

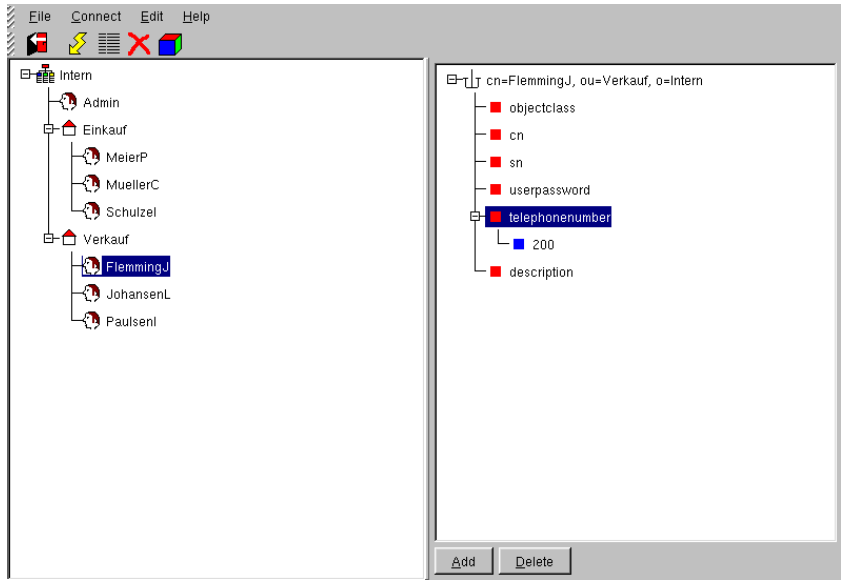

Abbildung 4.13: `kldap`: Objekte und Attribute

Mit einem Doppelklick auf ein Containerobjekt öffnet man dieses. Dabei werden die darunter liegenden Objekte sichtbar, die ihrerseits wieder mit einem Doppelklick geöffnet werden können. Um einen Container wieder zu schließen, ist die linke Maustaste ebenfalls doppelt zu drücken. Falls man die Attribute zu einem Objekt einsehen

will, markiert man den gewünschten Eintrag zunächst mit der Maus. Anschließend wählt man den Menüpunkt EDIT ATTRIBUTES... aus oder klickt das entsprechende Icon an. Die Attribute werden dann in der rechten Bildschirmhälfte angezeigt. Der Name des Attributs wird jeweils hinter einem roten Kasten aufgeführt. Ein Doppelklick darauf zeigt die entsprechenden Werte hinter einem blauen Kasten an. Auf diese Weise ist es nun möglich, alle Attribute einzusehen.

Wird in der rechten Bildschirmhälfte nun ein Attribut markiert, so kann ein Attributwert mit `Add` hinzugefügt werden. Das Entfernen des markierten Attributs erfolgt mit `Delete`. Falls im rechten Bildschirmabschnitt das Objekt an sich markiert wird, so kann über `Add` eine neue Eigenschaft erstellt werden (siehe Abbildung 4.14).

Abbildung 4.14: `kldap`: Erstellen eines Attributs

Damit dies gelingt, muss am LDAP-Server entweder der Parameter `schemacheck` auf `off` gestellt sein, oder in der Objektklasse `person` wird zusätzlich unter `allows` die Eigenschaft `mail` eingetragen. Anderenfalls wäre das Hinzufügen des Attributs aufgrund einer Schemaverletzung nicht möglich:

```
objectclass person
        requires
                objectClass,
                sn,
                cn
        allows
                description,
                seeAlso,
                telephoneNumber,
                userPassword,
                mail
```

Für das so erstellte Attribut kann dann ein Wert angegeben werden (siehe Abbildung 4.15).

Mit dem Programm `kldap` ist es ferner möglich, neue Objekte im Verzeichnisbaum zu erstellen. Dazu ist zunächst der Container zu markieren, unter dem das Objekt angelegt werden soll. Anschließend ist EDIT — ADD oder das zugehörige Symbol anzuwählen (siehe Abbildung 4.16).

Neben dem gewünschten Namen (`First DN Element`) kann die Objektklasse (`Object Class`) ausgewählt werden. Falls ein Objekt einer dort nicht aufgeführten

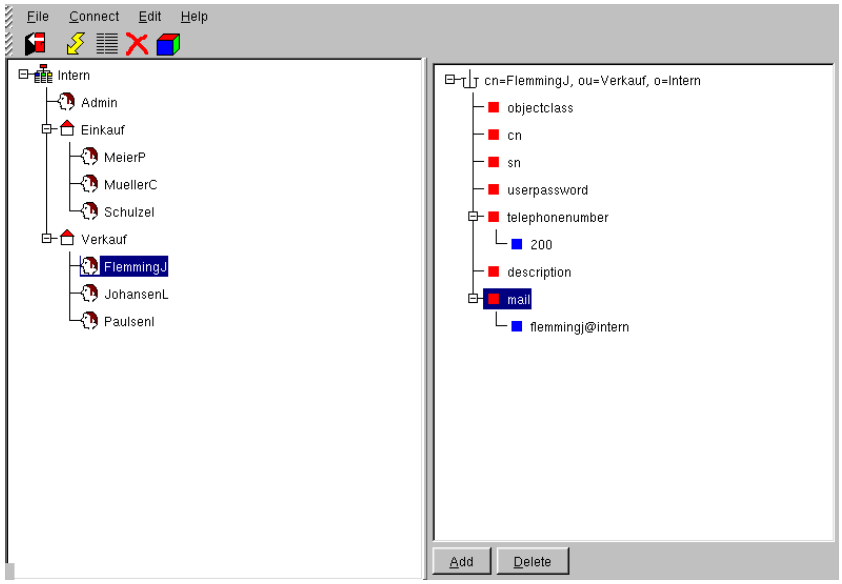

Abbildung 4.15: `kldap`: Erstellen eines Attributs — Wert einfügen

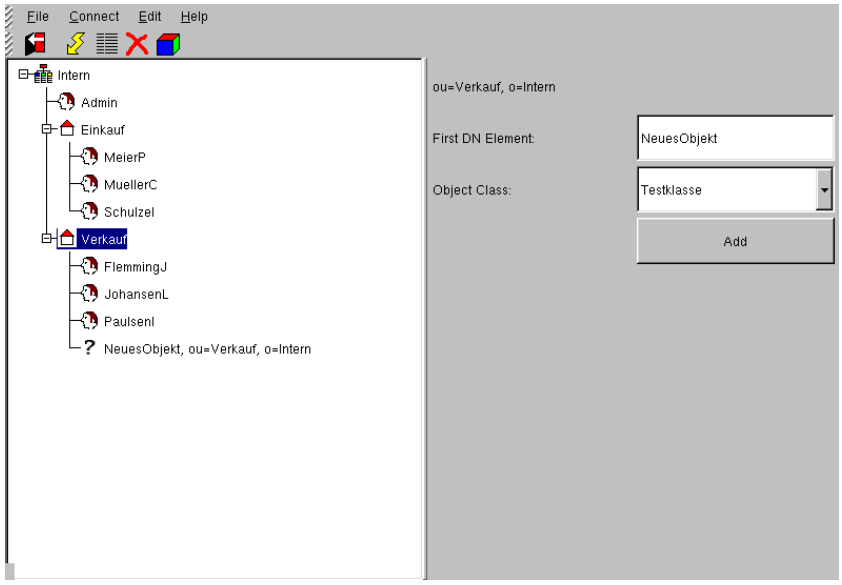

Abbildung 4.16: `kldap`: Erstellen eines Objekts

Klasse erstellt werden soll, so kann die Auswahl auch überschrieben werden. Beachten Sie, dass bei einer Objekterstellung die notwendigen Attribute ebenfalls definiert werden müssen (nur bei `schemacheck on`).

Mit dem Programm `kldap` ist dies jedoch zurzeit noch nicht möglich. `kldap` versucht, die angezeigten Objekte in der linken Bildschirmhälfte mit einem aussagekräftigen Symbol zu versehen. Falls dies nicht möglich ist, so wird dort lediglich ein Fragezeichen angegeben.

Für neu erstellte Objekte können anschließend die gewünschten Eigenschaften definiert werden. Der Name des Objekts entspricht bei Blattobjekten dem Attribut `cn`.

Um ein Objekt aus dem Verzeichnisdienst zu löschen, ist es zunächst zu markieren. Mit Edit — Delete kann es dann entfernt werden.

Das Programm `kldap` ist ferner in der Lage, Suchfunktionen am Verzeichnisdienst auszuführen. Dazu ist zunächst der Container mit der Maus zu markieren, der als Basis für die Suche verwendet werden soll. Anschließend wird mit der Anwahl des Menüpunkts Edit — Search ein separater Bildschirmdialog angezeigt (siehe Abbildung 4.17). In ihm wird das Suchkriterium eingegeben. Es muss dabei der Syntax genügen, die bereits im Zusammenhang mit dem Kommando `ldapsearch` in Abschnitt 4.6.6 auf Seite 106 beschrieben wurde. So liefert eine Suche nach

```
(&(objectclass=person)(!(cn=Admin)))
```

alle Objekte der Klasse `person` mit Ausnahme der Einträge, die als `Common Name` den Wert `Admin` aufweisen.

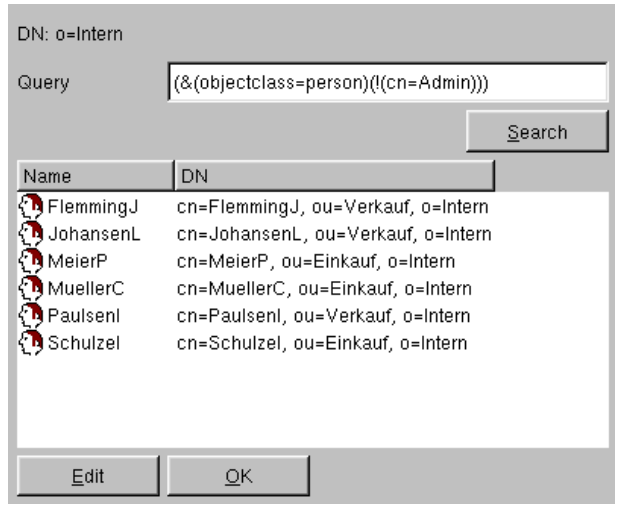

Abbildung 4.17: `kldap`: Suchen nach Objekten

Die so aufgeführten Ergebnisse können anschließend direkt bearbeitet werden. Durch das Anklicken von `Edit` wird die bereits bekannte Attributliste angezeigt und kann auch entsprechend verändert werden.

Zusammenfassend betrachtet, bietet das Programm `kldap` dem Administrator des Verzeichnisdienstes sowie allen Anwendern eine zentrale und leicht zu bedienende Möglichkeit, auf die gewünschten Informationen zuzugreifen und sie zu verändern.

In den Beispielanwendungen in Kapitel 5 wird `kldap` ebenfalls zur Visualisierung des Directory Information Tree verwendet.

4.6.15 Der Browser `netscape`

Die bisher vorgestellten Zugriffsmöglichkeiten basieren auf dem Betriebssystem Linux. Um die Daten der Verzeichnisdienste jedoch einer breiten Öffentlichkeit zur Verfügung stellen zu können, ist es unumgänglich, ein plattformübergreifendes Programm zu haben, mit dem die Daten eingesehen werden können.

Die Nutzung des Internets ist heutzutage von nahezu jeder beliebigen Plattform aus möglich. Durch Internet-Browser wird den Anwendern der Zugang zum Internet ermöglicht. Neben der Anzeige von HTML-Seiten bieten Browser ferner auch die Möglichkeit, auf die Daten eines LDAP-Verzeichnisdienstes zuzugreifen. Im Folgenden wird dieser Zugriff anhand des sehr verbreiteten Programms `netscape` beschrieben. Es wird unter Linux durch die Eingabe von

```
user1@host3:~ # netscape &
user1@host3:~ #
```

gestartet und bietet die Möglichkeit, das gewünschte Objekt des Directory Service direkt als Uniform Resource Locator URL anzugeben. Der Name der gesamten URL setzt sich dabei aus mehreren Teilen zusammen:

1. Die URL beginnt grundsätzlich mit dem Namen des Protokolls, das für den Zugriff auf die angegebene Adresse verwendet werden soll. Ihm folgen die Zeichen `://`. Für Internet-Anfragen handelt es sich demnach um `http://`, für LDAP-Anfragen um `ldap://`.
2. Anschließend erfolgt die Angabe des Servers, an den die Anfrage gesendet werden soll. Er kann mit seinem Namen oder seiner IP-Adresse angegeben werden (zum Beispiel `host1.intern`).
3. Nach einem weiteren Trennzeichen / wird nun der Distinguished Name des Objekts angegeben, auf das zugegriffen werden soll. Ein Beispiel wäre `o=Intern`.

Die URL für LDAP-Anfragen hat also allgemein die Form:

```
ldap://<Server>/<Distinguished Name>
```

Auf diese Weise ist es möglich, mit

```
ldap://host1.intern/o=Intern
```

auf das Objekt `Intern` des auf dem Server `host1.intern` installierten Verzeichnisdienstes zuzugreifen (siehe Abbildung 4.18).

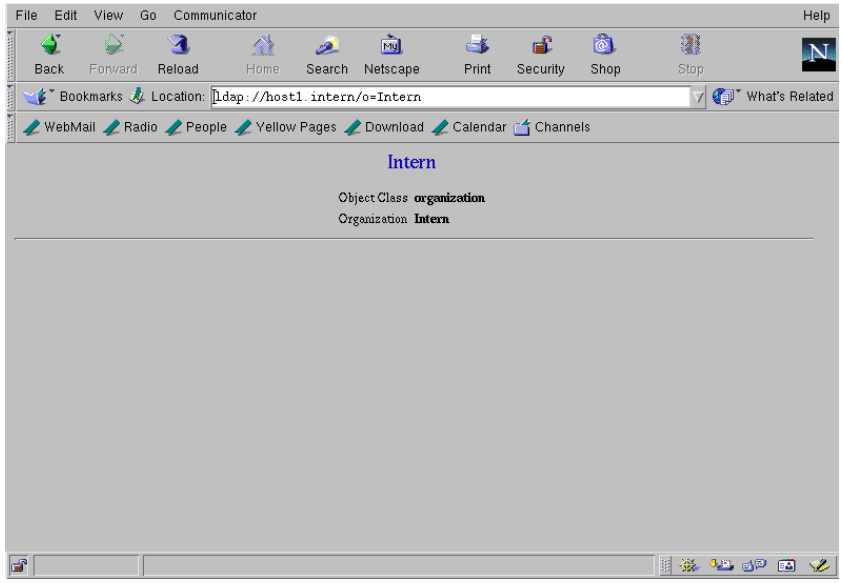

Abbildung 4.18: `netscape`: URL einer Organisation

Auf diese Weise können alle Objekte in Form einer URL im Programm `netscape` angesprochen werden. Der Zugriff auf das Benutzerobjekt `Flemmingj` erfolgt durch Angabe von

```
ldap://host1.intern/cn=Flemmingj, ou=Verkauf, o=Intern
```

im Uniform Resource Locator. Da es nicht möglich ist zu definieren, wie die Beglaubigung am DS-Tree erfolgen soll, erfolgt der Zugriff mit den definierten Standardrechten. Sie besagen in diesem Buch, dass grundsätzlich der lesende Zugriff erlaubt ist. Das Objekt `Admin` sowie alle `password`-Attribute bleiben jedoch im Verborgenen. Auf sie kann nur mit einer zuvor erfolgten Anmeldung zugegriffen werden. Diese Art des Zugriffs wird auch als Gastzugang bezeichnet. Im Beispiel ist daher das Kennwort des Anwenders `FlemmingJ` nicht sichtbar. Die Eigenschaft der E-Mail-Adresse wird im Browser so angezeigt, dass es möglich ist, durch Anklicken direkt das konfigurierte Mail-Programm zu starten und die aufgeführte Adresse bereits als Empfänger einzutragen (siehe Abbildung 4.19).

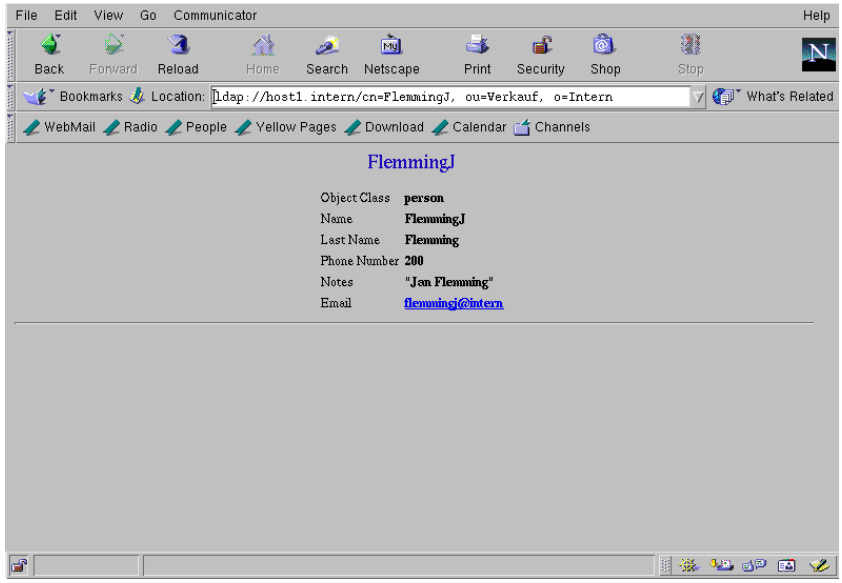

Abbildung 4.19: `netscape`: URL eines Benutzer

Auf die oben angegebene Weise kann die LDAP-Datenbank als Adressbuch benutzt werden. Zu jedem Benutzer wird im Beispiel festgehalten, welche Telefonnummer und welche Mail-Adresse er besitzt. Das Programm `netscape` bietet jedoch auch auf einem anderen Weg die Möglichkeit, ein Adressbuch zu pflegen. Es wird über den Menüpunkt ADDRESS BOOK unter COMMUNICATOR aufgerufen. Das Adressbuch (siehe Abbildung 4.20) ist in der Lage, die Informationen eines Verzeichnisdienstes zu nutzen. Dabei ist es nicht wie bisher notwendig, den Namen des Benutzerobjekts zu kennen, nach dem gefragt werden soll. Ferner werden automatisch alle Benutzerobjekte der Datenbank des entfernten LDAP-Servers angezeigt:

Nachdem in das Adressbuch verzweigt wurde, muss der Directory Service zunächst einmalig aufgenommen werden. Nach der Auswahl von FILE — NEW DIRECTORY erscheint die Eingabemaske aus Abbildung 4.21.

Neben einem Namen (`description`) werden der Server sowie dessen Wurzel angegeben. Der TCP-Port ist hinter `Port Number` einzugeben. Die maximale Anzahl der Treffer bei einer späteren Suche wird mit `Maximal Number of Hits` definiert. Standardmäßig erfolgt der Zugriff mit Gastrechten. Es ist jedoch auch möglich, eine Anmeldung auf Passwortebene zu verlangen. Das Kennwort entspricht dabei dem Attribut `userpassword`, der Anmeldename dem Attribut `mail`.

Nachdem der LDAP-Server im Adressbuch von `netscape` eingerichtet wurde, ist es nun möglich, auf die LDAP-Daten zuzugreifen. Dazu ist zum einen der neue

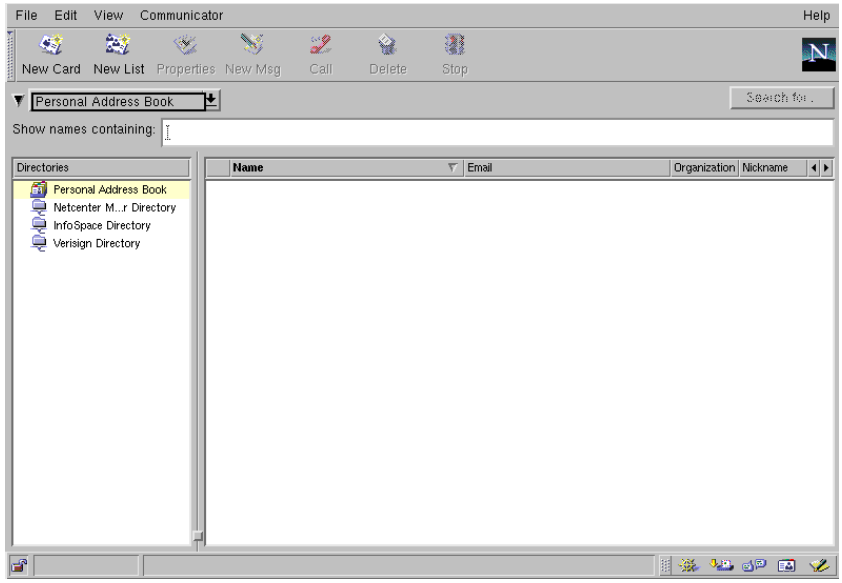

Abbildung 4.20: `netscape`: Adressbuch

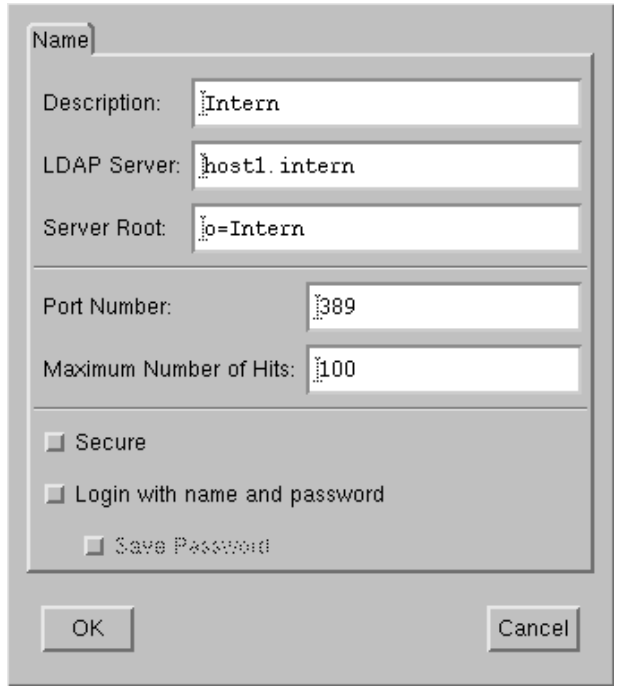

Abbildung 4.21: `netscape`: Adressbuch konfigurieren

Verzeichniseintrag mit dem Namen `Intern` anzuwählen. Zum anderen muss hinter `Show names containing` (siehe Abbildung 4.22) ein Suchbegriff eingetragen werden. Erst danach werden alle Benutzerobjekte des Directory Service angezeigt, die den entsprechenden Suchbegriff enthalten. Eine Anfrage nach `*` zeigt alle Anwender des DIT an. Es werden unter anderem die Attribute `cn`, `telephonenumber` und `mail` angezeigt.

Mit diesem Verfahren ist es möglich, sehr schnell die gewünschten Informationen eines Objekts einzusehen. Es ist jedoch auch möglich, das Suchkriterium sehr ausführlich anzugeben. Dazu muss man im oberen rechten Bildschirmbereich den Button SEARCH FOR... anwählen. In ihm kann eine ADVANCED SEARCH durchgeführt werden. Dabei können mehrere Kriterien miteinander kombiniert werden. Die Daten sind relativ intuitiv einzugeben und müssen an dieser Stelle nicht näher erläutert werden. Um eine zusätzliche Zeile in die Kombination mit aufzunehmen, ist MORE anzuwählen, FEWER entfernt eine Zeile.

In Abbildung 4.22 ist zunächst eine allgemeine Suche nach allen Einträgen dargestellt. In Abbildung 4.23 ist der Bildschirmdialog für eine erweiterte Suche nach Objekten abgedruckt, deren E-Mail-Adresse die Zeichenkette `intern` enthält und deren Telefonnummer mit der Zahl 2 beginnt.

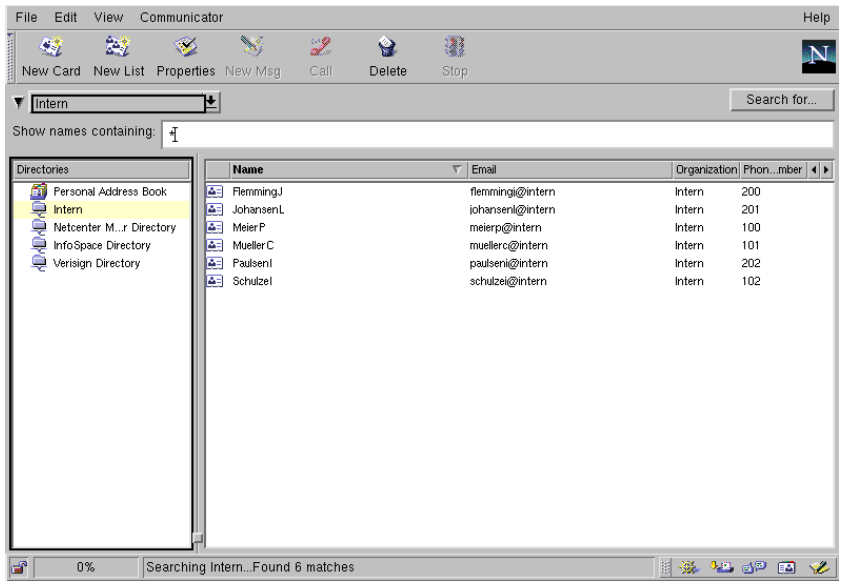

Abbildung 4.22: `netscape`: LDAP-Daten im Adressbuch

Die Ergebnisse einer Suche werden wie in Abbildung 4.22 dargestellt. Jeder Eintrag der Ergebnisliste kann nun auch separat betrachtet werden. Dazu muss man lediglich

mit der Maus doppelklicken auf das gewünschte Objekt. Anschließend öffnet sich zum Beispiel das Fenster aus Abbildung 4.24.

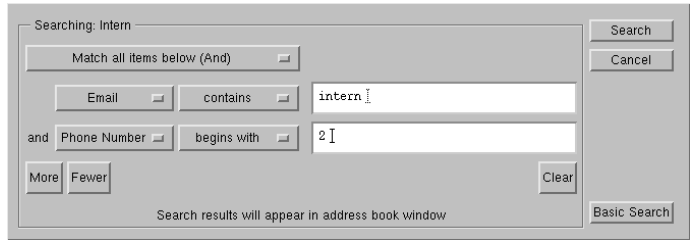

Abbildung 4.23: `netscape`: Erweiterte Suche

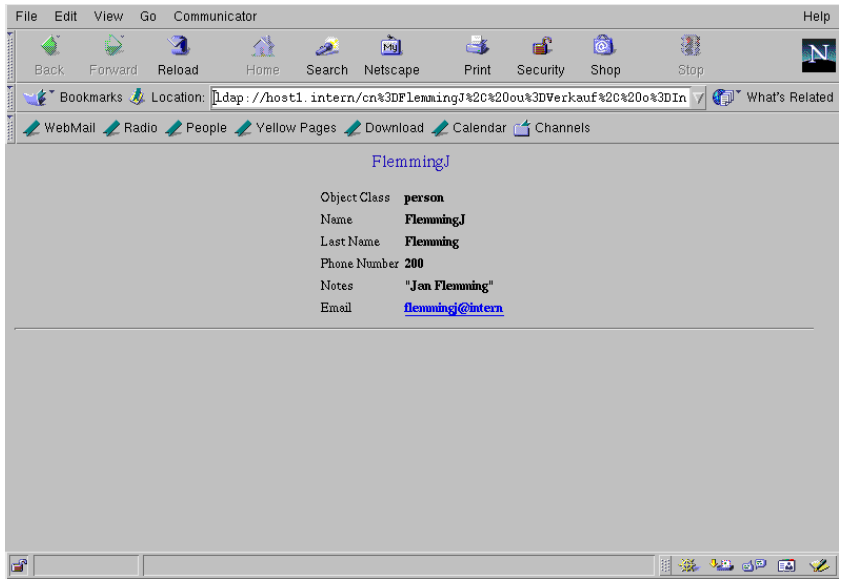

Abbildung 4.24: `netscape`: Ergebnis detailliert betrachtet

Die URL entspricht der bereits bekannten Form. Es sind lediglich einige Zeichen durch deren hexadezimalen Code ersetzt (vergleiche Tabelle 4.29).

Code	Zeichen
%3D	=
%2C	,
%20	Leerzeichen

Tabelle 4.29: `netscape`: Hexadezimale Zeichen

4.7 Die LDAP-Gateways

Die Funktionalität des LDAP-Servers sowie des -Clients wurden in den vorangegangenen Abschnitten detailliert beschrieben. In den folgenden Abschnitten wird die Funktionsweise der LDAP-Gateways erläutert.

4.7.1 Funktion

Bevor einige Gateways genauer vorgestellt werden können, soll zunächst ihre allgemeine Funktion erläutert werden.

Bisher wurde eine Reihe von Kommandos, Programmen und grafischen Anwendungen beschrieben, die für den Zugriff auf die Daten eines Verzeichnisdienstes konzipiert sind. Um sie zu benutzen, ist es jedoch grundsätzlich notwendig, sich mit ihren Gegebenheiten und der Bedienung vertraut zu machen. Um die Akzeptanz der Nutzung von Directory Services noch weiter zu erhöhen, existieren die so genannten Gateways. Sie zeichnen sich dadurch aus, dass der Anwender keine neuen Funktionalitäten und keine Bedienungshinweise zu erlernen braucht. Dies rührt daher, dass er bereits bestehende, ihm bekannte Anwendungen und Kommandos nutzen kann, um mit ihnen den LDAP-Zugriff zu realisieren. Dabei erfolgt die Kommunikation in den folgenden Schritten:

1. Zu Beginn sendet der Client eine Anfrage mit einer bekannten Anwendung an das Gateway.
2. Das Gateway analysiert den Client-Wunsch und wandelt ihn in eine LDAP-Anfrage um, die dann zum LDAP-Server gesendet wird.
3. Der Server versucht nun, die Anfrage zu beantworten, und sendet das Ergebnis dem Gateway zu.
4. Das Gateway wiederum leitet das Resultat an den Client weiter.

Tabelle 4.30 zeigt den Ablauf der Kommunikation in schematisierter Form.

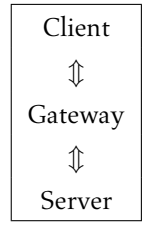

Tabelle 4.30: Funktionsweise der LDAP-Gateways

Die Kommunikation zwischen dem Gateway und dem Server basiert auf dem Lightweight Directory Access Protocol (LDAP). Die Verbindung zwischen Client und Gateway wird mit einem anderen Protokoll realisiert (siehe Tabelle 4.31).

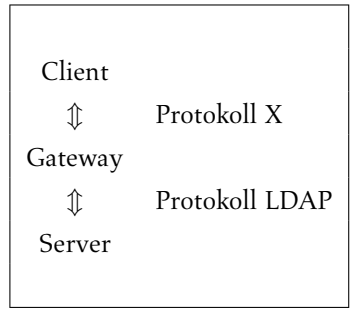

Tabelle 4.31: Kommunikation über LDAP-Gateways

Bei beiden Verbindungen handelt es sich in der Regel um Verbindungen über das Transmission Control Protocol (TCP) bzw. das User Datagram Protocol (UDP). Diese Protokolle zeichnen sich dadurch aus, dass sie in Ports unterteilt sind. Jedem Port ist eine Anwendung zugewiesen.

HTTP-Anfragen werden zum Beispiel über den Port 80 abgehandelt, LDAP-Anfragen über 389 (siehe Tabelle 4.32).

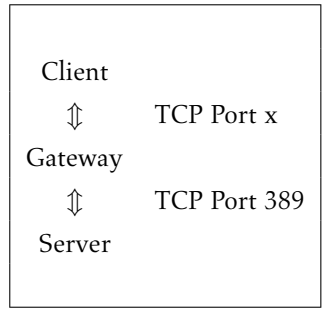

Tabelle 4.32: TCP-Kommunikation über LDAP-Gateways

Die einzige Aufgabe des Gateways besteht darin, die Client-Anfragen in das Protokoll LDAP (TCP-Port 389) zu konvertieren. Gateways können deshalb auch als Protokoll-Konverter bezeichnet werden.

Bei den Komponenten

- LDAP-Gateway und
- LDAP-Server

kann es sich um zwei verschiedene Rechner im Netzwerk handeln. Es ist jedoch auch möglich, dass die beiden Komponenten auf ein und demselben Host installiert sind.

4.7.2 Das Finger-Gateway

Um Anfragen, die mit der bekannten Finger-Anwendung getätigt werden, an einen LDAP-Server zu übermitteln, kann das im Folgenden vorgestellte Gateway verwendet werden.

Allgemeines

Mit dem Kommando `finger` ist es möglich, Informationen über einen Benutzer zu erhalten, der an einer entfernten Maschine arbeitet. Dazu ist es notwendig, dass dort ein `finger`-Server aktiv ist, der Anfragen am TCP-Port 79 entgegen nimmt. Aus den Konfigurationsdateien des Servers können dem Client anschließend Informationen über einen Anwender mitgeteilt werden.

Das folgende Beispiel zeigt den Aufruf des Kommandos `finger` auf dem Rechner `host2`, um Informationen über einen Benutzer am `host1` zu erlangen (siehe auch Abbildung 4.25).

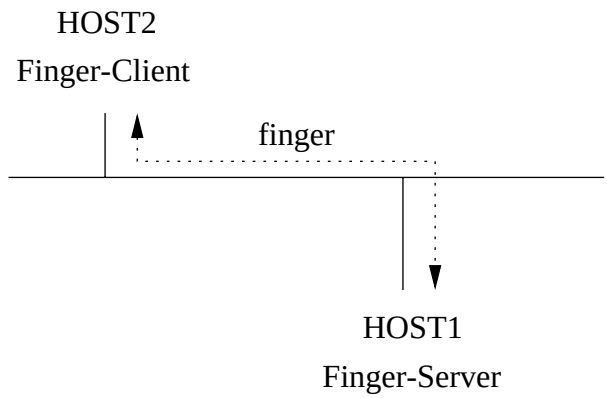

Abbildung 4.25: Das Protokoll `finger`

Beispiel 4.50: Aufruf des Kommandos `finger`

```
user1@host2:~ # finger jens@host1
[host1/192.168.17.1]

Welcome to Linux version 2.2.16 at host1.intern !

 9:06am up 1:57, 3 users, load average: 0.00, 0.08, 0.08
```

```
Login: jens                              Name: Jens Banning
Directory: /home/jens                    Shell: /bin/bash
On since Tue Nov 21 07:09 (MET) on tty1
On since Tue Nov 21 07:09 (MET) on tty2, idle 1:56
Mail last read Tue Sep 26 14:21 2000 (MEST)
No Plan.
user1@host2:~ #
```

Entsprechend erfolgt der allgemeine Aufruf in der Form:

```
finger <Benutzername>@<Hostname>
```

Im weiteren Verlauf der Ausführungen zum Finger-Gateway werden der Server und das Gateway auf dem Rechner `host1` installiert (siehe Abbildung 4.26).

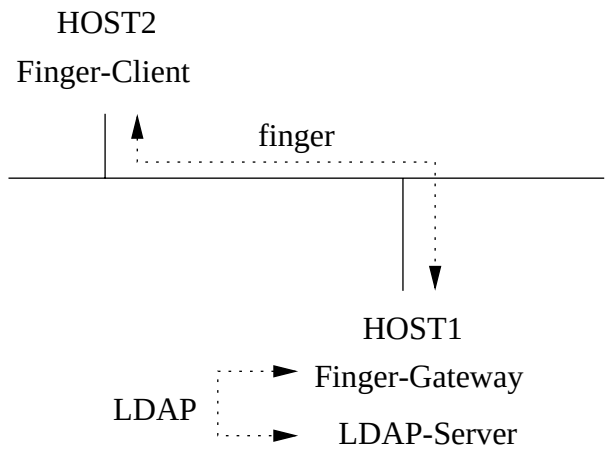

Abbildung 4.26: Finger-Gateway

Ein Finger-Gateway repräsentiert

- einen Finger-Server, der Anfragen von Clients entgegennimmt, und
- einen LDAP-Client, der die Anfragen zum LDAP-Server weiterleitet.

Es hat also nur die Funktionalität, die eingehenden Finger-Daten in das Lightweight Directory Access Protocol umzuwandeln. Daher können Gateways auch als Protokoll-Konverter bezeichnet werden.

Der Dämon `in.xfingerd`

Um ein Finger-Gateway einzurichten, ist es notwendig, auf dem gewünschten Rechner einen Dämonprozess zu starten, der die Umwandlung der Finger-Anfragen in das Protokoll LDAP durchführt. Dieser Dämon trägt den Namen `in.xfingerd`. Er

wird wie der Original-Finger-Server auch unter Linux über den Inet-Dämon gestartet. `inetd` ist standardmäßig auf jedem Linux-Rechner installiert. Er ist in der Lage, Server-Dienste erst dann zu aktivieren, wenn Anfragen für den Dienst eintreffen. So ist das Finger-Gateway nicht permanent aktiv, sondern erst in dem Moment, wenn es von einem Client angesprochen wird.

Bevor die Konfiguration des `inetd` mit der Datei `inetd.conf` beschrieben wird, muss zunächst angegeben werden, welche Optionen beim Aufruf von

```
in.xfingerd [Optionen]
```

verwendet werden können (siehe Tabelle 4.33).

Option	Beschreibung
`-x <Host>`	LDAP-Server
`-p <Port>`	Server-Port
`-i`	Interaktiver Modus
`-l`	Logging unterdrücken
`-f <Datei>`	Filterdatei
`-t <Datei>`	Template-Datei
`-c <Anzahl>`	Anzahl der DN-Komponenten

Tabelle 4.33: Optionen des LDAP-Gateways `in.xfingerd`

Mit dem Parameter `-x` besteht die Möglichkeit, dem Dämon `in.xfingerd` die Adresse des LDAP-Servers mitzuteilen. Sie kann als Name oder als IP-Nummer angegeben werden. Fehlt die Definition, so wird der in der Datei `ldap.conf` definierte Server verwendet. Der TCP-Port, an dem der LDAP-Server auf Anfragen wartet, kann über die Option `-p` definiert werden. Anderenfalls wird er ebenfalls aus der Datei `ldap.conf` im Verzeichnis `/etc/openldap/` gelesen.

Falls `in.xfingerd` direkt aus dem Verzeichnis `/usr/lib/openldap/` ohne Verwendung des `inetd` gestartet wird, so muss zusätzlich der Parameter `-i` benutzt werden. Damit wird das Gateway angeregt, im interaktiven Modus zu arbeiten. Während das Finger-Gateway aktiv ist, protokolliert es diverse Status- und Fehlermeldungen. Sie werden dem `syslogd` über `LOCAL4` übergeben. Er schreibt die Meldungen dann, sofern er nicht anders konfiguriert ist, in die Datei `/var/log/messages`. Diese Art der Protokollierung wird mit der Option `-l` unterdrückt. Näheres zum `syslog`-Dämon und dessen Konfiguration ist in [4] nachzulesen.

Das Finger-Gateway verwendet als Standardeinstellung die Filterdatei `ldapfilter.conf` aus dem Verzeichnis `/etc/openldap/`. Template-Informationen werden aus der Datei `ldaptemplates.conf` des gleichen Verzeichnisses gewonnen. Falls diese

beiden Vorgaben überschrieben, also andere Dateien verwendet werden sollen, muss deren Pfad dem Dämon über die Optionen `-f` und `-t` mitgeteilt werden.

Falls ein Finger-Client Anfragen an das Gateway sendet, kann die Anzahl der Komponenten des Distinguished Name, die im Ergebnis aufgeführt werden, über den Parameter `-c` vorgegeben werden. Ein DN der Form `cn=FlemmingJ, ou=Verkauf, o=Intern` würde bei einer Einstellung von `-c 2` lediglich als `cn=FlemmingJ, ou=Verkauf` aufgeführt.

Beispiel 4.51: Aufruf von `in.xfingerd` mit der Option `-i`

```
root@host1:~ # /usr/lib/openldap/in.xfingerd -i
OpenLDAP Finger Service …
```

Üblicherweise wird der Dämonprozess nicht permanent gestartet, sondern der `inetd` wird so konfiguriert, dass das Gateway nur bei vorliegenden Anfragen aktiviert wird. Der `inetd` ist ein Dienst, der in der Lage ist, alle TCP- und UCP-Ports zu überwachen. Sobald auf einem Port eine Aktivität auftritt, kann ein anderer Dienst gestartet werden. Bei Nichtaktivität wird der so gestartete Dienst wieder beendet. Die Konfiguration von `inetd` erfolgt über die Datei `inetd.conf`, die sich im Verzeichnis `/etc/` befindet. Bei ihr handelt es sich um ein ASCII-basiertes Dokument, in dem Kommentarzeilen mit dem Zeichen # eingeleitet werden.

Falls dort zuvor bereits ein Finger-Server erwähnt war, so muss er nun entsprechend auszukommentiert werden. Anschließend ist eine Zeile einzutragen, die das Gateway beschreibt. Die drei Punkte in der folgenden Ausgabe deuten an, dass in ihr nicht alle Zeilen der Datei wiedergegeben sind. Ferner sind die Angaben aus layouttechnischen Gründen auf zwei Zeilen aufgeteilt:

```
root@host1:~ # cat /etc/inetd.conf
…
#finger stream tcp nowait nobody
                  /usr/sbin/tcpd                in.fingerd -w
finger  stream tcp nowait nobody
                  /usr/lib/openldap/in.xfingerd in.xfingerd
…
root@host1:~ #
```

Die ursprüngliche Finger-Definition wurde mit dem Zeichen # deaktiviert, und anschließend wurde die neue Konfiguration eingetragen. Sie besagt, dass bei Finger-Anfragen (`finger`) über das Datenstromprotokoll (`stream`) TCP (`tcp`) sofort (`nowait`) mit den Rechten des Benutzers `nobody` der Dämon `/usr/lib/openldap/in.xfingerd` gestartet wird. Ob und wann welche Optionen zu verwenden sind, wird in der letzten Spalte der Eintragung definiert. Im oben gezeigten Beispiel werden demnach keine Parameter verwendet.

Abschließend ist es notwendig, den `inetd` dazu zu veranlassen, seine Konfigurationsdatei neu einzulesen. Erst dann werden die Änderungen auch wirksam. Dies geschieht bei einer SuSE-Distribution durch Aufruf des zugehörigen Skripts.

```
root@host1:~ # /etc/rc.d/init.d/inetd reload
Reload INET services (inetd)                                done
root@host1:~ #
```

Der Zugriff vom Client

Nachdem das Finger-Gateway nun konfiguriert wurde, können vom Client aus mit dem Kommando `finger` LDAP-Anfragen in Auftrag gegeben werden. Sie werden über das Gateway vom Server beantwortet. Die allgemeine Form, in der das Kommando `finger` aufzurufen ist, sieht so aus:

```
finger <Benutzername>@<Hostname>
```

oder so:

```
finger "<Suchkriterium>"@<Hostname>
```

Um Informationen über ein bestimmtes Benutzerobjekt des Verzeichnisdienstes zu erlangen, muss man dem Kommando lediglich den Namen des Objekts sowie den Hostnamen des Finger-Gateways mitteilen.

Beispiel 4.52: Aufruf des Kommandos `finger`

```
user1@host2:~ # finger FlemmingJ@host1
[host1/192.168.17.1]
OpenLDAP Finger Service…
1 exact match found for "FlemmingJ":
"FlemmingJ, Verkauf"
 Also Known As:
               FlemmingJ
 Work Phone:
               200
 E-Mail Address:
               flemmingj@intern
 Description:
               "Jan Flemming"
user1@host2:~ #
```

Es ist jedoch auch möglich, per `finger` ein komplettes Suchkriterium anzugeben. Die Syntax, nach der es zusammengestellt werden kann, entspricht der des Kommandos `ldapsearch` (siehe dazu Abschnitt 4.6.6 auf Seite 106). Um zum Beispiel alle Objekte anzuzeigen, deren Common Name mit den Zeichenketten `Flem` oder `Paul` beginnt, ist das folgende Filterkriterium zu verwenden:

```
(|(cn=Flem*)(cn=Paul*))
```

Falls die Suche exakt ein Ergebnis liefert, so erfolgt eine detaillierte Anzeige des gefundenen Objekts. Bei mehr als einem Ergebnis werden die Objekte lediglich mit ihrem Namen aufgeführt.

Beispiel 4.53: Aufruf des Kommandos `finger`

```
user1@host2:~ # finger "(|(cn=Flem*)(cn=Paul*))"@host1
[host1/192.168.17.1]
OpenLDAP Finger Service…
2 arbitrary filter matches for "(|(cn=Flem*)(cn=Paul*))":
  FlemmingJ
  PaulsenI
user1@host2:~ #
```

Falls die Suche zu einem Treffer führt, werden einige Attribute des Objekts ausgegeben. Um welche Eigenschaften es sich dabei handelt, kann über die Datei `ldaptemplates.conf` aus dem Verzeichnis `/etc/openldap/` des Gateway-Rechners bestimmt werden. Das Format und die Bedeutung der Dateieinträge wurden bereits in Abschnitt 4.6.3 auf Seite 101 beschrieben. In der Datei sind Templates (Schablonen) vorhanden, mit denen zu einem Objekt festgelegt wird, welche Attribute bei einer Anfrage ausgegeben werden sollen und mit welchen umgangssprachlichen Wörtern sie anzugeben sind. In der Konfigurationsdatei der Templates können beliebig viele Schablonen definiert werden. Sie werden in Bereichen aufgeführt, die jeweils einer Objektklasse oder mehreren Klassen entsprechen. So ist ein Template vorhanden, das Definitionen für die Objektklasse `person` beschreibt. Es wird durch einen Kommentar eingeleitet. In der Definition wird die Objektklasse angegeben, und die anzuzeigenden Attribute werden genannt. Im Folgenden werden die gerade genannten Abschnitte der Vorlage so angezeigt, wie sie per Standard definiert sind:

```
##########################################################################
# Person template
##########################################################################
…
# objectclass list
person
…
# list of items for display
item jpegbtn    "View Photo"            jpegPhoto
item audiobtn   "Play Sound"            audio
item cis,sort   "Also Known As"         cn
item cis,sort   "Title"                 title
item mls        "Work Address"          postalAddress
item cis        "Work Phone"            telephoneNumber
item cis        "Fax Number"            facsimileTelephoneNumber
```

```
item cis        "Pager Number"          pager
item mls        "Home Address"          homePostalAddress
item cis        "Home Phone"            homePhone
item cis        "User ID"               uid
item mail       "E-Mail Address"        mail
item cis        "Description"           description
item cis        "Favorite Beverage"     drink
item dn,sort    "See Also"              seeAlso
item time,ro    "Last Modified"         lastModifiedTime
item dn,ro      "Modified By"           lastModifiedBy
…
```

Erfolgt eine Finger-Anfrage an das Gateway auf ein Objekt der Klasse `person`, so werden die über `item` angegebenen Attribute ausgegeben. Hinter dem Wort `item` folgt eine Definition bezüglich der Syntax der im Folgenden genannten Eigenschaft. Welche Werte möglich sind, ist in der Datei `ldaptemplates.conf` in Form von Kommentaren festgehalten. Um eine Zeichenfolge zu beschreiben, bei der nicht zwischen Groß- und Kleinschreibung unterschieden wird, muss man hinter `item` den Wert `cis` aufführen. Als Nächstes wird ein umgangssprachlicher Begriff angegeben, mit dem die Eigenschaft beschrieben werden soll. In der letzten Spalte wird schließlich der Name des Attributs angegeben.

So bedeutet zum Beispiel die Zeile

```
item cis        "Work Phone"            telephoneNumber
```

dass die `telephoneNumber` in der Ausgabe mit dem Ausdruck `Work Phone` bezeichnet wird. Alle so über `item` definierten Attribute der im Template angegebenen Objektklasse werden bei einer Finger-Anfrage ausgegeben. Falls eine Eigenschaft im Verzeichnisdienst zu dem Objekt gar nicht existiert, so unterbleibt ihre Darstellung. Somit wird zum Beispiel die `Fax Number` nicht erwähnt.

Falls die Ausgabe der Finger-Angabe zusätzliche, bisher nicht erwähnte Eigenschaften liefern soll, können sie in der Datei `ldaptemplates.conf` ergänzt werden, z. B.:

```
item cis        "Surname"           sn
```

Beispiel 4.54: Aufruf des Kommandos `finger`

```
user1@host2:~ # finger FlemmingJ@host1
[host1/192.168.17.1]
OpenLDAP Finger Service…
1 exact match found for "FlemmingJ":
"FlemmingJ, Verkauf"
 Also Known As:
              FlemmingJ
 Work Phone:
              200
```

```
 Surname:
                Flemming
 E-Mail Address:
                flemmingj@intern
 Description:
                "Jan Flemming"
user1@host2:~ #
```

Bei der Datei `ldaptemplates.conf` handelt es sich um eine allgemeine Konfigurationsdatei des LDAP-Clients. Sie ist also nicht nur für das Finger-Gateway relevant, sondern auch für andere Gateways und Anwendungen.

4.7.3 Das Gopher-Gateway

Die LDAP-Software der OpenLDAP-Organisation liefert des Weiteren ein Gateway, mit dem es möglich ist, über die Anwendung Gopher eine Verbindung mit einem Verzeichnisdienst aufzunehmen.

Allgemeines

Bei der Gopher-Anwendung handelt es sich um die Möglichkeit, von einem Client aus auf Informationen des Servers zugreifen zu können. Damit allgemein eine Gopher-Verbindung im Netzwerk möglich ist, ist es notwendig, dass ein entsprechender `gopher`-Server gestartet ist.

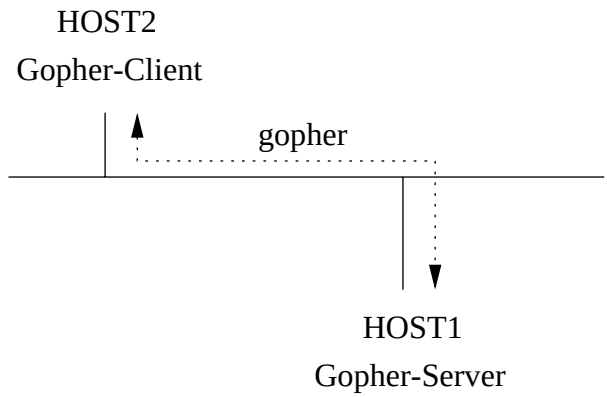

Abbildung 4.27: Das Protokoll `gopher`

Um mit dem Protokoll `gopher` LDAP-Anfragen generieren zu können, müssen auch diese zunächst einem Gopher-Gateway zugeführt werden. Das Gateway wandelt sie in das Protokoll LDAP um und gibt sie an den Server des Verzeichnisdienstes weiter.

Das Gopher-Gateway und der LDAP-Server können im Netzwerk auf unterschiedlichen Rechnern installiert sein. Im weiteren Verlauf der Ausführungen in diesem Buch werden das Gateway und der Server jedoch auf dem gleichen Rechner installiert. Er hat den Namen `host1`.

Entsprechend besteht das Gopher-Gateway aus

- einem Gopher-Server und
- einem LDAP-Client.

Es hat die Aufgabe, die Anfragen des Gopher-Protokolls in das Protokoll LDAP zu überführen.

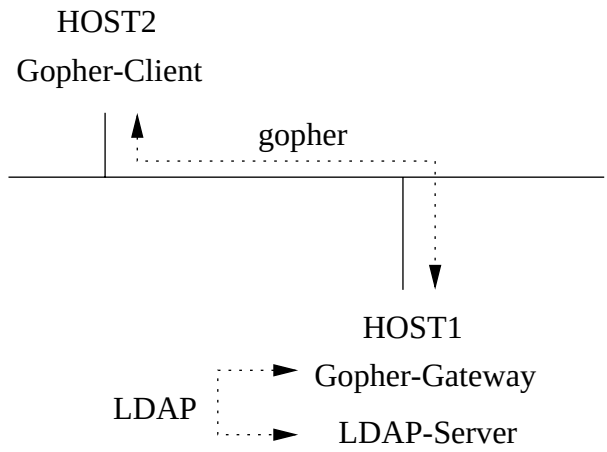

Abbildung 4.28: Gopher-Gateway

Der Dämon `go500`

Das Gopher-Gateway kann mit Hilfe von zwei Dämonprozessen gestartet werden. Der erste trägt den Namen `go500` und wird wie auch das Finger-Gateway über den Inet-Dämon gestartet.

`go500` kann mit einer Reihe von Optionen gestartet werden, die auch bei einer Konfiguration in der Datei `inetd.conf` verwendet werden können (siehe Tabelle 4.34).

```
go500 [Optionen]
```

Falls der LDAP-Server nicht in der Datei `ldap.conf` definiert ist, so muss seine Position im Netzwerk mit der Option `-x` angegeben werden. Ein alternativer TCP-Port, an dem das Gateway anzusprechen ist, kann dem Dämon über den Parameter `-p` mitgeteilt werden (Standard: `5555`).

Option	Beschreibung
`-x <Host>`	LDAP-Server
`-p <Port>`	go500-Port
`-I`	Gestartet vom Inet-Dämon
`-l`	Logging aktivieren
`-f <Datei>`	Filterdatei
`-t <Datei>`	Template-Datei
`-c <Anzahl>`	Anzahl der DN-Komponenten
`-d <Level>`	Debug-Level
`-b <Basis>`	Suchbasis

Tabelle 4.34: Optionen des LDAP-Gateways `go500`

Wenn `go500` über den Inet-Dämon gestartet wird, was in der Regel der Fall ist, so muss der Parameter `-I` verwendet werden. Das Logging von Meldungen und Aktivitäten kann über die Option `-l` eingeschaltet werden. Der `syslog`-Dämon bekommt die Informationen dann über `LOCAL3` zugestellt und schreibt sie per Default in die Datei `/var/log/messages`.

Mit den Parametern `-f` und `-t` ist es möglich, eine alternative Filter- bzw. Template-Datei anzugeben. Als Standardeinstellung werden die Dateien aus dem Verzeichnis `/etc/openldap/` verwendet.

Wenn das Gopher-Gateway dem Client Informationen über ein Objekt mitteilt, so wird dessen Distinguished Name angegeben. Es wird jedoch nur eine bestimmte Anzahl von Komponenten des DN aufgeführt. Bei der Verwendung von `-c 2` würde das Benutzerobjekt `Flemmingj` als `cn=FlemmingJ, ou=Verkauf` aufgeführt werden.

Außerdem existieren die Parameter `-d` und `-b`. Durch die Verwendung von `-d` kann ein Debug-Level definiert werden. `-b` gibt an, welche Basis das Gopher-Gateway benutzen soll, wenn es Anfragen an den Verzeichnisdienst weitergibt. Die Basis ist als Distinguished Name des entsprechenden Containerobjekts anzugeben.

Beispiel 4.55: Aufruf von `go500`

```
root@host1:~ # /usr/lib/openldap/go500
root@host1:~ #
```

Das Gopher-Gateway wird für gewöhnlich nicht direkt an der Linux-Eingabeaufforderung gestartet. Der Start des Dämons `go500` wird über den `inetd` nur dann angeregt, wenn tatsächlich Gopher-Anfragen eintreffen. Um dies zu realisieren, sind zwei Dateien zu bearbeiten.

Als Erstes muss der Datei `/etc/services` die folgende Zeile hinzugefügt werden:

```
go500      5555/tcp         # Go500-Gateway
```

Dieser Eintrag bedeutet, dass der TCP-Port 5555 dem System unter dem Namen `go500` bekannt gemacht wird. Falls bereits zuvor eine Eintragung in der Datei für den Port 5555 existierte, so kann sie entfernt werden.

Als Zweites erfolgt der Eintrag des Gateways in der Datei `/etc/inetd.conf` in der folgenden Form. Er ist komplett in einer Zeile einzugeben und hier nur aus Platzgründen aufgeteilt.

```
go500      stream tcp nowait nobody
                    /usr/lib/openldap/go500       go500 -I
```

Abschließend muss der `inetd` dazu veranlasst werden, seine Konfigurationsdatei neu einzulesen, damit die neue Eintragung aktiv wird:

```
root@host1:~ # /etc/rc.d/init.d/inetd reload
Reload INET services (inetd)                                done
root@host1:~ #
```

Der Dämon `go500gw`

Der Dämon `go500` realisiert ein Gopher-Gateway auf die Weise, dass durch Gopher alle per Gastzugang sichtbaren Einträge des Verzeichnisdienstes in der Gopher-Anwendung angezeigt werden.

Mit dem Dämon `go500gw` ist es ferner möglich, sich im gesamten Verzeichnisdienst zu bewegen und Suchanfragen zu starten. Der Dämon wird mit

```
go500gw [Optionen]
```

gestartet und befindet sich ebenfalls im Verzeichnis `/usr/lib/openldap/`. Tabelle 4.35 fasst die möglichen Optionen zusammen.

Ob beim Zugriff auf den Verzeichnisdienst Aliasobjekte verfolgt und angezeigt werden, kann mit dem Parameter `-a` definiert werden. Wird er angegeben, so erfolgt die Anzeige des über Alias angesprochenen Objekts, anderenfalls erfolgt sie nicht. An welchem Port das Gopher-Gateway anzusprechen ist, wird über `-p` (Default: `7777`) festgelegt. Mit der Option `-P` kann der Port des LDAP-Servers angegeben werden (`389`). Alle anderen Optionen haben die gleiche Bedeutung wie beim Dämon `go500`. Sie sind auf Seite 151 beschrieben.

Wie `go500` wird auch `go500gw` über den Inet-Dämon gestartet. Dazu muss zunächst über die Datei `/etc/services` definiert werden, unter welchem Namen der TCP-Port 7777, der für `go500gw` verwendet wird, anzusprechen ist. Dieses wird durch die Zeile

```
go500gw   7777/tcp        # Go500gw-Gateway
```

festgelegt. Ferner wird in der Datei `/etc/inetd.conf` der erst bei eintreffenden Anfragen zu startende Dämon `go500gw` in einer Zeile eingetragen:

```
go500gw   stream tcp nowait nobody
                    /usr/lib/openldap/go500gw    go500gw -I
```

Option	Beschreibung
`-x <Host>`	LDAP-Server
`-p <Port>`	go500gw-Port
`-P <Port>`	Port des LDAP-Servers
`-I`	Gestartet vom Inet-Dämon
`-l`	Logging aktivieren
`-f <Datei>`	Filterdatei
`-t <Datei>`	Template-Datei
`-c <Anzahl>`	Anzahl der DN-Komponenten
`-d <Level>`	Debug-Level
`-a`	Aliasobjekte verfolgen

Tabelle 4.35: Optionen des LDAP-Gateways `go500gw`

Nach der Neuinitialisierung von `inetd` mit

```
root@host1:~ # /etc/rc.d/init.d/inetd reload
Reload INET services (inetd)                                done
root@host1:~ #
```

kann das Gateway benutzt werden.

Der Zugriff vom Client

Das Gopher-Gateway kann in Form von zwei Dämonprozessen realisiert werden:

- Der Dämon `go500` ist am TCP-Port 5555 anzusprechen. Durch ihn werden die Objekte des Verzeichnisdienstes dem Gopher-Client zugeführt.
- Um in den Daten der LDAP-Datenbank zu browsen und um in ihnen auch suchen zu können, ist es notwendig, den Dämon `go500gw` auf dem Gateway zu konfigurieren.

Der Zugriff vom Client kann dabei grundsätzlich mit dem gleichen Anwendungsprogramm erfolgen. In diesem Abschnitt wird die Kommunikation mit dem Gopher-Gateway auf der Client-Seite mit dem Programm `xgopher` realisiert. Es ist in der grafischen Oberfläche gemäß der Syntax

```
xgopher <Hostname> <TCP-Port>
```

aufzurufen. Ihm muss der Name oder die IP-Adresse des Rechners mitgeteilt werden, auf dem das Gopher-Gateway aktiv ist. Ferner ist der TCP-Port aufzuführen, an dem der Gateway-Rechner angesprochen werden soll.

`xgopher` arbeitet unter einer grafischen Oberfläche. Um zum Beispiel das `go500`-Gateway am `host1` unter dem Port `5555` anzusprechen, ist die folgende Eingabe zu tätigen:

```
user1@host2:~ # xgopher host1 5555 &
user1@host2:~ #
```

Das Programm zeigt sich anschließend wie in Abbildung 4.29 und beinhaltet die Objekte des Directory Information Tree in alphabetischer Reihenfolge, ohne jedoch die hierarchische Darstellung zu verwenden.

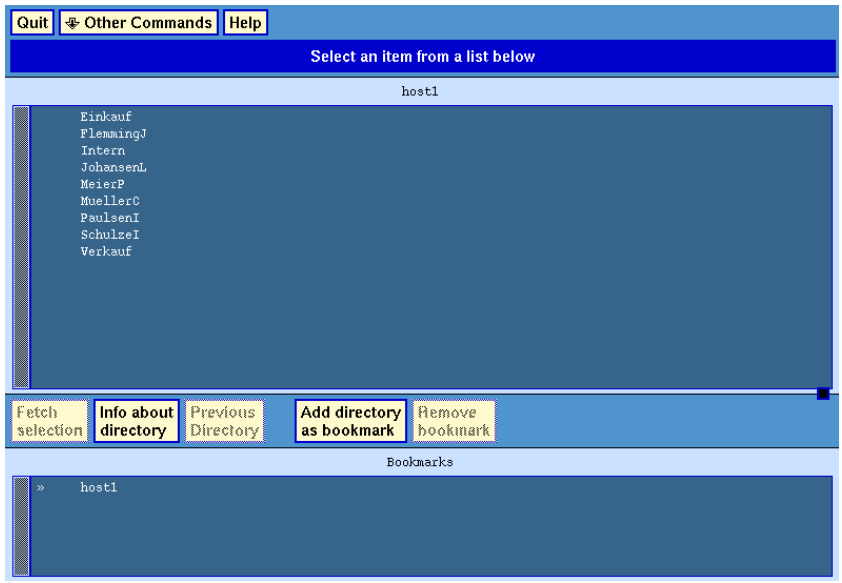

Abbildung 4.29: `xgopher`: Grafische Oberfläche

Um die Attribute zu den angezeigten Objekten einzusehen, muss man den entsprechenden Eintrag in der Objektliste mit der Maus doppelt anklicken. Es erscheint ein separates Fenster, in dem die Eigenschaften aufgeführt sind (siehe Abbildung

4.30). Welche Attribute mit welchen Namen aufgeführt werden, ist in der Datei `ldaptemplates.conf` aus dem Verzeichnis `/etc/openldap/` festgelegt.

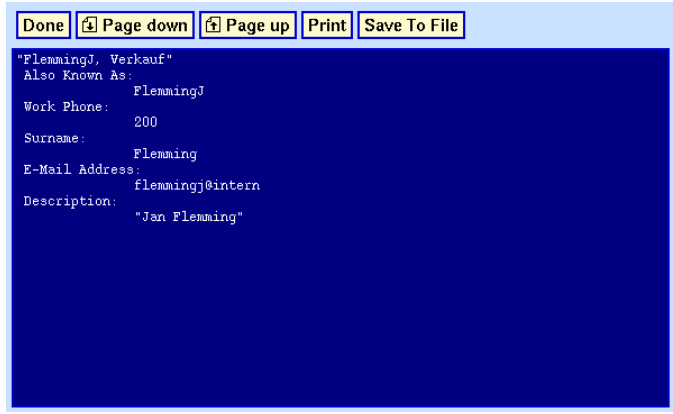

Abbildung 4.30: `xgopher`: Detaillierte Anzeige

Wie sie erweitert werden kann, ist auf Seite 147 in Zusammenhang mit dem Client-Zugriff auf das Finger-Gateway erwähnt. Dort wurde definiert, dass der Nachname (`Surname`) ebenfalls angezeigt wird.

Um mit der Anwendung `xgopher` auf das Gateway `go500gw` zugreifen zu können, muss der folgende Aufruf in der grafischen Oberfläche erfolgen:

```
user1@host2:~ # xgopher host1 7777 &
user1@host2:~ #
```

Es öffnet sich anschließend das in Abbildung 4.31 gezeigte Anwendungsprogramm mit der Möglichkeit, den Verzeichnisdienst zu durchsuchen.

Das Gateway `go500gw` verwendet als Suchbasis die Wurzel des Verzeichnisdienstes. Damit der Start von `xgopher` erfolgreich verläuft, ist es notwendig, dass das `suffix` des LDAP-Servers den Wert `""` besitzt.

Durch einen Doppelklick auf `<idx> Search root` kann anschließend ein Suchkriterium eingegeben werden (siehe Abbildung 4.32).

4.7.4 Das Mail-Gateway

Nahezu jeder Rechner verfügt heutzutage über ein Programm zum Verfassen und Lesen von elektronischer Post. So gibt es eine fast nicht mehr überschaubare Vielzahl von Mail-Programmen für die verschiedenen Rechnerplattformen. Nicht nur für

Linux, sondern auch für andere Betriebssysteme sind somit entsprechende Anwendungen vorhanden. Das im Folgenden vorgestellte Mail-Gateway bietet die Möglichkeit, plattformunabhängig mit einem beliebigen Mail-Programm auf die Daten des Verzeichnisdienstes als Gast zuzugreifen.

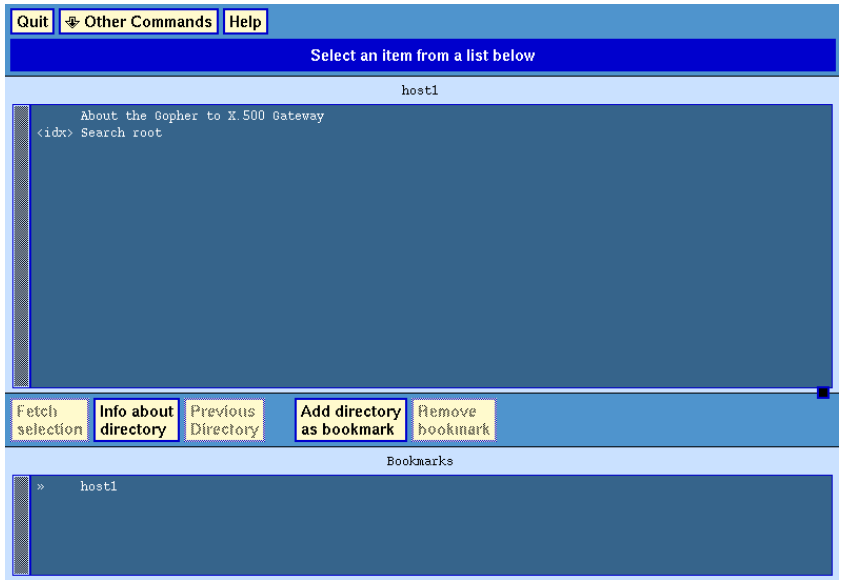

Abbildung 4.31: xgopher: Grafische Oberfläche

Abbildung 4.32: xgopher: Eingabe des Suchkriteriums

Allgemeines

E-Mails werden zwischen dem Sender und dem Empfänger mit dem Simple Mail Transfer Protocol (SMTP) übertragen. Es arbeitet über den TCP-Port 25. Damit der Sendevorgang erfolgreich stattfinden kann, ist es notwendig, dass ein Mail-Server (SMTP-Server) die Daten des Senders entgegennimmt.

Um mittels einer Mail-Anwendung auf dem Client Anfragen an eine LDAP-Datenbank zu senden, werden die E-Mails zunächst an ein Mail-Gateway gesendet. Es konvertiert die SMTP-Daten in das Protokoll LDAP und nimmt anschließend mit dem LDAP-Server Verbindung auf, der letztendlich den Client-Wunsch zu beantworten versucht. Das Ergebnis wird über das Gateway dem Client per E-Mail zugeschickt.

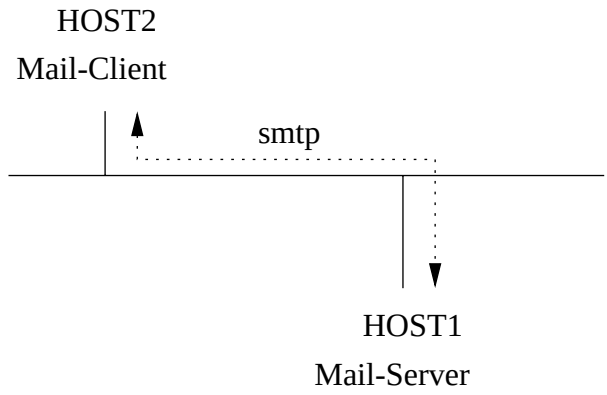

Abbildung 4.33: Das Protokoll `smtp`

Auch bei dieser Art der Konvertierung können das Gateway und der LDAP-Server auf unterschiedlichen Rechnern eingerichtet werden. Im Folgenden wird jedoch eine gemeinsame Installation auf dem Rechner `host1` vorausgesetzt (siehe Abbildung 4.34).

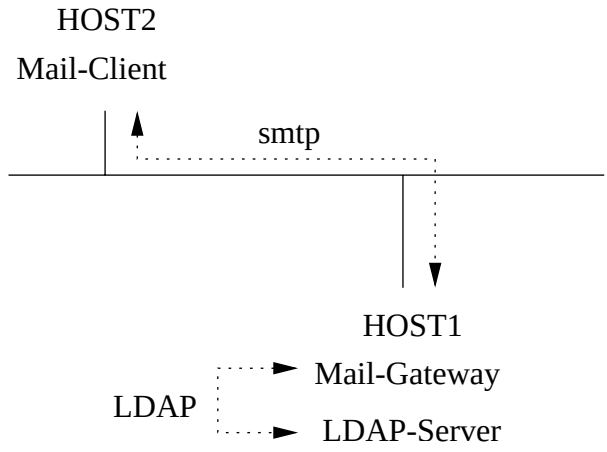

Abbildung 4.34: Mail-Gateway

Das Mail-Gateway besteht aus den Komponenten

- Mail-Server und
- LDAP-Client.

Der Dämon rcpt500

Das Mail-Gateway wird durch die Verwendung des zugehörigen Dämons rcpt500 erzeugt. Er wird mit einem bereits installierten Mail-Server verbunden.

Bevor beschrieben werden kann, wie rcpt500 mit einem bestehenden Mail-Server verknüpft wird, werden in Tabelle 4.36 die Optionen genannt, die beim Start des Gateways verwendet werden können.

```
rcpt500 [Optionen]
```

Option	Beschreibung
-h <Host>	LDAP-Server
-p <Port>	LDAP-Port
-l	Logging aktivieren
-t <Datei>	Template-Datei
-c <Anzahl>	Anzahl der DN-Komponenten
-a	Aliasobjekte verfolgen
-b <Basis>	Suchbasis
-z <Wert>	Maximale Anzahl der Ergebnisse

Tabelle 4.36: Optionen des LDAP-Gateways rcpt500

Über die Parameter -h und -p können der Rechner und der TCP-Port angegeben werden, auf dem der LDAP-Server zu finden ist. Anderenfalls wird der Host aus der Datei ldap.conf verwendet. Das Logging kann mit -l aktiviert werden. Durch die Verwendung der Option -t ist es möglich, den Pfad zu einer Template-Datei anzugeben, sofern er sich von /etc/openldap/ldaptemplates.conf unterscheidet. Die Anzahl der DN-Komponenten, die dem Client mitgeteilt werden, ist mit -c, die Basis für jede Suche mit -b zu definieren. Falls Aliasobjekte verfolgt werden sollen, kann -a verwendet werden. Über -z wird definiert, wie viele Suchergebnisse dem Client maximal mitgeteilt werden.

Nach der Installation eines Linux-Rechners ist im Allgemeinen der Mail-Server sendmail aktiv. Er ist in der Lage, Nachrichten per SMTP zu empfangen und zu senden. Damit das Mail-Gateway einwandfrei funktioniert, muss ein Server wie sendmail vorhanden und aktiv sein.

Damit ein Client per E-Mail LDAP-Anfragen an das Gateway senden kann, ist es notwendig, dass auf dem Gateway-Rechner ein spezieller E-Mail-Account vorhanden ist, über den die Anfragen und die Ergebnisse ausgetauscht werden können. Wenn eine E-Mail dann an diesen Account gesendet wird, so soll rcpt500 gestartet werden. Der Text der Mail ist dabei dem Gateway-Prozess zu übergeben. Dieser Sachverhalt wird

in Zusammenhang mit `sendmail` durch einen Eintrag in der Datei `/etc/aliases` realisiert.

```
ldap:                    "|/usr/lib/openldap/rcpt500 -l -c 3"
```

Da der Mail-Server und der LDAP-Server auf dem Rechner `host1` aktiviert werden sollen, bedeutet die obige Zeile, dass jede Mail, die an `ldap@host1.intern` adressiert ist, dem Mail-Gateway übergeben wird. Es konvertiert die Daten in das Protokoll LDAP und sendet sie dem LDAP-Server am lokalen Rechner über den TCP-Port 389 zu. Ferner wird das Logging der Gateway-Meldungen aktiviert. Der LDAP-Server löst nun die Anfrage auf. Das Ergebnis wird dem Client dann vom Gateway per E-Mail zugesendet. Dabei werden die Objekte mit den letzten drei Komponenten des Distinguished Name erwähnt.

Bei der Datei `/etc/aliases` handelt es sich um eine Textdatei, in der die entsprechenden Einstellungen zum Mail-Server `sendmail` eingetragen werden können. Diese Datei wird jedoch nicht direkt für die Konfiguration des Mail-Servers verwendet. Sie ist des Weiteren in ein Binärformat zu überführen. Dies geschieht sehr einfach durch den Aufruf des Kommandos `newaliases`. Es analysiert die Textdatei und wandelt sie um:

```
root@host1:~ # newaliases
/etc/aliases: 48 aliases, longest 58 bytes, 794 bytes total
root@host1:~ #
```

Anschließend ist die vollzogene Änderung aktiv und kann genutzt werden.

Der Zugriff vom Client

Auf den Rechner `host1` sind nun der Mail-Server, das Mail-Gateway und der LDAP-Server konfiguriert und einsatzbereit.

LDAP-Anfragen können vom Client aus nun mit einem beliebigen Mail-Programm ausgeführt werden. Eine sehr verbreitete und beliebte Anwendung unter Linux ist das Programm `pine` (siehe Abbildung 4.35). Es kann direkt in der Textkonsole gestartet werden und braucht keine grafische Oberfläche:

```
user1@host2:~ # pine
```

Um eine neue Mail zu erstellen, ist der Punkt Compose Message anzuwählen. In der anschließenden Bildschirmmaske wird nun die gewünschte E-Mail eingegeben. In dem Feld `To :` ist die Adresse des Mail-Gateways einzutragen. Unter `Subject :` wird das LDAP-Suchkriterium eingestellt. Es wird später vom Gateway dem LDAP-Server zugefügt und muss mit dem Schlüsselwort `find` beginnen. Nachdem die Eingaben getätigt wurden, kann die E-Mail mit der Tastenkombination **Ctrl+X** abgeschickt werden (siehe Abbildung 4.36):

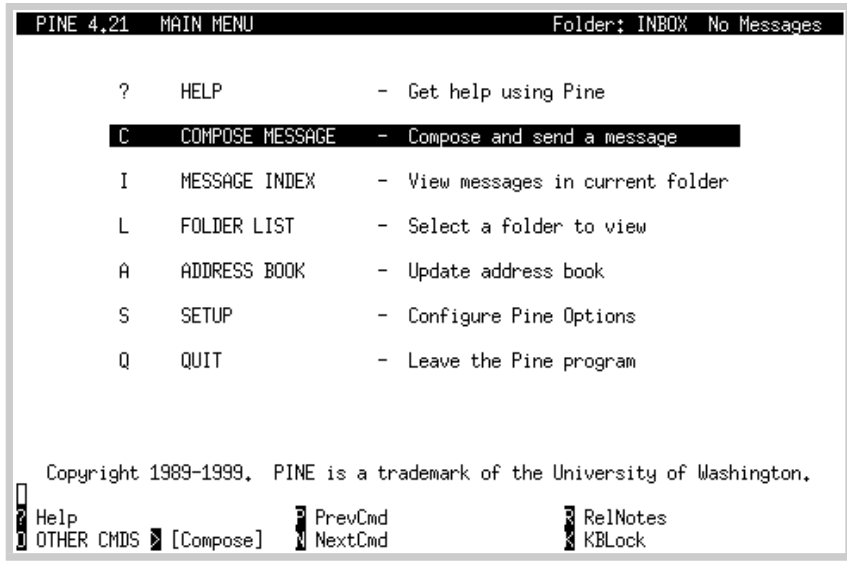

Abbildung 4.35: `pine`: Hauptmenü

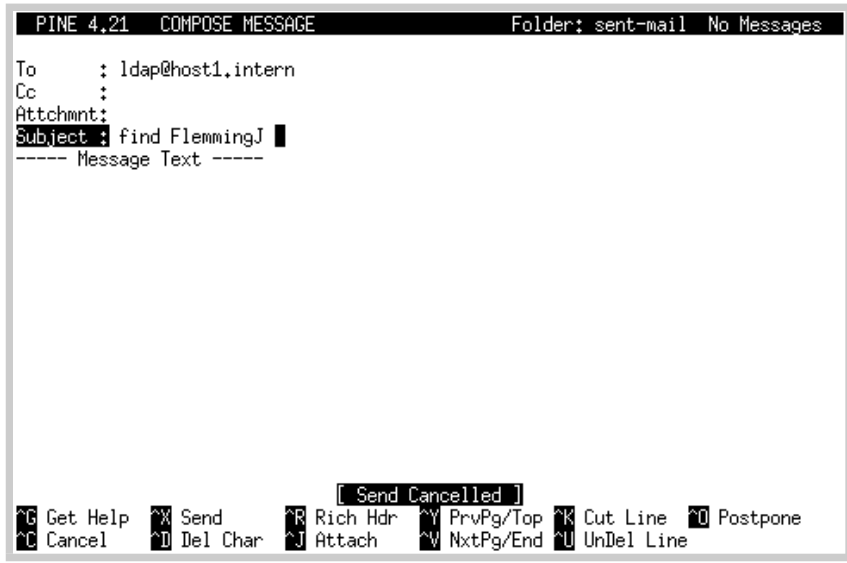

Abbildung 4.36: `pine`: Mail versenden

Die LDAP-Anfrage wird nun über das Gateway am LDAP-Server aufgelöst und beantwortet. Das Ergebnis wird dann dem anfragenden Benutzer wieder per E-Mail zugesendet. Im Beispiel wird es demnach an `user1@host2.intern` gesendet.

Mit dem Programm `pine` können eingehende Nachrichten nach dem Start über den Menüpunkt MESSAGE INDEX eingesehen werden. Es wird dann eine Liste aller neuen E-Mails angezeigt, und die gewünschte muss nur noch ausgewählt werden.

Es sei nochmals erwähnt, dass es völlig unerheblich ist, mit welchem Mail-Programm und von welchem Betriebssystem aus das Mail-Gateway angesprochen wird. Falls das hier kurz vorgestellte Programm `pine` unter Linux verwendet wird, so können weitere Informationen zu dieser Anwendung in [5] nachgelesen werden.

Da in den letzten Abschnitten bereits global in der Datei `ldaptemplates.conf` im Verzeichnis `/etc/openldap/` definiert wurde, dass bei Anfragen auf Objekte der Klasse `person` der Nachname (`Surname`) mit angegeben werden soll, findet er sich auch in der Ausgabe des Mail-Gateways (siehe Abbildung 4.37).

```
  PINE 4.21   MESSAGE TEXT                Folder: INBOX  Message 1 of 1 ALL

Date: Thu, 23 Nov 2000 08:46:32 +0100
From: Directory Query Program <Dir-Query@host1.intern>
To: user1@host2.intern
Subject: Re: find FlemmingJ

1 exact match found for 'FlemmingJ':

"FlemmingJ, Verkauf, Intern"
 Also Known As:
                FlemmingJ
 Work Phone:
                200
 Surname:
                Flemming
 E-Mail Address:
                flemmingj@intern
 Description:
                "Jan Flemming"

                            [ALL of message]
? Help       < MsgIndex   P PrevMsg      -  PrevPage  D Delete     R Reply
O OTHER CMDS > ViewAttch  N NextMsg     Spc NextPage  U Undelete   F Forward
```

Abbildung 4.37: `pine`: Mail lesen

Da das Programm `rcpt500` mit der Option `-c 3` gestartet wurde, ist der Benutzer `FlemmingJ` als `FlemmingJ, Verkauf, Intern` aufgeführt.

Auf die oben genannte Weise ist es sehr leicht möglich, Informationen zu einem bestimmten Objekt zu bekommen, sofern auf den DIT-Eintrag ohne gesonderte Anmeldung mit den Standardrechten zugegriffen werden kann.

Neben der einfachen Angabe eines Benutzerobjekts kann in der E-Mail, die an das Gateway gesendet werden soll, jedoch auch ein komplettes Suchkriterium genutzt werden, wie es auch bei der Verwendung des Kommandos `ldapsearch` zu verwenden war.

Welcher Syntax dieses Kriterium allgemein folgen muss, ist in Abschnitt 4.6.6 beschrieben. Er beginnt auf der Seite 106.

So beschreibt das folgende Kriterium alle Objekte des Directory Information Tree, deren Common Name (`cn`) mit der Zeichenkette `Flem` oder (`|`) der Zeichenkette `Paul` beginnt:

```
(|(cn=Flem*)(cn=Paul*))
```

Es kann auch als `Subject` einer E-Mail verwendet werden (siehe Abbildung 4.38).

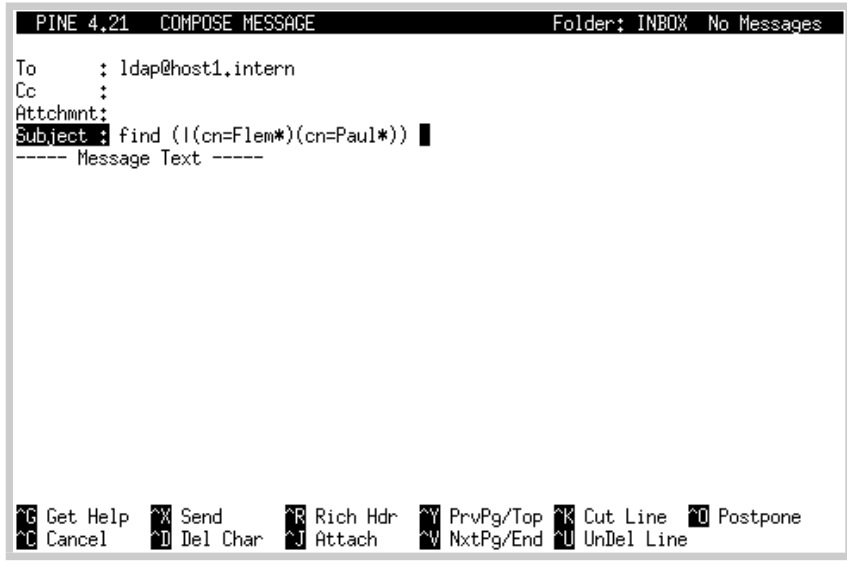

Abbildung 4.38: `pine`: Mail versenden

Sobald die Mail versandt wurde, wird sie vom LDAP-Server über das Mail-Gateway bearbeitet und dem Anwender anschließend ebenfalls als elektronische Post zugeschickt.

Bei mehr als einem Ergebnis werden lediglich die Objekte, jedoch nicht deren Attribute aufgeführt (siehe Abbildung 4.39).

Mit der in diesem Abschnitt dargelegten Zugriffsmöglichkeit auf die Daten des Verzeichnisdienstes ist der Flexibilität bezüglich des Anwendungsbereichs der LDAP-Software nahezu keine Grenze gesetzt. Durch die große Verbreitung von E-Mail-Programmen ist somit im Prinzip jeder Anwender in der Lage, auf den Directory Information Tree zuzugreifen.

Abbildung 4.39: `pine`: Mail lesen

4.7.5 Das Web-Gateway

Der plattformübergreifende Zugang zu den Daten des Directory Service kann zum einen sehr komfortabel und einfach über das Mail-Gateway `rcpt500` erfolgen.

Nahezu genauso verbreitet wie E-Mail-Programme sind Internet-Browser. Um sich mit ihnen im gesamten Verzeichnisdienst zu bewegen und um in ihnen schnell und einfach suchen zu können, ist der Einsatz eines Web-Gateways notwendig.

Allgemeines

Der Austausch von HTML-Dokumenten erfolgt in Netzwerken grundsätzlich über das HTTP-Protokoll. Es basiert ebenfalls auf dem Transmission Control Protocol (TCP). Für die korrekte Kommunikation ist auf der Client-Seite ein Browser wie zum Beispiel `netscape` erforderlich. Auf der anderen Seite müssen die HTTP-Anfragen von einem Web-Server (HTTP-Server) bearbeitet werden (siehe Abbildung 4.40).

Auch das Web-Gateway, welches im Folgenden auf dem gleichen Rechner installiert wird wie der LDAP-Server, konvertiert die ihm zugetragenden HTTP-Anfragen ins Protokoll LDAP und sendet sie dem Server zu.

Das Web-Gateway wird durch die Kombination der beiden grundlegenden Dienste

- Web-Server und

- LDAP-Client

umgesetzt.

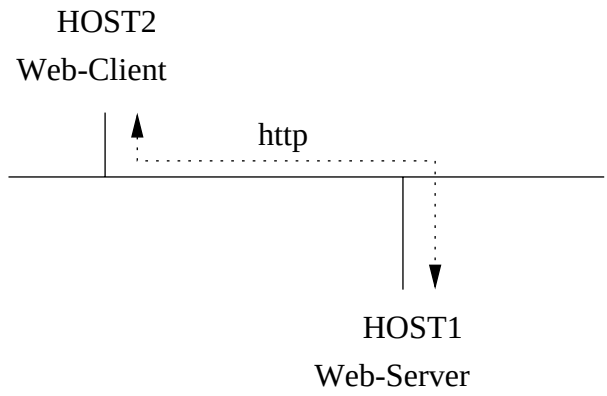

Abbildung 4.40: Das Protokoll `http`

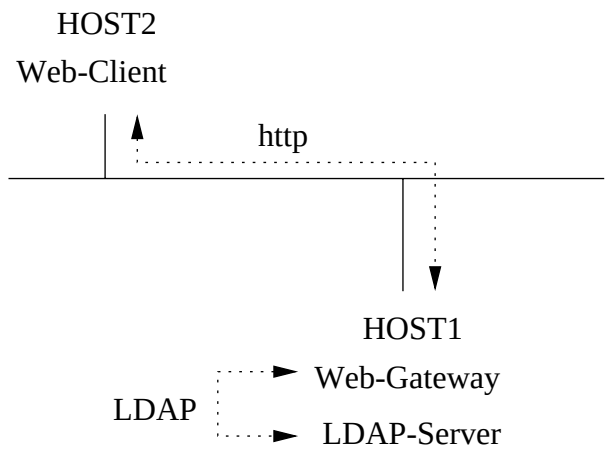

Abbildung 4.41: Web-Gateway

Der Dämon `web500gw`

Um ein Web-Gateway zu erzeugen, muss auf dem ausgewählten Rechner der Dämon `web500gw` konfiguriert und aktiviert werden. In diesem Abschnitt werden zunächst die Optionen beschrieben, die bei dessen Start verwendet werden können (siehe Tabelle 4.37).

```
web500gw [Optionen]
```

Der Dämon ist im Verzeichnis `/usr/sbin/` zu finden. Dieser Pfad kann jedoch je nach Linux-Distribution variieren.

Option	Beschreibung
`-x <Host>`	LDAP-Server
`-P <Port>`	LDAP-Port
`-p <Port>`	`web500gw`-Port
`-I`	Gestartet vom Inet-Dämon
`-l`	Logging aktivieren
`-e <Pfad>`	Verzeichnis weiterer Dateien
`-c <Datei>`	Konfigurationsdatei
`-a`	Aliasobjekte verfolgen
`-D <Level>`	LDAP-Debuglevel
`-d <Level>`	`web500gw`-Debuglevel

Tabelle 4.37: Optionen des LDAP-Gateways `web500gw`

Dem Gateway können die Adresse und der Port des anzusprechenden LDAP-Servers über die Optionen `-x` und `-P` mitgeteilt werden. Der Port, an dem per HTTP eine Verbindung mit dem Web-Gateway hergestellt werden kann, wird über die Option `-o` festgelegt. Falls `web500gw` über den Inet-Dämon gestartet wird, ist dieses mit dem Parameter `-I` zu kennzeichnen. Die Option `-l` aktiviert das Logging über `LOCAL3`, und `-a` dient zum Auflösen und Verfolgen von Aliasobjekten in der Datenbank.

Der Pfad zu weiteren Konfigurationsdateien, wie `ldaptemplates.conf` und `ldapfilter.conf` wird mit der Option `-e` eingestellt. Per Default wird zurzeit das Verzeichnis `/etc/web500/` verwendet.

Der Dämon `web500gw` wird zum einen über die Optionen und zum anderen über eine Konfigurationsdatei gesteuert. Sie hat den Namen `web500gw.conf` und befindet sich ebenfalls in `/etc/web500/`. Falls jedoch die Einstellungen aus einer anderen Datei gelesen werden sollen, ist dem Dämon dies durch Verwendung der Option `-c` mitzuteilen.

Außerdem kann das Debugging der LDAP- und der Gateway-Aktionen durch die Parameter `-D` und `-d` aktiviert werden.

Das Gateway zur Konvertierung zwischen den Protokollen HTTP und LDAP wird, wie die bereits vorgestellten Gateways auch, vom Inet-Dämon erst dann gestartet, wenn Anfragen am entsprechenden TCP-Port eintreffen. Damit dieses geschehen kann, ist es zunächst erforderlich, in der Datei `/etc/services` einen Eintrag zu

ergänzen, der dem TCP-Port 8888, an dem das Gateway reagiert, einen Namen zuweist:

```
web500gw  8888/tcp        # Web500gw-Gateway
```

Anschließend kann mit dem soeben definierten Namen in der Datei `/etc/inetd.conf` eine Zeile eingetragen werden, die den Start des Web-Gateways bei eintreffenden Anfragen anregt:

```
web500gw stream tcp nowait nobody /usr/sbin/web500gw web500gw -I
```

Falls am Port `web500gw` (TCP: 8888) mit dem Datenstrom-Protokoll TCP (`stream tcp`) ein Verbindungswunsch erkannt wird, so wird sofort (`nowait`) mit den Rechten des Benutzers `nobody` das Web-Gateway unter Verwendung der Option `-I` gestartet.

Abschließend muss der Inet-Dämon neu gestartet werden. Der Neustart erfolgt unter einer SuSE-Distribution durch den folgenden Aufruf:

```
root@host1:~ # /etc/rc.d/init.d/inetd reload
Reload INET services (inetd)                              done
root@host1:~ #
```

Die Datei `web500gw.conf`

Das Web-Gateway kann nicht nur durch die Optionen beim Aufruf des Dämons `web500gw` gesteuert werden, sondern auch durch eine ASCII-basierte Konfigurationsdatei. Sofern über die Option `-c` keine alternative Datei angegeben wurde, wird `/etc/web500/web500gw.conf` verwendet. In ihr sind eine Vielzahl von Einstellungen vorzunehmen, deren Syntax und Bedeutung in diesem Abschnitt beschrieben werden.

Der Parameter `port`

Über den Parameter `port` wird in der Konfigurationsdatei eingestellt, an welchem TCP-Port das Web-Gateway arbeitet:

```
port: <Portnummer>
```

Der so eingestellte Wert wird durch die Verwendung der Option `-p` beim `web500gw`-Aufruf überschrieben.

```
port: 8888
```

Der Parameter `ldapserver`

Mit `ldapserver` wird dem Gateway der Name oder die IP-Adresse des LDAP-Servers mitgeteilt:

```
ldapserver: <Hostname>
```

Diese Standardeinstellung wird durch die Option `-x` beim Gateway-Aufruf überschrieben.

```
ldapserver: host1.intern
```

Der Parameter `ldapport`

Neben dem Namen des Rechners, an dem der LDAP-Server aktiv ist, kann auch dessen TCP-Port angegeben werden:

```
ldapport: <Portnummer>
```

Falls die Option `-P` in Zusammenhang mit `web500gw` benutzt wird, so wird dieser Wert ignoriert.

```
ldapport: 389
```

Der Parameter `otherservers`

Falls im Uniform Resource Locator über dieses Gateway auch andere LDAP-Server benutzt werden dürfen, so kann dies mit dem Parameter `otherservers` festgelegt werden:

```
otherservers: <yes|no>
```

Die Standardeinstellung lautet:

```
otherservers: yes
```

Der Parameter `timelimit`

Wie viele Sekunden maximal für LDAP-Operationen verwendet werden, wird mit

```
timelimit: <Sekunden>
```

festgelegt, z. B.:

```
timelimit: 240
```

Der Parameter `sizelimit`

Die Anzahl der Ergebnisse, die nach einer LDAP-Anfrage maximal benutzt werden, kann mit dem Parameter `sizelimit` angegeben werden:

```
sizelimit: <Anzahl>
```

Falls eine Anzahl von 0 benutzt wird, so bedeutet dieses, dass keine Grenze verwendet wird. Es werden somit alle Ergebnisse angenommen:

```
sizelimit: 0
```

Der Parameter homedn

Die Standardposition des Web-Gateways im Verzeichnisdienst wird über das Wort homedn definiert:

```
homedn: <Dinstinguished Name>
```

Es wird der DN des gewünschten Containerobjekts angegeben:

```
homedn: o=Intern
```

Der Parameter rootishome

Mit dem Parameter rootishome wird festgelegt, ob der über homedn angegebene Container benutzt wird, falls keine genaue Angabe bezüglich des gewünschten Objektes erfolgt:

```
rootishome: <on|off>
```

```
rootishome: on
```

Der Parameter web500dn

Die LDAP-Daten, die später im Internet-Browser über das Web-Gateway angezeigt werden, entsprechen denen, die mit einem Gastzugang sichtbar sind. Falls das Web-Gateway spezielle Rechte am LDAP-Server verwenden soll, so muss zuvor eine Anmeldung erfolgen. Mit diesem Parameter wird der Distinguished Name des Benutzerobjekts angegeben.

```
web500dn: [Distinguished Name]
```

Fehlt die DN-Angabe, so bedeutet dies einen Zugang mit den Defaultrechten des Servers:

```
web500dn:
```

Der Parameter web500pw

Das zu web500dn gehörige Passwort wird mit

```
web500pw: [Passwort]
```

angegeben:

```
web500pw:
```

Der Parameter derefaliases

Falls Alias-Objekte im Verzeichnisdienst vom Gateway verfolgt werden sollen, so wird dies durch die Verwendung von

```
derefaliases: on
```

festgelegt.

Der Parameter `syslog`

Das Web-Gateway ist in der Lage, alle Anforderungen zu protokollieren, die an es gestellt werden. Dazu werden dem `syslogd` die Meldungen über `LOCAL3` zugeführt. Mit

```
syslog: <yes|no>
```

kann eingestellt werden, ob dies stattfinden soll:

```
syslog: yes
```

Der Parameter `logformat`

Welches Format die Logeinträge haben, wird mit `logformat` konfiguriert:

```
logformat: <Format>
```

Das Format soll an dieser Stelle nicht näher beschrieben werden, da die Standardeinstellung ausreichend ist.

```
logformat: %h "%r" %s %e "%a" %b %l "%f" %T %x
```

Der Parameter `showonematch`

Über das Web-Gateway ist es auch möglich, Suchanfragen an den LDAP-Server abzusetzen. Falls sie genau ein Ergebnis liefern, kann entweder nur dessen Name oder auch dessen Attributliste angezeigt werden.

```
showonematch: <yes|no>
```

Bei `yes` werden in einem solchen Fall auch die Attribute aufgeführt.

```
showonematch: yes
```

Der Parameter `ufnsearch`

Wenn man Objekte im Verzeichnisdienst suchen will, muss man diese über ihren Distinguished Name angeben. Falls jedoch die Objektklassen bei der DN-Angabe weggelassen werden, kann die Suche trotzdem erfolgreich sein, wenn der folgende Parameter gesetzt ist:

```
ufnsearch: <on|off>
```

So ist zum Bespiel eine Suche nach `Flemmingj`, `Verkauf`, `Intern` möglich. Die Abkürzung UFN steht für User Friendly Notation.

```
ufnsearch: on
```

Der Parameter `subsearch`

Falls Suchoperationen im Verzeichnisdienst ausgeführt werden, so ist es möglich, deren Reichweite zu definieren. Für alle mit dem Parameter `subsearch` genannten Objektklassen erfolgt eine Suche im gesamten Subtree.

```
subsearch: <Objektklassen>
```

Die Klassen sind durch Kommas getrennt anzugeben:

```
subsearch: organization, organizationalUnit
```

Der Parameter `lastmodified`

Falls Objekte das Attribut `lastModifiedTime` aufweisen, so kann dessen Wert im HTML-Header angezeigt werden:

```
lastmodified: <on|off>
```

Falls dies nicht geschehen soll, so ist der Parameter auszuschalten.

```
lastmodified: off
```

Der Parameter `etcdir`

Das Web-Gateway benötigt, um einwandfrei zu funktionieren, diverse Dateien. Sie werden in dem Verzeichnis gesucht, das mit `etcdir` festgelegt wird:

```
etcdir: <Verzeichnis>
```

In der Regel handelt es sich um den Pfad `/etc/web500/`:

```
etcdir: /etc/web500
```

Der Parameter `access`

Mit dem Parameter `access` können Rechte festgelegt werden. So ist es zum Beispiel möglich, dass für alle Anfragen aus einer bestimmten Domäne eine Anmeldung beim Verzeichnisdienst erfolgt, wodurch die entsprechenden Clients mehr Rechte bekommen, als per Default festgelegt ist. In der Konfigurationsdatei `web500gw.conf` muss mindestens eine `access`-Zeile existieren. Mehrere Zeilen sind ebenfalls zulässig.

```
access <Name:Ausdruck:Recht:Limit:Sprache:Position:DN:PW:Suffix>
```

Für die Platzhalter sind die folgenden Werte einzutragen:

1. Als Erstes steht ein beliebiger Name, mit dem die Zugriffsregel identifiziert wird.
2. Anschließend folgt ein Ausdruck, für welche Rechner die Regel gelten soll. Die Zeichen `.*` bedeuten, dass alle Rechner von der Regel betroffen sind.
3. Danach erfolgt die Angabe eines Rechtes, wie zum Beispiel `read`.
4. Danach wird die maximale Anzahl der zu benutzenden Ergebnisse definiert (`Limit`).
5. Dann folgt eine Vorgabe bezüglich der Sprache.
6. Die Standardposition im Verzeichnisdienst wird in der sechsten Spalte definiert.
7. Mit welchem DN die Anmeldung am Directory Service für die Rechner erfolgen soll, ist Inhalt der siebten Spalte.
8. Anschließend ist das zugehörige Passwort aufzuführen.
9. Zum Schluss kann noch ein Suffix definiert werden.

Die folgende Zeile bedeutet demnach, dass alle Rechner lesenden Zugriff haben, maximal 50 Ergebnisse einer Suche verwenden, die Sprache Englisch benutzen und in `o=Intern` beheimatet sind:

```
access: World : .* : read : 50 : en : o=Intern : : :
```

Der Parameter `language`

Ebenfalls mindestens einmal muss eine Zeile mit dem Parameter `language` definiert werden. Sie beschreibt, wer welche Sprache nutzen darf und welche Endung die zugehörigen Dateien im Verzeichnis `etcdir` haben:

```
language: <Sprache:Ausdruck:Endung>
```

Die folgende Zeile erlaubt Englisch für alle. Die Dateien haben keine Endung.

```
language en:.*:
```

Der Parameter `browser`

Ferner besteht in der Datei die Möglichkeit, das Erscheinungsbild der HTML-Seiten zu konfigurieren. Diese Konfiguration ist vom verwendeten Browser abhängig und kann deshalb mehrfach in der Datei vorkommen:

```
browser: <Name:Ausdruck:Optionen:Flags:Display:Navigation>
```

So beschreibt die folgende Anweisung, dass für alle Browser alle HTML-Tags benutzt werden können. Ferner erfolgt eine Darstellung in Tabellenform, wobei zum Navigieren Menüs verwendet werden. Weitere Informationen zur `browser`-Zeile sind in der Konfigurationsdatei aufgeführt, die nach der Installation vorhanden ist.

```
browser: All : .* : html32 : table : top,menu
```

Nachdem nun alle relevanten Parameter beschrieben wurden, wird in Beispiel 4.56 die komplette Konfigurationsdatei abgedruckt.

Beispiel 4.56: Konfigurationsdatei des Web-Gateways

```
root@host1:~ # cat /etc/web500/web500gw.conf
#
# Konfigurationsdatei des Web-Gateways
#
# web500gw.conf
#
# Erstellt von Jens Banning
#

#
# Allgemeines
#

ldapserver:     host1.intern
ldapport:       389

port:           8888
otherservers:   yes

timelimit:      240
sizelimit:      0

homedn:         o=Intern
rootishome:     on

web500dn:
web500pw:

syslog:         yes
logformat:      %h "%r" %s %e "%a" %b %l "%f" %T %x

showonematch:   yes
ufnsearch:      on

subsearch:      organization, organizationalUnit
lastmodified:   off

etcdir:         /etc/web500

#
# Rechte
#

access:         World : .* : read : 50 : en : o=Intern : : :
```

```
#
# Sprachen
#

language:       en : .* :

#
# Browser
#

browser:        All : .* : html32 : table : top,menu

root@host1:~ #
```

Der Zugriff vom Client

Abbildung 4.42 zeigt die Homepage des Web-Gateways. Sie wird über die URL

`http://host1.intern:8888/`

aufgerufen. Alternativ kann dieser Angabe auch direkt ein DN folgen, z. B.:

`http://host1.intern:8888/cn=FlemmingJ, ou=Verkauf, o=Intern`

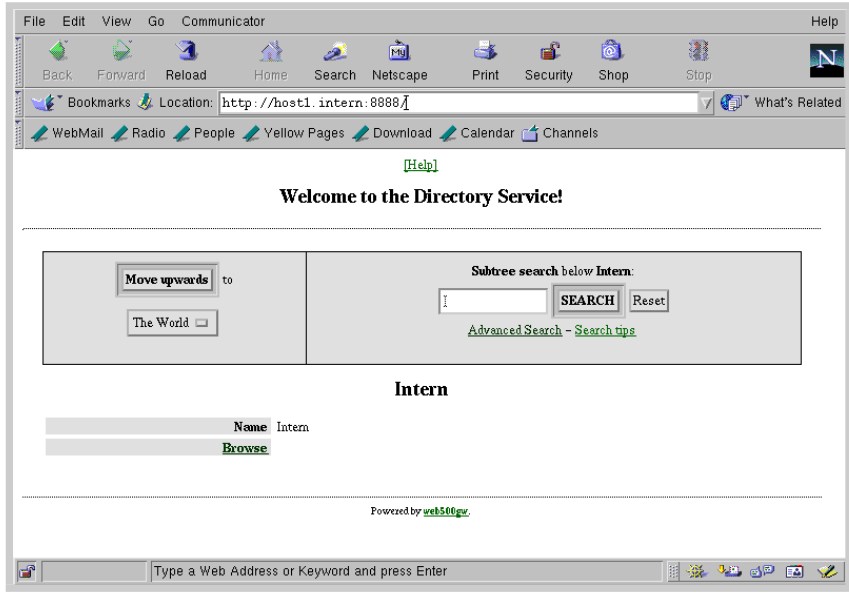

Abbildung 4.42: `netscape`: Web-Gateway

Die dadurch erscheinende Maske erlaubt es dem Anwender, sich nun beliebig im Verzeichnisdienst zu bewegen (siehe Abbildung 4.42). Die Navigation erfolgt dabei intuitiv und bedarf keiner Beschreibung.

Das Suchen nach Objekten im Verzeichnisdienst kann auf zwei Arten erfolgen. Zum einen kann direkt in dem Feld vor SEARCH ein Suchkriterium eingegeben werden. Es muss der Syntax aus Abschnitt 4.6.6 genügen.

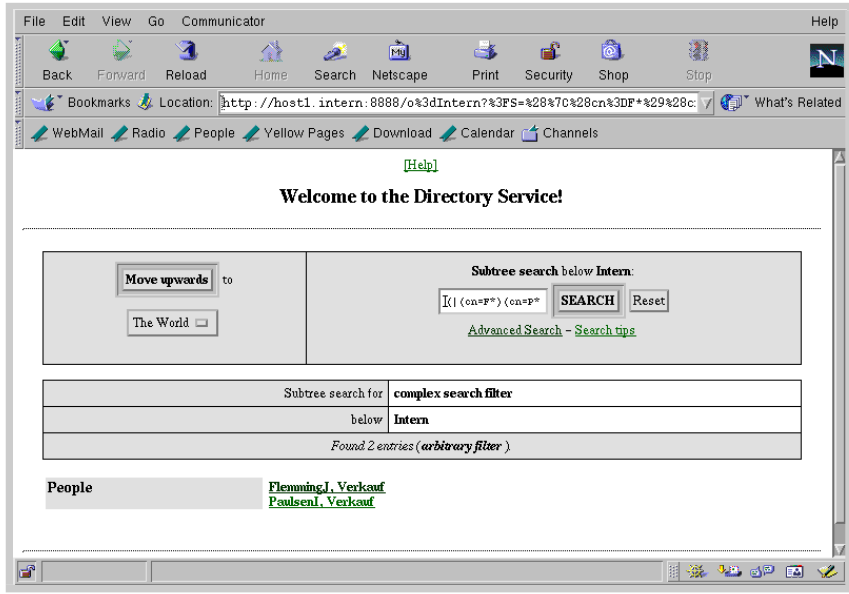

Abbildung 4.43: `netscape`: Suche über das Web-Gateway

Ferner ist es auch möglich, eine fortgeschrittene Suche zu starten. Dies geschieht durch den Punkt `Advanced Search` (siehe Abbildung 4.44).

Falls eine einfache Suche zum Beispiel nach `FlemmingJ` genau ein Ergebnis liefert, so wird das Objekt mit all seinen Attributen angezeigt (siehe Abbildung 4.45).

Das Web-Gateway benutzt nicht die Datei `ldaptemplates.conf` aus dem Verzeichnis `/etc/openldap/`, sondern aus dem Verzeichnis `/etc/web500/`. Damit das Attribut `sn` ebenfalls als `Surname` angezeigt wird, muss dort in der Personen-Schablone die folgende Zeile ergänzt werden:

```
item cis        "Surname"               sn
```

Auf diese Weise wird festgehalten, dass es sich bei dem Attribut `sn` um einen Case Ignore String handelt, also einen Text, bei dem nicht zwischen Groß- und Kleinschreibung unterschieden wird. Das Attribut wird in der HTML-Seite des Gateways mit dem Namen `Surname` bezeichnet.

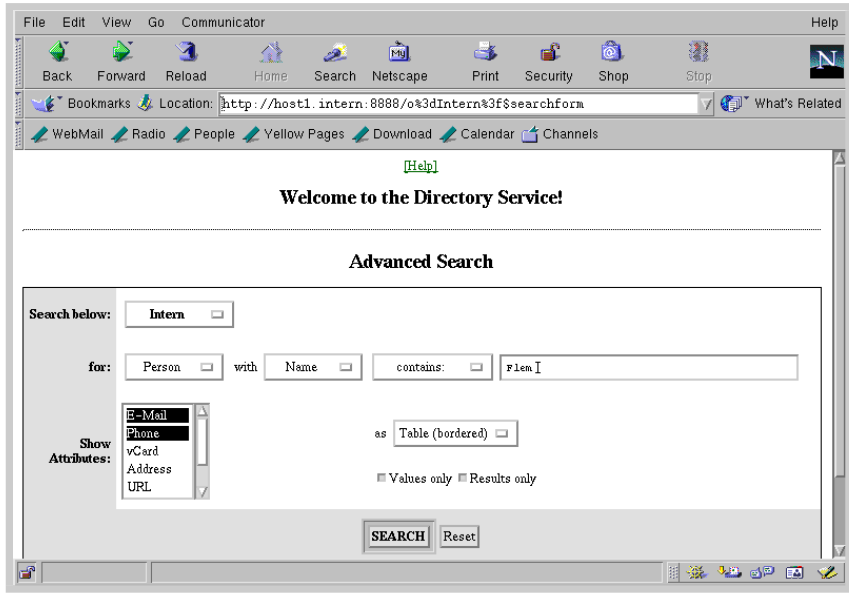

Abbildung 4.44: `netscape`: Erweiterte Suche über das Web-Gateway

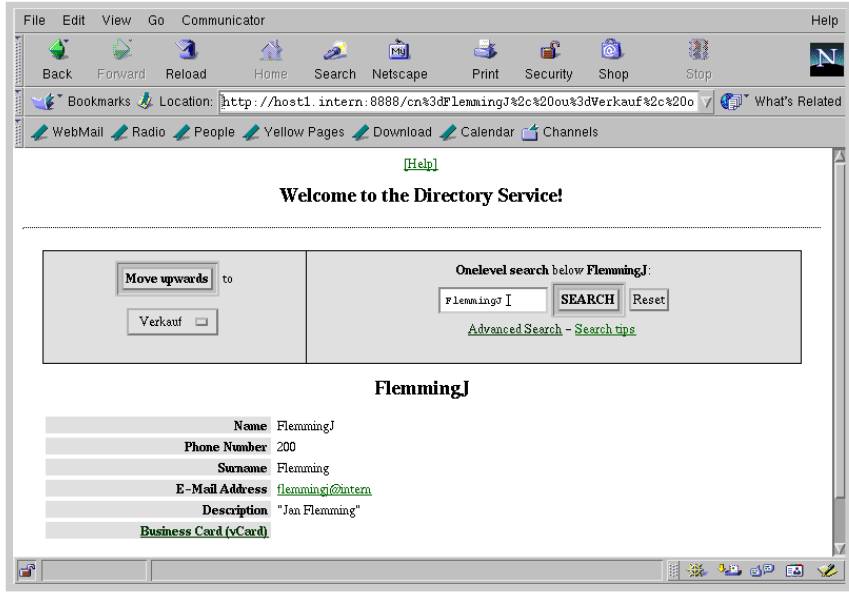

Abbildung 4.45: `netscape`: Detaillierte Ansicht im Web-Gateway

5 Beispielanwendungen

Bisher wurden in diesem Buch alle wichtigen Informationen zu den Komponenten der LDAP-Software von der OpenLDAP-Organisation beschrieben. Die Funktionen

- des LDAP-Servers,
- des LDAP-Clients und
- der LDAP-Gateways

sind bekannt und können nun verwendet werden.

In diesem Kapitel werden drei komplexe Beispiele aus der Praxis vorgestellt, die auf den zuvor beschriebenen Komponenten aufbauen.

In Abschnitt 5.1 auf der Seite 178 wird eine Hardware-Inventarisierung realisiert. Über sie werden Informationen über die Rechner im Netzwerk im Directory Service abgespeichert.

Im zweiten Beispiel auf der Seite 193 in Abschnitt 5.2 wird die zentrale Verwaltung des Network File System (NFS) beschrieben. Dazu werden in der LDAP-Struktur Objekte festgehalten, die jeden Rechner im Netzwerk beschreiben, und die für ihn festlegen, welche Verzeichnisse er welchen Hosts im Netz zur Verfügung stellen soll. Die so definierten Werte werden beim Starten der NFS-Server auf den Rechnern verwendet.

Im letzten Beispiel wird zunächst ein Server des Network Information System (NIS) installiert. Er dient zur zentralen Pflege von Benutzerkonten im Netzwerk. Die Anwender werden im Beispiel ferner automatisch in der LDAP-Datenbank festgehalten. Sofern sich Clients an ihrem Rechner anmelden wollen, erfolgt die Verifizierung der User-ID und des Passwortes üblicherweise am NIS-Server. In dem Beispiel in Abschnitt 5.3 auf Seite 203 werden der Benutzername und das Kennwort jedoch direkt an der LDAP-Datenbank und nicht am NIS-Server verifiziert. Da im Directory Service alle NIS-Benutzer vorhanden sind, können somit deren Passwörter auf diese Weise geändert werden.

Die Beispiele in diesem Kapitel sind voneinander unabhängig und bauen daher nicht aufeinander auf. Die Replizierung der LDAP-Daten wird generell nicht durchgeführt. In der Praxis sollte eine Duplizierung der Daten des Verzeichnisdienstes jedoch so, wie im letzten Kapitel beschrieben, grundsätzlich erfolgen.

5.1 Hardware-Inventarisierung

Als erste Anwendung soll der Verzeichnisdienst dazu benutzt werden, von jedem Rechner im Netzwerk die Daten einer dort durchgeführten Hardware-Inventarisierung festzuhalten.

5.1.1 Funktion

Die Funktionsweise der Hardware-Inventarisierung basiert dabei zum einen auf einem Shell-Skript, das auf jedem Linux-Client installiert werden muss. Es soll beim Rechnerstart ausgeführt werden und die gewünschten Informationen über die Hardware zusammentragen. Anschließend soll es die so gewonnenen Daten in einem separaten Container des Verzeichnisdienstes abspeichern. Als Objektname ist dabei der Hostname des Rechners zu verwenden.

Als LDAP-Server verwenden wir die im letzten Kapitel erwähnte Datenbank. Sie besteht aus der Organisation `Intern` und den organisatorischen Einheiten `Einkauf` und `Verkauf`. Die Host-Objekte sollen dabei in einer organisatorischen Einheit `Hardware` abgelegt werden (siehe Abbildung 5.1).

An den LDAP-Clients müssen im Prinzip keine Änderungen vorgenommen werden, da sie lediglich die Verbindung zum Server herstellen müssen, um die Daten entsprechend einpflegen zu können.

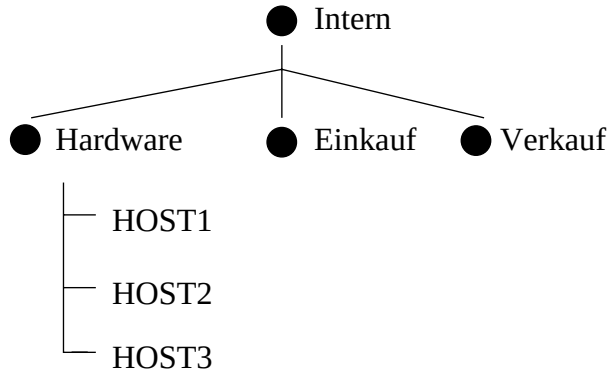

Abbildung 5.1: Hardware-Inventarisierung unter `Intern`

Das Shell-Skript, das die Informationen über die Hardware sammelt und zum LDAP-Server überträgt, ist so zu installieren, dass es abgearbeitet wird, wenn das Linux-System in den Standard-Runlevel wechselt. Ein Runlevel ist eine Art Rechnerkonfiguration, der eine bestimmte Nummer zugeordnet ist. Der Runlevel 2 zum Beispiel enthält bei einer SuSE-Distribution die Konfiguration des Netzwerkes im Mehrbenutzerbetrieb bei einer Anmeldung im Textmodus. Falls der Linux-Rechner beim

Starten in den Runlevel 2 wechselt, so ist das Skript zur Hardware-Inventarisierung in dessen Konfiguration mit einzubeziehen. Weitere Informationen finden Sie in den Büchern [4] und [5].

5.1.2 Konfiguration des Servers

Der LDAP-Server kann zunächst so konfiguriert werden, wie es bisher in diesem Buch üblich war:

```
root@host1:~ # cat /etc/openldap/slapd.conf
#
# Konfigurationsdatei des LDAP-Servers
#
# slapd.conf
#
# Erstellt von Jens Banning
#

#
# Allgemeines
#

include         /etc/openldap/slapd.at.conf
include         /etc/openldap/slapd.oc.conf

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

loglevel        256
schemacheck     on

sizelimit       500
timelimit       3600

#
# Datenbank
#

database        ldbm
lastmod         off

cachesize       1000
dbcachesize     100000

directory       /var/lib/ldap
suffix          "o=Intern"

rootdn          "cn=Admin, o=Intern"
```

```
rootpw          {CRYPT}zsOQLoUVZvS6E

#
# Rechte
#

defaultaccess   read

access to attr=userPassword
       by self write
       by * none

access to attr=telephoneNumber
       by self write
       by * read

access to dn="cn=Admin,o=Intern"
       by * none

access to *
       by * read

root@host1:~ #
```

Somit hat das Objekt `Admin` im Container `Intern` alle Rechte auf den gesamten Verzeichnisdienst, sofern es sich mit dem Kennwort `linux` am Directory Information Tree (DIT) anmeldet. Der Administrator kann dadurch in allen Containern Einträge erstellen, ändern und löschen. Für die Hardware-Inventarisierung ist es jedoch notwendig, dass sich die Clients ebenfalls mit einer Kennung am DIT anmelden, die es erlaubt, die Host-Objekte im Container `Hardware` zu erstellen. Das `Admin`-Objekt kann hier nicht benutzt werden, da es für diesen Anwendungszweck zu viele Rechte in anderen Containern besitzt. Aus diesem Grunde wird unter `Hardware` ein Benutzerobjekt mit dem Namen `HWAdmin` erstellt. Dieser Hardware-Administrator ist nur in der Lage, die Objekte unterhalb der organisatorischen Einheit `Hardware` zu pflegen. In den anderen Zweigen der Baumstruktur hat er nur die per Default definierten Zugriffsmöglichkeiten.

Die Datei `slapd.conf` muss nun dahingehend verändert werden, dass `access`-Anweisungen erstellt werden, die dem `HWAdmin` die notwendigen Rechte zuweisen. Die Attribute des `HWAdmin` sollen jedoch von ihm selbst nicht verändert werden können. Dieses Privileg bleibt dem DIT-Administrator `Admin` vorbehalten. Ferner ist das Passwort für alle anderen Objekte nicht sichtbar.

```
#
# Rechte
#

defaultaccess         read
```

```
access to dn="cn=HWAdmin,ou=Hardware,o=Intern" attr=userPassword
       by self read
       by * none

access to dn="cn=HWAdmin,ou=Hardware,o=Intern"
       by * read

access to dn="ou=Hardware,o=Intern"
       by dn="cn=HWAdmin,ou=Hardware,o=Intern" write
       by * read

access to attr=userPassword
       by self write
       by * none

access to attr=telephoneNumber
       by self write
       by * read

access to dn="cn=Admin,o=Intern"
       by * none

access to *
       by * read

root@host1:~ #
```

5.1.3 Erstellen der Datenbank

Als Datenbank kann zunächst die bereits in diesem Buch vorgestellte Organisation `Intern` verwendet werden. Um im weiteren Verlauf die Hardware-Informationen in Objekten abzulegen, muss das LDAP-Schema erweitert werden. Dazu definieren wir eine Objektklasse `host`. Sie ist in die Datei `/etc/openldap/slapd.oc.conf` einzutragen:

```
objectclass host
        requires
                objectClass
        allows
                cn,
                cpumodel,
                cpuspeed,
                ramsize,
                swapsize,
                hdamodel,
                hdbmodel,
                description
```

Der Typ der Objektklasse ist als Eigenschaft grundsätzlich anzugeben. Ferner ist es möglich, neben dem Common Name (CN), das Modell und die Geschwindigkeit der CPU (`cpumodel`, `cpuspeed`) aufzuführen. Die Gesamtgröße der Speichermodule kann in der Eigenschaft `ramsize`, die Größe des Swap-Bereichs in `swapsize` festgehalten werden. Die Attribute `hdamodel` und `hdbmodel` beschreiben das Modell der ersten und der zweiten IDE-Platte. Die Hardware-Inventarisierung beschränkt sich somit auf diese leicht zu erweiternde Attributliste. Abschließend wird über `description` eine Beschreibung des Objekts angegeben. Die Attribute müssen nicht weiter spezifiziert werden, da sie standardmäßig vom Typ `cis` (Case Ignore String) sind.

Nachdem das Schema erweitert wurde, kann nun der LDAP-Server gestartet werden. Der generelle Start des LDAP-Servers beim Systemstart wird bei einer SuSE-Distribution erreicht, indem mittels `yast` die Variable `START_LDAP` auf `yes` gesetzt wird.

```
root@host1:~ # /etc/rc.d/init.d/ldap start
Starting ldap-server.                                          done
root@host1:~ #
```

Nachdem der LDAP-Server konfiguriert wurde und aktiv ist, können mit dem Kommando `ldapadd` der Container `Hardware` und das Objekt `HWAdmin` erstellt werden. Dazu ist eine Datei im LDIF-Format zu erstellen, die die beiden Objekte beschreibt:

```
root@host1:~ # cat hardware.ldif
dn: ou=Hardware, o=Intern
objectclass: organizationalUnit
ou: Hardware
description: "Hardwareinventarisierung"

dn: cn=HWAdmin, ou=Hardware, o=Intern
objectclass: person
cn: HWAdmin
sn: HWAdmin
userpassword: hardware
description: "Hardware Administrator"

root@host1:~ #
```

Anschließend können die in der LDIF-Datei genannten Objekte mit dem Kommando `ldapadd` in der Datenbank erstellt werden:

```
root@host1:~ # ldapadd -D "cn=Admin, o=Intern" -w linux \
# -f hardware.ldif
adding new entry ou=Hardware, o=Intern

adding new entry cn=HWAdmin, ou=Hardware, o=Intern

root@host1:~ #
```

Somit hat der Directory Information Tree nun das in Abbildung 5.2 gezeigte Aussehen.

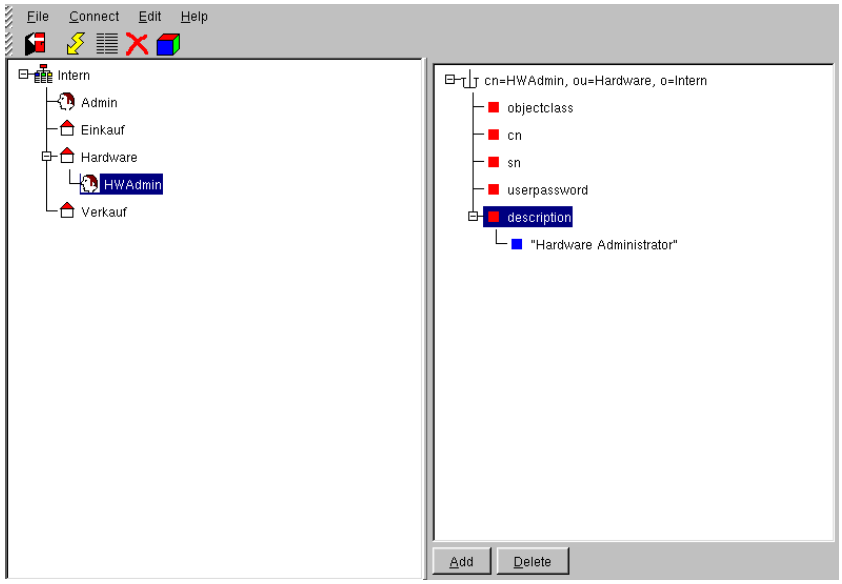

Abbildung 5.2: `Hardware` und `HWAdmin`

5.1.4 Konfiguration des Clients

Nachdem der LDAP-Server für die gewünschten Funktionen vorbereitet ist, muss der Client dahin gehend eingerichtet werden, dass er die Informationen über die Hardware zusammenträgt und sie dann beim Systemstart in der LDAP-Datenbank ablegt.

Im ersten Schritt sind die Rechner im Netzwerk, deren Hardware aufgenommen werden soll, als LDAP-Clients einzurichten. Dazu kann man die folgende Konfigurationsdatei `ldap.conf` verwenden:

```
root@host2:~ # cat /etc/openldap/ldap.conf
#
# Konfigurationsdatei des LDAP-Clients
#
# ldap.conf
#
# Erstellt von Jens Banning
#

BASE    o=Intern
HOST    host1.intern
```

```
PORT    389

SIZELIMIT       12
TIMELIMIT       15
DEREF           never

root@host2:~ #
```

Sie definiert den LDAP-Server, an den später die Daten gesendet werden sollen.

Informationen über die zugrunde liegende Hardware eines jeden Linux-Systems befinden sich im virtuellen Dateisystem `proc`. Dieses Dateisystem ist in das Verzeichnis `/proc/` eingebunden und enthält ein Abbild des Linux-Kernels. Der Kern des Betriebssystems analysiert beim Booten die Hardware-Komponenten. Die Ergebnisse dieses Vorgangs sind später im `proc`-Dateisystem zu finden. Beispielsweise befindet sich das Ergebnis der CPU-Prüfung in der Datei `/proc/cpuinfo`:

```
root@host2:~ # cat /proc/cpuinfo
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 5
model           : 2
model name      : Pentium 75 - 200
stepping        : 12
cpu MHz         : 121.707
fdiv_bug        : no
hlt_bug         : no
sep_bug         : no
f00f_bug        : yes
coma_bug        : no
fpu             : yes
fpu_exception   : yes
cpuid level     : 1
wp              : yes
flags           : fpu vme de pse tsc msr mce cx8
bogomips        : 242.48
root@host2:~ #
```

Informationen über den Hauptspeicher des Systems sind in `/proc/meminfo` zu finden:

```
root@host2:~ # cat /proc/meminfo
        total:    used:    free:  shared: buffers:  cached:
Mem:  49913856 48250880  1662976        0  9576448 25931776
Swap: 157401088   241664 157159424
MemTotal:     48744 kB
MemFree:       1624 kB
MemShared:        0 kB
```

```
Buffers:        9352 kB
Cached:        25324 kB
BigTotal:          0 kB
BigFree:           0 kB
SwapTotal:    153712 kB
SwapFree:     153476 kB
root@host2:~ #
```

Neben dem Speicher und der CPU soll auch das Modell der ersten beiden IDE-Festplatten angegeben werden. Bei Nicht-IDE-Systemen erfällt dieser Bereich. Die IDE-Modelle der Festplatten werden festgehalten in den Dateien `/proc/ide/hda/model` und `/proc/ide/hdb/model`:

```
root@host2:~ # cat /proc/ide/hda/model
QUANTUM FIREBALL_TM1280A
root@host2:~ # cat /proc/ide/hdb/model
FX400E
root@host2:~ #
```

Auf diese Weise sind im `proc`-Dateisystem alle relevanten Informationen zu finden, wodurch sich die Hardware-Inventarisierung leicht erweitern lässt.

Aus den oben genannten Dateien werden nun mit den Mitteln der Shell-Programmierung die gewünschten Informationen extrahiert. Die so gewonnenen Werte werden dann im Verzeichnis `/tmp/` in Form einer LDIF-Datei abgespeichert. Sie wird anschließend verwendet, um das entsprechende Objekt im Verzeichnisdienst zu erstellen.

So ist es zunächst notwendig, ein Shell-Skript zu erstellen, das die gewünschte Funktionalität ausführt. Damit es später beim Systemstart verwendet werden kann, wird es im Verzeichnis `/etc/rc.d/init.d/` abgespeichert, wobei der Pfad mitunter distributionsabhängig ist. Das im Weiteren vorgestellte Skript trägt den Namen `hw2ldap` und wird anschließend detailliert beschrieben:

```
root@host2:/etc/rc.d/init.d # cat hw2ldap
#!/bin/bash
#
# Hardware-Inventarisierung -> LDAP
#
# hw2ldap
#
# Erstellt von Jens Banning
#

#
# CPU-Info (Modell und MHz).
#
```

```
cpumodel=‘cat /proc/cpuinfo|head -5|tail -1|cut -d ":" -f 2‘
cpuspeed=‘cat /proc/cpuinfo|head -7|tail -1|cut -d ":" -f 2‘

#
# MEM-Info (RAM und SWAP).
#

ramsize=‘cat /proc/meminfo|head -4 |tail -1|cut -d ":" -f 2‘
swapsize=‘cat /proc/meminfo|head -11|tail -1|cut -d ":" -f 2‘

#
# Festplatten-Info (Modell hda und hdb).
#

hdamodel=‘cat /proc/ide/hda/model‘
hdbmodel=‘cat /proc/ide/hdb/model‘

#
# LDIF-Datei schreiben.
#

echo "dn: cn=$HOSTNAME, ou=Hardware, o=Intern" \
                                  >/tmp/hw2ldap.ldif
echo "objectclass: host"         >>/tmp/hw2ldap.ldif
echo "cn: $HOSTNAME"             >>/tmp/hw2ldap.ldif
echo "cpumodel: $cpumodel"       >>/tmp/hw2ldap.ldif
echo "cpuspeed: $cpuspeed MHZ"   >>/tmp/hw2ldap.ldif
echo "ramsize: $ramsize"         >>/tmp/hw2ldap.ldif
echo "swapsize: $swapsize"       >>/tmp/hw2ldap.ldif
echo "hdamodel: $hdamodel"       >>/tmp/hw2ldap.ldif
echo "hdbmodel: $hdbmodel"       >>/tmp/hw2ldap.ldif
echo "description: Host with name $HOSTNAME" >>/tmp/hw2ldap.ldif

#
# Alte Hardware-Info loeschen.
# Neue Hardware-Info erstellen.
#

ldapdelete -D "cn=HWAdmin, ou=Hardware, o=Intern" \
           -w hardware "cn=$HOSTNAME, ou=Hardware, o=Intern" \
           &>/dev/null


ldapadd  -D "cn=HWAdmin, ou=Hardware, o=Intern" -w hardware \
         -f /tmp/hw2ldap.ldif

root@host2:/etc/rc.d/init.d #
```

Im Skript werden die gewünschten Informationen aus den Dateien im Verzeichnis `/proc/` extrahiert und zunächst in Variablen abgespeichert. Dazu werden die Linux-Kommandos `head`, `tail`, `cat` und `cut` verwendet. Betrachten wir dazu die folgende Zeile etwas näher.

```
cpumodel=‘cat /proc/cpuinfo|head -5|tail -1|cut -d ":" -f 2‘
```

Hier wird die Datei `/proc/cpuinfo` mit dem Befehl `cat` ausgegeben. Anschließend werden diese Ausgaben über den Pipe-Mechanismus als Eingaben des Aufrufs `head -5` verwendet. Er betrachtet lediglich die ersten fünf Zeilen. Da das Modell der CPU in der fünften Zeile zu finden, wird von der `head`-Ausgabe mit `tail -1` die letzte Zeile extrahiert. Das Ergebnis dieser drei verschachtelten Kommandos lautet:

```
model name      : Pentium 75 - 200
```

Von dieser Ausgabe ist lediglich der Wert hinter dem Doppelpunkt von Interesse. Das Kommando `cut` schneidet ihn aus, indem es in der Annahme, dass das Spaltentrennzeichen der Doppelpunkt ist, das zweite Feld betrachtet. Am Ende wird der Shell-Variablen `cpumodel` der Wert `Pentium 75-200` zugewiesen.

Nachdem neben dem Modell der CPU die weiteren Hardware-Informationen in Shell-Variablen abgespeichert worden sind, können sie nun zu einer Datei vom Format LDIF zusammengestellt werden. Dazu werden die einzelnen Zeilen mit dem Kommando `echo` erstellt, wobei die Variable `HOSTNAME` den Namen des Linux-Rechners enthält. Dies geschieht, indem die Ausgaben in die Datei `/tmp/hw2ldap.ldif` umgeleitet werden. Somit ergibt sich folgende Datei:

```
root@host2:~ # cat /tmp/hw2ldap.ldif
dn: cn=host2, ou=Hardware, o=Intern
objectclass: host
cn: host2
cpumodel: Pentium 75 - 200
cpuspeed: 121.707 MHZ
ramsize: 48744 kB
swapsize: 153712 kB
hdamodel: QUANTUM FIREBALL_TM1280A
hdbmodel: FX400E
description: Host with name host2
root@host2:~ #
```

Die Hardware-Inventarisierung ist nun abgeschlossen, und es ist im Folgenden notwendig, die Informationen der LDIF-Datei im Verzeichnisdienst abzulegen. Falls bereits ein Objekt des entsprechenden Rechners im Container `Hardware` existiert, so wird es zunächst gelöscht. Damit das Löschen durchgeführt werden kann, erfolgt eine Anmeldung am Directory Service (DS) mit dem Benutzer `HWAdmin`. Da das Kommando `ldapdelete` eine Fehlermeldung ausgibt, falls das zu löschende Objekt nicht existiert, werden die Ausgaben nicht angezeigt und verworfen (`&>/dev/null`).

Schließlich erstellt `ldapadd` unter Verwendung der zuvor generierten LDIF-Datei das Host-Objekt in der LDAP-Datenbank.

Damit das Shell-Skript beim Rechnerstart aktiviert wird, reicht es nicht aus, es in dem oben genannten Verzeichnis abzulegen. Zusätzlich ist in dem Verzeichnis des Standard-Runlevels ein symbolischer Link zu erstellen, der auf das Skript zeigt. Sein Name muss mit dem Buchstaben `S` beginnen, woran das System erkennt, dass das Skript, auf das der Link zeigt, beim Betreten des Runlevels gestartet werden soll. Es ist natürlich notwendig, dass die Hardware-Inventarisierung erst durchgeführt wird, wenn bereits das Netzwerk aktiv ist. Deshalb folgt im Namen des Links hinter dem `S` eine zweistellige Zahl, die festlegt, wann das Skript gestartet wird. Um es zum Beispiel am Ende aufzurufen, muss der Link mit `S99` beginnen.

Falls der Standard-Runlevel, der beim Systemstart betreten wird, `2` ist, so befinden sich die zugehörigen Links im Verzeichnis `/etc/rc.d/init.d/rc2.d/`. Das Anlegen eines symbolischen Links erfolgt mit dem Kommando `ln` unter Verwendung der Option `-s`:

```
root@host2:/etc/rc.d/init.d/rc2.d # ln -s ../hw2ldap S99hw2ldap
root@host2:/etc/rc.d/init.d/rc2.d # ls -l S99hw2ldap
lrwxrwxrwx 1 root root 10 Dec  1 08:06 S99hw2ldap -> ../hw2ldap
root@host2:/etc/rc.d/init.d/rc2.d #
```

Weitere Informationen zu der Funktionsweise der Linux-Runlevel finden Sie in den Büchern [4] und [5].

5.1.5 Aktualisieren der Datenbank

Nachdem das Skript nun erstellt und in die Konfiguration des Runlevels 2 eingebunden wurde, wird es bei jedem Start des Rechners im Runlevel 2 ausgeführt. Somit erfolgt die Hardware-Inventarisierung bei jedem Boot-Vorgang, wodurch die Informationen im Verzeichnisdienst die technischen Gegebenheiten immer korrekt wiedergeben.

Im Folgenden sind die letzten Ausgaben abgedruckt, die während des Bootvorgangs angezeigt werden.

```
...
Starting RPC portmap daemon                                done
Starting syslog services                                   done
Loading keymap qwertz/banning.map.gz                       done
Initializing random number generator                       done
Starting service at daemon:                                done
Starting console mouse support (gpm):                      done
Starting INET services (inetd)                             done
Starting lpd                                               done
```

```
Initializing SMTP port. (sendmail)                              done
Starting SSH daemon:                                            done
Starting CRON daemon                                            done
Starting identd                                                 done
Starting Name Service Cache Daemon                              done
adding new entry cn=host2, ou=Hardware, o=Intern

Master Resource Control: runlevel 2 has been              reached


Welcome to SuSE Linux 7.0 (i386) - Kernel 2.2.16 (tty1).

host2 login:
```

Sie zeigen, dass ein neues Objekt mit dem Namen des Rechners im Container `Hardware` erstellt wurde.

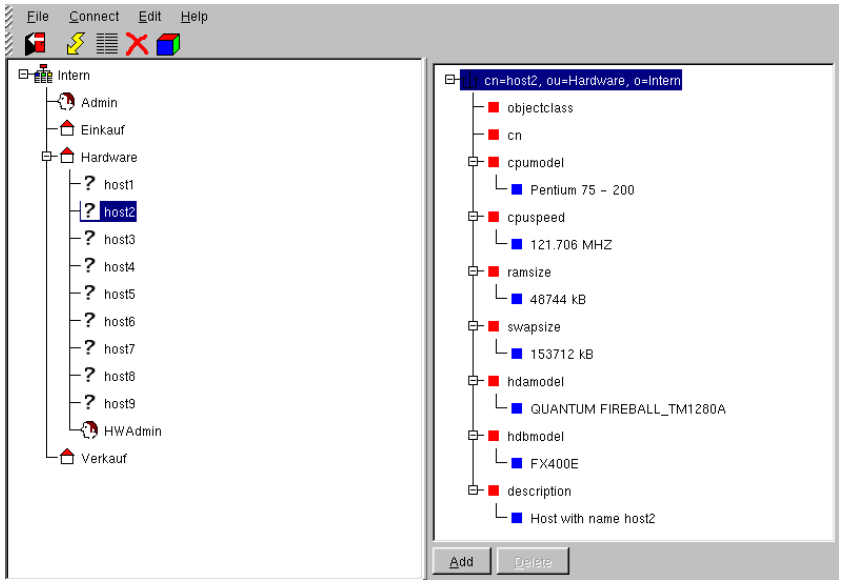

Abbildung 5.3: `host`-Objekte im Verzeichnisdienst

Die Hardware-Informationen der so erstellten Objekte können anschließend mit den bekannten Anwendungen eingesehen werden (siehe Abbildung 5.3). Neben dem grafischen Programm `kldap` ist der Zugriff natürlich auch auf der Kommandoebene zum Beispiel mit `ldapsearch` möglich.

5.1.6 Zugriff über das Web-Gateway

Um den Zugriff auf die Hardware-Informationen der Rechner plattformübergreifend zur Verfügung zu stellen, kann das Web-Gateway eingesetzt werden. Es bietet dem Anwender die Möglichkeit, leicht und komfortabel über einen Internet-Browser seiner Wahl auf die Daten des Verzeichnisdienstes zuzugreifen und in ihnen zu suchen.

Zum Einsatz des Gateways ist es notwendig, die Zeile

```
web500gw  8888/tcp        # Web500gw-Gateway
```

in die Datei `/etc/services` einzutragen. Ferner muss in der Datei `/etc/inetd.conf` eingestellt werden, dass das Gateway bei eintreffenden Anfragen gestartet werden soll:

```
web500gw stream tcp nowait nobody /usr/sbin/web500gw web500gw -I
```

Das Web-Gateway wird seinerseits über eine eigene Konfigurationsdatei eingerichtet. Sie befindet sich im Verzeichnis `/etc/web500/` und trägt den Namen `web500gw.conf`:

```
root@host1:~ # cat /etc/web500/web500gw.conf
#
# Konfigurationsdatei des Web-Gateways
#
# web500gw.conf
#
# Erstellt von Jens Banning
#

#
# Allgemeines
#

ldapserver:     host1.intern
ldapport:       389

port:           8888
otherservers:   yes

timelimit:      240
sizelimit:      0

homedn:         o=Intern
rootishome:     on

web500dn:
web500pw:
```

```
syslog:          yes
logformat:       %h "%r" %s %e "%a" %b %l "%f" %T %x

showonematch:    yes
ufnsearch:       on

subsearch:       organization, organizationalUnit
lastmodified:    off

etcdir:          /etc/web500

#
# Rechte
#

access:          World : .* : read : 50 : en : o=Intern : : :

#
# Sprachen
#

language:        en : .* :

#
# Browser
#

browser:         All : .* : html32 : table : top,menu
root@host1:~ #
```

Bevor hier der Zugriff auf die Daten des Verzeichnisdienstes über das Web-Gateway dargestellt werden kann, ist jedoch noch ein weiterer Konfigurationsschritt notwendig. In der Datei `ldaptemplates.conf` wird festgehalten, mit welchen umgangssprachlichen Namen Objekte und deren Eigenschaften beschrieben werden sollen. Da in diesem Beispiel eine zusätzliche Objektklasse `host` mit diversen Attributen eingerichtet wurde, müssen für sie noch die zu verwendenden Templates definiert werden. Diese Definition ist daher in der Datei `ldaptemplates.conf` im Verzeichnis `/etc/web500/` zu ergänzen:

```
# Host template
"Host"
"Hosts"
"host icon"
# template options
""
# objectclass list
host
END
```

```
# name of attribute to authenticate as
""
# default attribute name to use when forming RDN of a new entry
cn
# default location when adding new entries
""
# rules used to define default values for new entries
END
# list of items for display
item cis         "CPU Speed"     cpuspeed
item cis         "CPU Model"     cpumodel
item cis         "RAM Size"      ramsize
item cis         "SWAP Size"     swapsize
item cis         "HDA Model"     hdamodel
item cis         "HDB Model"     hdbmodel
item cis         "Description"   description
END
```

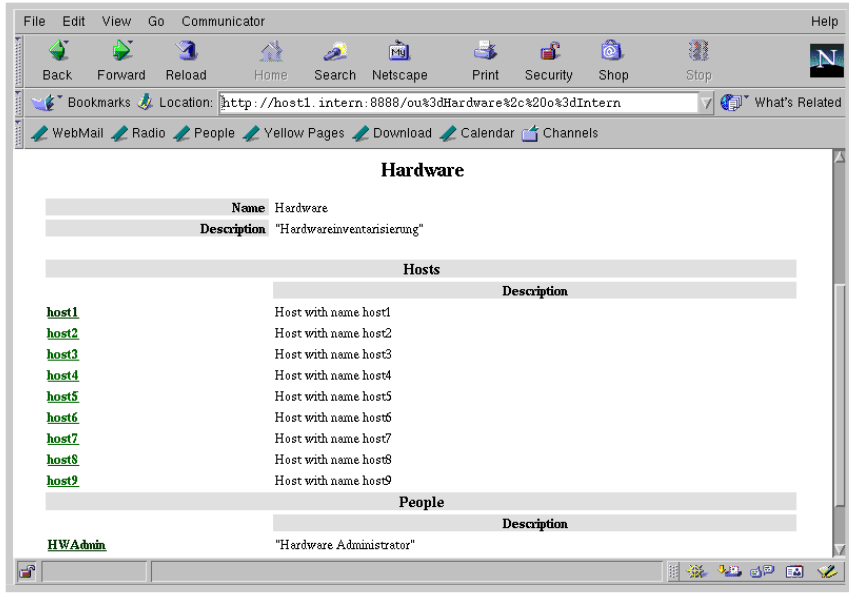

Abbildung 5.4: Web-Gateway zur Hardware-Inventarisierung

Der Zugriff auf das Gateway stellt sich dann wie in Abbildung 5.4 dar, in der bereits in den Container `Hardware` verzweigt wurde. Durch die Anwahl des gewünschten Objekts werden anschließend dessen Eigenschaften angezeigt (siehe Abbildung 5.5).

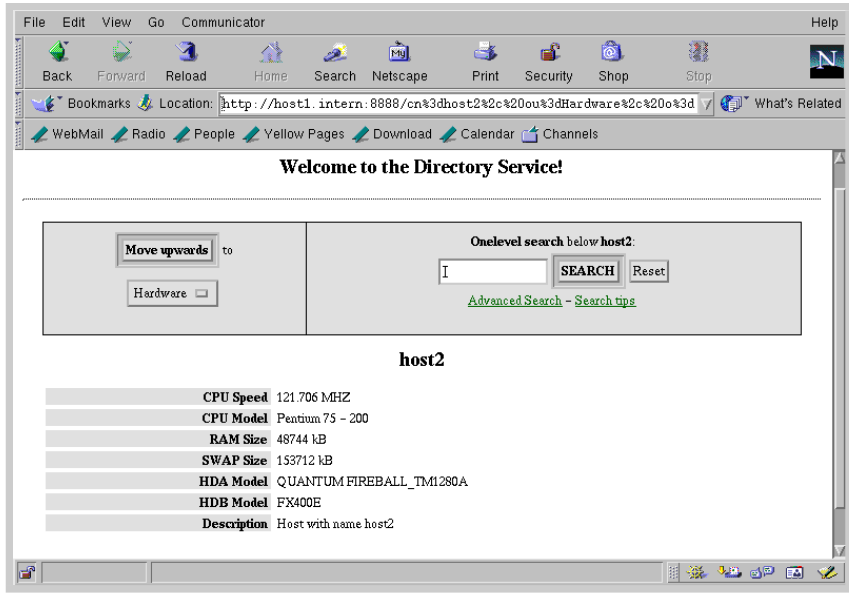

Abbildung 5.5: Web-Gateway zur Hardware-Inventarisierung

5.2 NFS-Verwaltung

Das Network File System (NFS) bietet Rechnern im Netzwerk die Möglichkeit, Teile ihres Verzeichnisbaums anderen Hosts zur Verfügung zu stellen. Die Konfiguration der NFS-Server erfolgt über die Datei `exports` im Verzeichnis `/etc/`. Am NFS-Client können dann die auf diese Weise zur Verfügung gestellten Verzeichnisse mit dem Befehl `mount` eingebunden werden.

Im Folgenden wird die zentrale Verwaltung der NFS-Server im Netzwerk per LDAP beschrieben. Weitere Informationen zu NFS und dem NFS-Client finden Sie in [6].

5.2.1 Funktion

Die NFS-Verwaltung erfolgt zunächst dadurch, dass im Verzeichnisdienst Objekte erstellt werden, die zu jedem gewünschten Host die Verzeichnisse beschreiben, die er anderen zur Verfügung stellen soll. Die Objekte können vom Administrator (`Admin`) erstellt und verändert werden. Die Position der NFS-Objekte im Directory Service hängt von den Bedürfnissen des Administrators ab. Im Beispiel werden sie in einem separaten Container abgespeichert. Er trägt den Namen `NFS` und existiert auf der gleichen Ebene wie `Einkauf` und `Verkauf` in der Organisation `Intern` (siehe Abbildung 5.6).

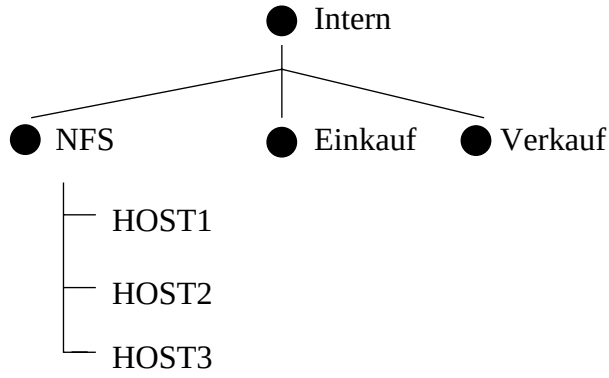

Abbildung 5.6: NFS-Verwaltung unter `Intern`

Nachdem die Objekte im DIT vorhanden sind, müssen sie nun im weiteren Verlauf zur Konfiguration der NFS-Server verwendet werden. Dazu wird das NFS-Startskript aus dem Verzeichnis `/etc/rc.d/init.d/` dahin gehend verändert, dass es die Informationen aus der LDAP-Datenbank verwendet. Mit den so gewonnenen Daten wird die Datei `/etc/exports` erstellt, die der NFS-Server anschließend verwendet.

5.2.2 Konfiguration des Servers

Um den LDAP-Server zu starten, kann die Konfigurationsdatei verwendet werden, die bereits im letzten Kapitel benutzt wurde:

```
root@host1:~ # cat /etc/openldap/slapd.conf
#
# Konfigurationsdatei des LDAP-Servers
#
# slapd.conf
#
# Erstellt von Jens Banning
#

#
# Allgemeines
#

include         /etc/openldap/slapd.at.conf
include         /etc/openldap/slapd.oc.conf

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

loglevel        256
```

```
schemacheck     on

sizelimit       500
timelimit       3600

#
# Datenbank
#

database        ldbm
lastmod         off

cachesize       1000
dbcachesize     100000

directory       /var/lib/ldap
suffix          "o=Intern"

rootdn          "cn=Admin, o=Intern"
rootpw          {CRYPT}zsOQLoUVZvS6E

#
# Rechte
#

defaultaccess   read

access to attr=userPassword
       by self write
       by * none

access to attr=telephoneNumber
       by self write
       by * read

access to dn="cn=Admin,o=Intern"
       by * none

access to *
       by * read

root@host1:~ #
```

In einem neu zu erstellenden Container NFS haben somit alle Anwender lesende Rechte. Lediglich das Admin-Objekt kann die Daten verändern.

5.2.3 Erstellen der Datenbank

In der LDAP-Datenbank befinden sich die aus dem letzten Kapitel bekannten Objekte der Organisation `Intern`, die in einen `Einkauf` und einen `Verkauf` unterteilt ist. Damit der Administrator Objekte für die NFS-Verwaltung erstellen kann, ist zunächst das Schema des Verzeichnisbaumes um eine neue Objektklasse zu ergänzen. Sie erhält den Namen `NFShost` und wird in der Datei `/etc/openldap/slapd.oc.conf` eingetragen:

```
objectclass NFShost
        requires
                objectClass
        allows
                cn,
                export,
                description
```

Die Objektklasse besteht im Prinzip aus vier Attributen. Neben der `objectClass` ist der Common Name (`cn`) des Objekts anzugeben. Mit der Eigenschaft `export` wird ein Verzeichnis beschrieben, das der NFS-Server, der durch das Objekt definiert wird, anderen Rechnern im Netzwerk zur Verfügung stellen soll. Da `export` mehrere Werte annehmen kann, ist es somit möglich, auch mehrere Verzeichnisse zu beschreiben. Letztlich kann über `description` eine Beschreibung des Objekts angegeben werden.

Unter Linux wird bei Datei- und Verzeichnisnamen generell zwischen Groß- und Kleinschreibung unterschieden. Somit muss man genau prüfen, wie die Werte des Attributs `exports` geschrieben sind, da es möglich ist, dass die Verzeichnisse `/Daten/` und `/daten` parallel existieren können. Damit diese Unterscheidung im LDAP-Schema realisiert wird, muss man die Eigenschaft `export` in der Datei `/etc/openldap/slapd.at.conf` als Case Exact String eintragen:

```
attribute       export                                  ces
```

Da bei den anderen beiden Attributen die Schreibweise unerheblich ist, werden sie als Case Ignore String betrachtet, was grundsätzlich als Defaultwert angenommen wird.

Der LDAP-Server kann nun durch den Aufruf des entsprechenden Skripts gestartet werden.

```
root@host1:~ # /etc/rc.d/init.d/ldap start
Starting ldap-server.                                   done
root@host1:~ #
```

Die LDAP-Datenbank ist nun aktiv, und das Schema ist entsprechend erweitert worden. Als Erstes soll der Container `NFS` erstellt werden. Dazu ist zunächst eine Datei

im LDIF-Format zu erstellen, die anschließend vom Kommando `ldapadd` verwendet wird:

```
root@host1:~ # cat nfs.ldif
dn: ou=NFS, o=Intern
objectclass: organizationalUnit
ou: NFS
description: "NFS-Verwaltung"

root@host1:~ # ldapadd -D "cn=Admin, o=Intern" -w linux \
# -f nfs.ldif
adding new entry ou=NFS, o=Intern

root@host1:~ #
```

Die Rechner, deren NFS-Dienste per LDAP verwaltet werden sollen, sind nun als Objekte der Klasse `NFShost` im Verzeichnisdienst anzulegen. Als Objektname empfiehlt sich der Hostname des entsprechenden Teilnehmers im Netzwerk. In diesem Beispiel werden drei NFS-Server definiert, die jeweils zwei Verzeichnisse all den Rechnern zur Verfügung stellen, deren Fully Qualified Domain Name (FQDN) mit `.intern` endet:

```
root@host1:~ # cat nfsdaten.ldif
dn: cn=host1, ou=NFS, o=Intern
objectclass: nfshost
cn: host1
export: "/home  *.intern(rw)"
export: "/cdrom *.intern(ro)"
description: "NFS-Server host1"

dn: cn=host2, ou=NFS, o=Intern
objectclass: nfshost
cn: host2
export: "/daten *.intern(rw)"
export: "/cdrom *.intern(ro)"
description: "NFS-Server host2"

dn: cn=host3, ou=NFS, o=Intern
objectclass: nfshost
cn: host3
export: "/tmp   *.intern(rw)"
export: "/cdrom *.intern(ro)"
description: "NFS-Server host3"

root@host1:~ # ldapadd -D "cn=Admin, o=Intern" -w linux \
# -f nfsdaten.ldif
adding new entry cn=host1, ou=NFS, o=Intern
```

```
adding new entry cn=host2, ou=NFS, o=Intern

adding new entry cn=host3, ou=NFS, o=Intern

root@host1:~ #
```

So soll der Rechner `host2` zum Beispiel sein Verzeichnis `/cdrom/` allen Rechnern aus der Domäne `intern` lesend (read only: `ro`) zur Verfügung stellen. Auf das Verzeichnis `/daten/` sollen auch schreibende Zugriffe (read write: `rw`) möglich sein.

Die Datenbank des Verzeichnisdienstes sieht somit aus, wie in Abbildung 5.7 gezeigt.

Abbildung 5.7: Objekte unter `NFS`

5.2.4 Datenbankpflege mit `kldap`

Um in der Datenbank nun weitere Informationen einzutragen, können die bekannten Kommandos und Anwendungen benutzt werden. Die grafische Pflege der LDAP-Daten kann mit dem Programm `kldap` erfolgen. Es wird in der grafischen Oberfläche KDE gestartet. Auf diese Weise können Werte gelöscht und hinzugefügt werden (siehe Abbildung 5.8).

Lediglich der Administrator des Verzeichnisdienstes hat die notwendigen Rechte, um die Objekte zu verändern (siehe Abbildung 5.9).

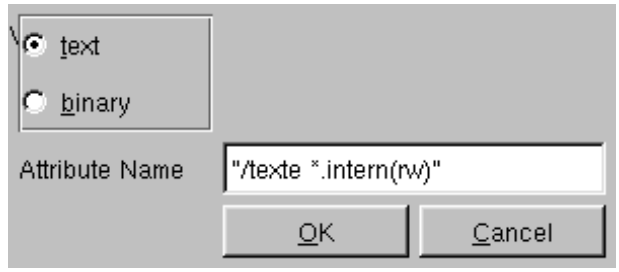

Abbildung 5.8: Hinzufügen eines `export`-Wertes

Abbildung 5.9: Objekte unter `NFS`

5.2.5 Konfiguration des Clients

Die Konfiguration der LDAP-Clients, auf denen der NFS-Server gestartet werden soll, besteht aus mehreren Schritten. Als Erstes ist die Konfigurationsdatei `ldap.conf` so einzurichten, dass sich die Anfragen des LDAP-Clients an den gewünschten LDAP-Server richten:

```
root@host2:~ # cat /etc/openldap/ldap.conf
#
# Konfigurationsdatei des LDAP-Clients
#
# ldap.conf
```

```
#
# Erstellt von Jens Banning
#

BASE    o=Intern
HOST    host1.intern
PORT    389

SIZELIMIT       12
TIMELIMIT       15
DEREF           never

root@host2:~ #
```

Im zweiten Schritt muss nun die in der LDAP-Datenbank abgespeicherte NFS-Konfiguration umgesetzt werden. Dazu betrachten wir zunächst das Startskript des NFS-Servers. Es befindet sich im Verzeichnis `/etc/rc.d/init.d/` und hat bei einer SuSE-Distribution den Namen `nfsserver`. Der folgende Ausschnitt zeigt lediglich den Teil des Skripts, der beim Starten des NFS-Servers aufgerufen wird:

```
case "$1" in
  start)
    PARAMS=""
    test "$REEXPORT_NFS" = yes && PARAMS="--re-export"

    echo -n "Starting NFS server"
    startproc /usr/sbin/rpc.mountd $PARAMS  || return=$rc_failed
    startproc /usr/sbin/rpc.nfsd   $PARAMS  || return=$rc_failed
    echo -e "$return"
    ;;
```

Zur Konfiguration wird die Datei `/etc/exports` verwendet. Dieses Skript ist nun dahin gehend zu ändern, dass vor dem Start des NFS-Servers die Informationen aus der LDAP-Datenbank gelesen werden. Der modifizierte Teil der Startprozedur ist im Folgenden dargestellt und wird anschließend beschrieben:

```
case "$1" in
  start)
    PARAMS=""
    test "$REEXPORT_NFS" = yes && PARAMS="--re-export"

    echo -n "Starting NFS server (LDAP)"

    #
    # LDAP-Server befragen und anschliessend
    # die Konfigurationsdatei schreiben.
    #
```

```
anzahlzeilen=‘ldapsearch -b "ou=NFS, o=Intern" \
              "cn=$HOSTNAME" export|wc -l‘

anzahlexports=$[$anzahlzeilen-1]

ldapsearch -b "ou=NFS, o=Intern" "cn=$HOSTNAME" export \
      |tail -$anzahlexports|cut -d ’"’ -f 2 >/etc/exports

#

startproc /usr/sbin/rpc.mountd $PARAMS  || return=$rc_failed
startproc /usr/sbin/rpc.nfsd   $PARAMS  || return=$rc_failed
echo -e "$return"
;;
```

Es wurden drei Zeilen eingefügt. Zunächst wird `ldapsearch` aufgerufen. Im Container `NFS` wird das Objekt gesucht, dessen Common Name dem Hostnamen des LDAP-Clients entspricht. Es werden jedoch lediglich die Werte der `export`-Eigenschaft ausgegeben. Die Ausgabe hat demnach die Form:

```
cn=host2, ou=NFS, o=Intern
export="/daten *.intern(rw)"
export="/cdrom *.intern(ro)"
export="/texte *.intern(rw)"
```

Es ist zu erkennen, dass in der ersten Zeile der Distinguished Name des Objekts aufgeführt ist. Da er jedoch ignoriert werden muss, wird die erste Zeile übersprungen. Dazu wird zunächst durch den Aufruf von `wc -l` gezählt, wie viele Zeilen die Ausgabe von `ldapsearch` umfasst. Im Beispiel sind dies vier Zeilen. Das Ergebnis wird in der Variablen `anzahlzeilen` abgelegt. Anschließend wird sie um eins vermindert. Somit ist in der Variablen `anzahlexports` der Wert 3 abgespeichert.

Im letzten Schritt wird nun nach der Konfiguration des Hosts im NFS-Container gesucht, wobei lediglich die letzten `anzahlexports` (im Beispiel drei) Zeilen betrachtet werden. Ferner wird mit dem Befehl `cut` nur der Text zwischen den Anführungszeichen betrachtet:

```
/daten *.intern(rw)
/cdrom *.intern(ro)
/texte *.intern(rw)
```

Die Zeilen werden durch die Ausgabeumleitung in die Datei `/etc/exports` geschrieben. Anschließend wird der NFS-Server mit den per LDAP erlangten Einstellungen aktiviert:

```
root@host2:~ # cat /etc/exports
/daten *.intern(rw)
/cdrom *.intern(ro)
```

```
/texte *.intern(rw)
root@host2:~ #
```

Der Start des NFS-Servers erfolgt dadurch, dass dem Skript `nfsserver` im Verzeichnis `/etc/rc.d/init.d/` der Parameter `start` übergeben wird:

```
root@host2:~ # /etc/rc.d/init.d/nfsserver start
Starting NFS server (LDAP)                                  done
root@host2:~ #
```

Damit der LDAP-Client den NFS-Server bei jedem Systemstart aktiviert, muss im gewünschten Runlevel ein entsprechender Link auf das Startskript erstellt werden. Bei einer SuSE-Distribution ist dieser Link bereits vorhanden. Mit dem Programm `yast` muss jedoch die SuSE-spezifische Variable `NFS_SERVER` auf `yes` gesetzt werden.

Bei jedem Neustart des gesamten Linux-Systems wird der NFS-Dienst dann mit den in der LDAP-Datenbank definierten Werten gestartet:

```
Starting service at daemon:                                 done
Starting console mouse support (gpm):                       done
Starting INET services (inetd)                              done
Starting lpd                                                done
Initializing SMTP port. (sendmail)                          done
Starting SSH daemon:                                        done
Starting CRON daemon                                        done
Starting identd                                             done
Starting Name Service Cache Daemon                          done
Starting NFS server (LDAP)                                  done
Master Resource Control: runlevel 2 has been              reached


Welcome to SuSE Linux 7.0 (i386) - Kernel 2.2.16 (tty1).

host1 login:
```

Nachdem der LDAP-Client nun als NFS-Server konfiguriert wurde, können die per NFS zur Verfügung gestellten Verzeichnisse von anderen Rechnern im Netzwerk in deren Verzeichnisbaum eingebunden werden. Dazu existieren zwei verschiedene Möglichkeiten:

1. Zum einen kann ein NFS-Verzeichnis mit dem Befehl `mount` an das lokale Dateisystem angehängt werden. `mount` kann vom Systemadministrator gestartet werden. Als Dateisystemtyp (`-t`) ist `nfs` zu verwenden:

   ```
   root@host4:~ # mount -t nfs host2:/daten /daten
   ```

   Auf diese Weise wird das Verzeichnis `/daten/` des Rechners `host2` in den Verzeichnisbaum von `host4` eingebunden. Erfolgt am NFS-Client ein Zugriff auf Daten im Verzeichnis `/daten/`, so wird damit der Rechner `host2` angesprochen.

2. Der Befehl `mount` führt nur zu einer temporären Verbindung zwischen NFS-Server und -Client. Spätestens nach dem Neustart des Systems am NFS-Client ist der Zugriff auf den Server nicht mehr möglich. Daher ist es zum anderen sinnvoll, die Einstellung in der zentralen Konfigurationsdatei der Dateisysteme aufzunehmen. Die Datei befindet sich im Verzeichnis `/etc/` und hat den Namen `fstab`. Der obige `mount`-Befehl wird durch die folgende Zeile permanent ausgeführt:

```
host2:/cdrom    /cdrom    nfs    defaults    0    0
```

Abbildung 5.10 zeigt den Zusammenhang zwischen LDAP-Server und -Client sowie NFS-Server und -Client.

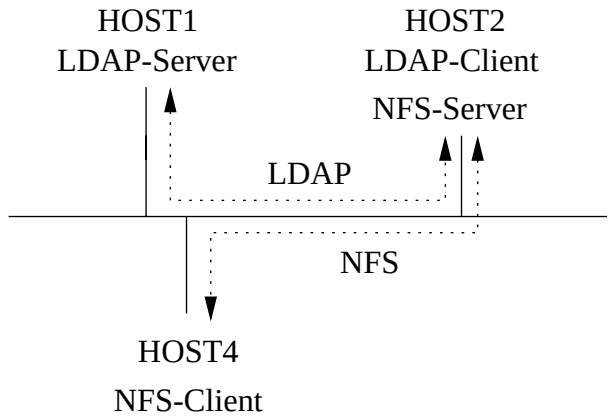

Abbildung 5.10: Zusammenhang zwischen LDAP und NFS

5.3 NIS-Passwortverwaltung

Der Network Information Service (NIS) dient dazu, Benutzerkonten zentral an einem Rechner (NIS-Server) zu pflegen, damit die als NIS-Client eingerichteten Hosts bei deren lokaler Anmeldung die Konten des Servers zur Verfügung haben. Falls also zum Beispiel der Rechner `host1` als NIS-Server eingerichtet ist und dort lokal ein Benutzer `FlemmingJ` vorhanden ist, dann können sich alle NIS-Clients (zum Beispiel `host2`) mit dieser Benutzerkennung und dem zugehörigen Passwort am System anmelden. Der Benutzer `FlemmingJ` existiert am `host2` jedoch nicht.

Sofern dieses NIS-System aktiv ist, erfolgt wie beschrieben die Verifizierung des Benutzernamens und des Passworts am NIS-Server. In diesem Beispiel wird gezeigt, wie es generell möglich ist, Benutzer und Kennwörter an einem Verzeichnisdienst überprüfen zu lassen.

Im Weiteren wird der Rechner `host1` als NIS- und LDAP-Server eingerichtet. Die Rechner `host2`, `host3` usw. werden als NIS- und LDAP-Clients konfiguriert.

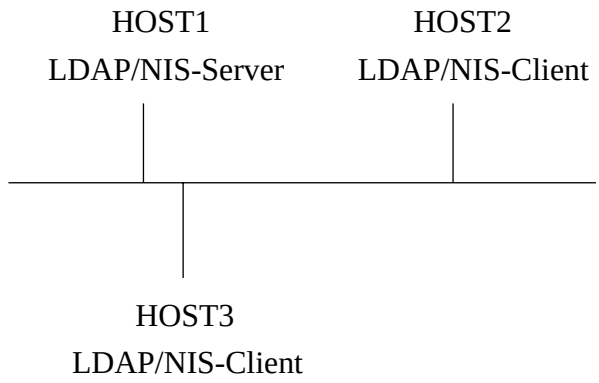

Abbildung 5.11: Zusammenhang zwischen LDAP und NIS

5.3.1 Funktion

Wie die Kombination aus LDAP und NIS funktioniert, kann wie folgt beschrieben werden:

1. Der NIS-Server wird dahin gehend konfiguriert, dass er bestimmte Einträge aus der zentralen Benutzerdatei `/etc/passwd` allen NIS-Clients im Netzwerk zur Verfügung stellen kann.
2. Der LDAP-Server wird so eingerichtet, dass er die gleichen Benutzerobjekte abspeichert, die der NIS-Server verwaltet.
3. An den Clients wird die Position des NIS-Servers bekannt gegeben.
4. An den Clients wird die Position des LDAP-Servers bekannt gegeben.
5. Bis zu diesem Punkt existieren die Dienste NIS und LDAP unabhängig voneinander. Damit sie kombiniert werden, ist es notwendig, dass alle Clients so eingerichtet werden, dass sie den Benutzernamen und das Kennwort an der LDAP-Datenbank verifizieren. Die weiteren Eigenschaften wie zum Beispiel die nummerische Benutzerkennung und die zu startende Shell werden per NIS in Erfahrung gebracht.

In den folgenden Unterabschnitten werden die einzelnen Schritte beschrieben, die zur Realisierung dieses Beispiels durchgeführt werden müssen.

5.3.2 Konfiguration des NIS-Servers

Als Erstes befassen wir uns mit der Konfiguration des NIS-Servers.

NIS ist in der Lage, den Clients eine Vielzahl von Informationen des Servers zur Verfügung zu stellen. Im ersten Schritt ist festzulegen, dass der NIS-Dienst lediglich

die Benutzer betreffen soll, und zwar jene mit einer ID größer gleich 500. Dazu sind in der Datei `/var/yp/Makefile` die folgenden Einträge notwendig:

```
MINUID=500

MERGE_PASSWD=true

all: passwd
```

Da NIS das Shadow-Passwortsystem nicht unterstützt, müssen die Einträge der Dateien `passwd` und `shadow` für diesen Zweck vereint werden, was durch `MERGE_PASSWD` ausgedrückt wird.

Die Kommunikation zwischen NIS-Server und -Client erfolgt über eine NIS-Domäne. Sie muss ebenfalls im System verankert werden. Bei einer SuSE-Distribution kann dies durch das Setzen des Parameters `YP_DOMAINNAME` geschehen. Bei Red Hat muss man `linuxconf` verwenden. Der Parameter kann auch an der Linux-Shell mit dem Kommando `domainname` dynamisch gesetzt werden:

```
root@host1:~ # domainname nisintern
root@host1:~ #
```

Der NIS-Server muss nun permanent beim Systemstart gestartet werden. Diese Einstellung kann in jeder Distribution mit einem spezifischen Tool durchgeführt werden (`yast`, `linuxconf`, ...).

Die Konfigurationsdatei des Servers `/etc/ypserv.conf` muss in der Regel nicht verändert werden.

Letztlich müssen noch zwei Schritte durchgeführt werden. Zum einen ist der NIS-Server zu starten:

```
root@host1:~ # /etc/rc.d/init.d/ypserv start
Starting service YP server                                  done
root@host1:~ #
```

NIS verwaltet die Informationen in Datenbanken. Damit diese Datenbanken erzeugt werden, ist zum anderen das Kommando `/usr/lib/ypinit -m` aufzurufen:

```
root@host1:~ # /usr/lib/ypinit -m

At this point, we have to construct a list of the hosts which
will run NIS servers.  host1.intern is in the list of NIS
server hosts.  Please continue to add the names for the other
hosts, one per line.  When you are done with the list, type a
<control D>.
        next host to add:  host1.intern
        next host to add:
```

```
The current list of NIS servers looks like this:

host1.intern

Is this correct?  [y/n: y]
We need some  minutes to build the databases…
Building /var/yp/nisintern/ypservers…
Running /var/yp/Makefile…
gmake[1]: Entering directory /var/yp/nisintern'
Updating passwd.byname…
Updating passwd.byuid…
gmake[1]: Leaving directory /var/yp/nisintern'

root@host1:~ #
```

Damit ist die Konfiguration des NIS-Servers abgeschlossen.

5.3.3 Konfiguration des LDAP-Servers

Für den LDAP-Server kann die bereits mehrfach benutzte Konfigurationsdatei aus dem letzten Kapitel verwendet werden:

```
root@host1:~ # cat /etc/openldap/slapd.conf
#
# Konfigurationsdatei des LDAP-Servers
#
# slapd.conf
#
# Erstellt von Jens Banning
#

#
# Allgemeines
#

include         /etc/openldap/slapd.at.conf
include         /etc/openldap/slapd.oc.conf

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

loglevel        256
schemacheck     on

sizelimit       500
timelimit       3600

#
```

```
# Datenbank
#

database        ldbm
lastmod         off

cachesize       1000
dbcachesize     100000

directory       /var/lib/ldap
suffix          "o=Intern"

rootdn          "cn=Admin, o=Intern"
rootpw          {CRYPT}zs0QLoUVZvS6E

#
# Rechte
#

defaultaccess   read

access to attr=userPassword
       by self write
       by * none

access to attr=telephoneNumber
       by self write
       by * read

access to dn="cn=Admin,o=Intern"
       by * none

access to *
       by * read

root@host1:~ #
```

Somit kann jeder Benutzer sein eigenes Passwort ändern, die Kennwörter anderer jedoch nicht einsehen.

5.3.4 Erstellen der Datenbank

Benutzer, die per NIS im Netzwerk zur Verfügung gestellt werden sollen, werden auf dem NIS-Server in der gewohnten Weise mit dem Administrationstool der Distribution oder direkt mit dem Kommando `useradd` hinzugefügt. Anschließend muss man das Kommando `ypinit` erneut starten, damit die NIS-Datenbanken aktualisiert werden.

In der LDAP-Datenbank muss man nun parallel die mit NIS erstellten Benutzer als Objekte abspeichern. Dies kann zum einen per Hand geschehen, zum anderen ist es jedoch auch möglich, die LDAP-Datenbank anfänglich direkt aus einer LDIF-Datei zu erstellen. Dazu kann das Kommando `ldif2ldbm` verwendet werden. Es wandelt die zugrunde liegende LDIF-Datei in die LDAP-Datenbank um. Anschließend sind in der Datenbank lediglich die Objekte vorhanden, die per LDIF definiert wurden. Alte Einträge sind überschrieben worden. Sofern dieses Verhalten unerwünscht ist, können die Objekte der LDIF-Datei auch mit dem Kommando `ldapadd` hinzugefügt werden.

```
root@host1:~ # cat intern.ldif
dn: o=Intern
objectclass: organization
o: Intern

dn: cn=Admin, o=Intern
objectclass: person
cn: Admin
sn: Admin
telephonenumber: 800
description: "LDAP Administrator"
mail: admin@intern

dn: ou=Einkauf, o=Intern
objectclass: organizationalUnit
ou: Einkauf
description: "Abteilung Einkauf"

dn: ou=Verkauf, o=Intern
objectclass: organizationalunit
ou: Verkauf
description: "Abteilung Verkauf"

dn: cn=MeierP, ou=Einkauf, o=Intern
objectclass: person
cn: MeierP
sn: Meier
userpassword: geheim
telephonenumber: 100
description: "Peter Meier"
mail: meierp@intern

dn: cn=MuellerC, ou=Einkauf, o=Intern
objectclass: person
cn: MuellerC
sn: Mueller
userpassword: geheim
telephonenumber: 101
```

```
description: "Christian Mueller"
mail: muellerc@intern

dn: cn=SchulzeI, ou=Einkauf, o=Intern
objectclass: person
cn: SchulzeI
sn: Schulze
userpassword: geheim
telephonenumber: 102
description: "Ingrid Schulze"
mail: schulzei@intern

dn: cn=JohansenL, ou=Verkauf, o=Intern
objectclass: person
cn: JohansenL
sn: Johansen
userpassword: geheim
telephonenumber: 201
description: "Lucas Johansen"
mail: johansenl@intern

dn: cn=PaulsenI, ou=Verkauf, o=Intern
objectclass: person
cn: PaulsenI
sn: Paulsen
userpassword: geheim
telephonenumber: 202
description: "Ida Paulsen"
mail: paulseni@intern

dn: cn=FlemmingJ, ou=Verkauf, o=Intern
objectclass: person
cn: FlemmingJ
sn: Flemming
userpassword: geheim
telephonenumber: 200
description: "Jan Flemming"
mail: flemmingj@intern
root@host1:~ # ldif2ldbm -i intern.ldif
root@host1:~ #
```

In der Datenbank ist nun die bekannte Struktur der Organisation `Intern` vorhanden. Sie umfasst drei Benutzerobjekte im `Einkauf` und drei im `Verkauf`. Da die Objekte später für den Anmeldevorgang benutzt werden sollen, ist es notwendig, dass jeweils ein Passwort definiert wird.

5.3.5 Aktualisieren der Datenbank per Skript

Es ist für die Funktionsweise wichtig, dass die Benutzer sowohl in der Datei `/etc/passwd` des NIS-Servers als auch in der Datenbank des LDAP-Servers abgespeichert werden. Falls neue Benutzer zu erstellen sind, müssen drei Konfigurationsschritte durchgeführt werden:

1. Als Erstes ist der Benutzer auf dem NIS-Server zu erstellen. Er wird dadurch in die Datei `/etc/passwd` eingetragen.
2. Anschließend ist das Kommando `ypinit` zu starten. Erst nach dem Start dieses Kommandos ist der Anwender für die NIS-Clients verfügbar.
3. Zum Abschluss muss, damit eine Datenkonsistenz zwischen NIS und LDAP besteht, der Anwender an der gewünschten Stelle im Verzeichnisdienst als Objekt mit zugehörigem Passwort erstellt werden.

Diese drei Schritte können am LDAP/NIS-Server natürlich vom Systemadministrator von Hand durchgeführt werden. Wesentlich besser und einfacher ist es jedoch, ein Shell-Skript zu schreiben, das diese drei Aktionen selbstständig ausführt. Dieses Skript kann dann im Verzeichnis `/usr/sbin/` abgelegt werden:

```
root@host1:/usr/sbin # cat nisldapuseradd
#!/bin/bash
#
# Erstellen eines NIS- und LDAP-Benutzers
#
# Aufruf:   nisldapuseradd <Benutzer> <Passwort> <Container>
#
# Beispiel: nisldapuseradd RabeD geheim "ou=Einkauf, o=Intern"
#
# Erstellt von Jens Banning
#

#
# Analyse der Parameter.
#

if [ $# -ne 3 ]; then
   echo
   echo "nisldapuseradd <Benutzer> <Passwort> <Container>"
   echo
   exit 1
fi

benutzer=$1
passwort=$2
container=$3
```

```
#
# 1. Schritt: Benutzer in /etc/passwd erstellen.
#

useradd $benutzer

#
# 2. Schritt: ypinit starten.
#

/usr/lib/yp/ypinit -m

#
# 3. Schritt: LDAP-Objekt erstellen
#

echo "dn: cn=$benutzer, $container" >/tmp/nisldap.ldif
echo "objectclass: person"         >>/tmp/nisldap.ldif
echo "cn: $benutzer"               >>/tmp/nisldap.ldif
echo "sn: $benutzer"               >>/tmp/nisldap.ldif
echo "userpassword: $passwort"     >>/tmp/nisldap.ldif

ldapadd -D "cn=Admin, o=Intern" -W -f /tmp/nisldap.ldif

rm /tmp/nisldap.ldif

root@host1:/usr/sbin # cd
root@host1:~ # nisldapuseradd RabeD geheim \
# "ou=Einkauf, o=Intern"

At this point, we have to construct a list of the hosts which
will run NIS servers.  host1.intern is in the list of NIS
server hosts.  Please continue to add the names for the other
hosts, one per line.  When you are done with the list, type a
<control D>.
        next host to add:  host1.intern
        next host to add:
The current list of NIS servers looks like this:

host1.intern

Is this correct?  [y/n: y]  We need some  minutes to build
the databases…
Building /var/yp/nisdomain/ypservers…
Running /var/yp/Makefile…
gmake[1]: Entering directory '/var/yp/nisdomain'
Updating passwd.byname…
Updating passwd.byuid…
```

```
gmake[1]: Leaving directory '/var/yp/nisdomain'
Enter LDAP Password:

adding new entry cn=RabeD, ou=Einkauf, o=Intern

root@host1:~ #
```

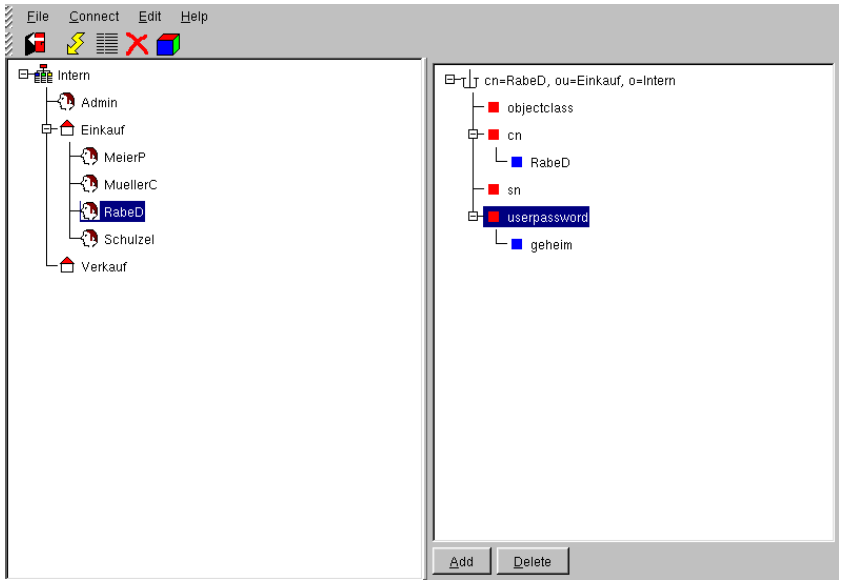

Abbildung 5.12: NIS-Benutzer in der LDAP-Datenbank

Nun ist der Benutzer korrekt im NIS- und LDAP-Dienst eingetragen (siehe Abbildung 5.12). Das Passwort wurde lediglich in der LDAP-Datenbank abgelegt, da es später per LDAP verifiziert werden soll. Der Eintrag in der Datei `/etc/shadow` wird nicht verwendet.

5.3.6 Konfiguration des Clients

Wie in Abbildung 5.11 auf Seite 204 zu sehen ist, müssen die Client-Rechner in der dargestellten Umgebung zweifach konfiguriert werden:

- Als Erstes ist es notwendig, die gewünschten Hosts im Netzwerk als NIS-Clients einzurichten. Dadurch können sie die Informationen des NIS-Servers zur Anmeldung verwenden.
- Da die Überprüfung des Anmeldenamens und des Kennwortes an den Daten des LDAP-Servers erfolgen soll, müssen die Rechner ebenfalls als LDAP-Client eingerichtet werden.

Betrachten wir zunächst den NIS-Client. Seine Konfiguration teilt sich in mehrere Schritte auf:

1. Damit die NIS-Kommunikation zwischen Server und Client überhaupt stattfinden kann, ist es zwingend erforderlich, dass bei beiden Partnern die gleiche NIS-Domäne eingestellt wird. Dazu kann man temporär den Befehl `domainname` benutzen. Ferner kann die NIS-Konfiguration auch mit dem Administrationstool der verwendeten Distribution erfolgen. Im Tool `yast` der SuSE-Distribution ist wie beim NIS-Server auch die Variable `YP_DOMAINNAME` auf `nisintern` zu setzen.

2. Ferner muss man mit dem distributionsabhängigen Tool einstellen, dass der NIS-Client permanent beim Rechnerstart aktiviert wird (`START_YPBIND` bei SuSE).

3. Die Konfigurationsdatei des Client-Rechners heißt `yp.conf` und befindet sich im Verzeichnis `/etc/`. In ihr wird der Name bzw. die IP-Adresse des NIS-Servers festgehalten. Sie kann mit einem Editor erstellt werden. Bei einigen Distributionen wird sie auch automatisch angelegt und mit den gewünschten Daten gefüllt:

```
root@host2:~ # cat /etc/yp.conf
#
# Konfigurationsdatei des NIS-Clients
#
# Erstellt von Jens Banning
#

ypserver 192.168.17.1

root@host2:~ #
```

4. Im vorletzten Schritt muss nun die Datei `/etc/passwd` des Clients um eine Zeile ergänzt werden. Sie gibt an, dass für die Anmeldung neben den lokalen Daten auch die Informationen des NIS-Servers relevant sind. Die Zeile ist am Ende zu ergänzen und besteht aus einen Pluszeichen und sechs Doppelpunkten:

```
root@host2:~ # tail -2 /etc/passwd
zope:x:64:2:Zope Server:/var/lib/zope:/bin/false
+::::::
root@host2:~ #
```

5. Im letzten Schritt wird der NIS-Client gestartet.

```
root@host1:~ # /etc/rc.d/init.d/ypclient start
Starting ypbind                                          done
root@host1:~ #
```

   Damit ist die Verbindung zwischen Server und Client komplett eingerichtet.

Ob die Kommunikation zwischen Client und Server tatsächlich ordnungsgemäß stattfindet, kann mit dem Kommando `ypwhich` geprüft werden:

```
root@host2:~ # ypwhich
host1.intern
root@host2:~ #
```

Bis zu diesem Zeitpunkt können die vom NIS-Server zur Verfügung gestellten Benutzer für die Anmeldung verwendet werden. Die Verifizierung des Passworts erfolgt jedoch per NIS und nicht wie gewünscht per LDAP. Daher ist der Client-Rechner auch als LDAP-Client einzurichten. Auf diese Weise werden der Server, dessen Port und die Suchbasis definiert:

```
root@host2:~ # cat /etc/openldap/ldap.conf
#
# Konfigurationsdatei des LDAP-Clients
#
# ldap.conf
#
# Erstellt von Jens Banning
#

BASE    o=Intern
HOST    host1.intern
PORT    389

SIZELIMIT       12
TIMELIMIT       15
DEREF           never

root@host2:~ #
```

5.3.7 PAM-Konfiguration am Client

An dieser Stelle sind NIS- und LDAP-Client eingerichtet. Jetzt muss jedoch noch festgelegt werden, dass die Verifizierung des Benutzernamens und des Kennworts an der LDAP-Datenbank erfolgt. Dazu sei generell einiges zur Authentifizierung unter Linux gesagt: Linux ist ein dynamisches Betriebssystem, das in Bezug auf die Flexibilität nahezu keine Wünsche offen lässt. So kann auch die Authentifizierung beim Anmeldevorgang leicht an die gewünschten Gegebenheiten angepasst werden.

Dies wird durch die so genannten PAM-Module realisiert. PAM steht für Pluggable Authentication Modules. Durch diese Module wird zum Beispiel definiert, an welcher Stelle das Passwort auf Korrektheit zu überprüfen ist. Der Sinn dieser austauschbaren Module besteht darin, dass Linux somit für zukünftige, zurzeit noch unbekannte Beglaubigungsarten offen ist. Es wären in einem solchen Fall keine Kommandos und Anwendungen neu zu schreiben, sondern es reicht, lediglich die betroffenen PAM-Module zu erneuern.

Sie befinden sich standardmäßig im Verzeichnis `/etc/pam.d/`. Dort sind zwei Dateien mit den Namen `login` und `passwd` vorhanden. Sie beschreiben, wie die Login-Daten und das Passwort verifiziert werden sollen. Die Ausgabe der Dateien ist an dieser Stelle leicht abgekürzt, was durch die drei Punkte am Ende einer Zeile angedeutet werden soll.

```
root@host2:/etc/pam.d # cat login
#%PAM-1.0
auth     requisite      /lib/security/pam_unix.so       …
auth     required       /lib/security/pam_securetty.so
auth     required       /lib/security/pam_nologin.so
#auth    required       /lib/security/pam_homecheck.so
auth     required       /lib/security/pam_env.so
auth     required       /lib/security/pam_mail.so
account  required       /lib/security/pam_unix.so
password required       /lib/security/pam_pwcheck.so    …
password required       /lib/security/pam_unix.so       …
session  required       /lib/security/pam_unix.so       …
session  required       /lib/security/pam_limits.so

root@host2:/etc/pam.d # cat passwd
#%PAM-1.0
auth     required       /lib/security/pam_unix.so       …
account  required       /lib/security/pam_unix.so
password required       /lib/security/pam_pwcheck.so    …
password required       /lib/security/pam_unix.so       …
session  required       /lib/security/pam_unix.so

root@host2:/etc/pam.d #
```

Neben diesen beiden Dateien existieren in dem Verzeichnis natürlich noch weitere Konfigurationen. Nähere Informationen zu PAM sind zum einen in der Manual Page `man pam` zu finden und zum anderen in den Büchern [4] und [5].

Es liegt nahe, dass in diesen Dateien Änderungen vorgenommen und dass auch die notwendigen Bibliotheken installiert werden müssen. Deshalb ist es zunächst erforderlich, das RPM-Paket

```
pam_ldap
```

zu installieren. Im Verzeichnis `/usr/share/doc/packages/pam_ldap/pam.d/` befinden sich dann die neuen PAM-Module. Sie müssen in das PAM-Verzeichnis `/etc/pam.d/` kopiert werden:

```
root@host2:~ # cp /usr/share/doc/packages/pam_ldap/pam.d/* \
# /etc/pam.d
root@host2:~ #
```

In den Konfigurationsdateien `login` und `passwd` wurde nun die LDAP-Authentifizierung zusätzlich eingetragen:

```
root@host2:/etc/pam.d # cat login
#%PAM-1.0
auth     required       /lib/security/pam_securetty.so
auth     required       /lib/security/pam_nologin.so
auth     sufficient     /lib/security/pam_ldap.so
auth     required       /lib/security/pam_unix_auth.so  …
auth     required       /lib/security/pam_homecheck.so
auth     required       /lib/security/pam_env.so
auth     required       /lib/security/pam_mail.so
account  sufficient     /lib/security/pam_ldap.so
account  required       /lib/security/pam_unix.so
password required       /lib/security/pam_pwcheck.so    …
password required       /lib/security/pam_ldap.so
password required       /lib/security/pam_unix.so       …
session  required       /lib/security/pam_unix.so       …
session  required       /lib/security/pam_limits.so

root@host2:/etc/pam.d # cat passwd
#%PAM-1.0
auth     sufficient     /lib/security/pam_ldap.so
auth     required       /lib/security/pam_unix.so       …
account  sufficient     /lib/security/pam_ldap.so
account  required       /lib/security/pam_unix.so
password required       /lib/security/pam_pwcheck.so    …
password sufficient     /lib/security/pam_ldap.so
password required       /lib/security/pam_unix.so       …
session  required       /lib/security/pam_unix.so

root@host2:/etc/pam.d #
```

Zum Abschluss ist noch ein Konfigurationsschritt auf dem Client durchzuführen. Per Default werden beim Anmelden alle Benutzerobjekte unterhalb des Containers verwendet, der mit `BASE` in der Datei `ldap.conf` angegeben ist. Dort wird der vom Anwender eingegebene Benutzername als Attribut eines Objekts gesucht. Um welches Attribut es sich dabei handelt, ist ebenfalls in der Datei `ldap.conf` einzutragen. Da die Anmeldenamen in diesem Beispiel stets im Common Name `cn` abgelegt sind, muss die Datei `/etc/openldap/ldap.conf` wie folgt ergänzt werden. Per Default wird der Benutzername in der hier nicht verwendeten Eigenschaft `uid` vermutet:

```
root@host2:~ # cat /etc/openldap/ldap.conf
#
# Konfigurationsdatei des LDAP-Clients
#
# ldap.conf
#
```

```
# Erstellt von Jens Banning
#

BASE    o=Intern
HOST    host1.intern
PORT    389

SIZELIMIT       12
TIMELIMIT       15
DEREF           never

pam_login_attribute cn

root@host2:~ #
```

Das Kennwort wird grundsätzlich mit dem zugehörigen Attribut `userPassword` verglichen.

Nun ist es dem Anwender auf der Client-Seite möglich, sich zum Beispiel mit dem Benutzernamen `RabeD` und dem Kennwort `geheim` anzumelden:

```
Starting identd                                              done
Starting Name Service Cache Daemon                           done
Master Resource Control: runlevel 2 has been              reached

Welcome to SuSE Linux 7.0 (i386) - Kernel 2.2.16 (tty1).

host2 login: RabeD
Password:
Last login: Thu Dec 14 09:03:24 on tty1
Have a lot of fun…
No directory /home/RabeD!
Logging in with home = "/".
RabeD@host2:/ #
```

Es sei noch darauf hingewiesen, dass es sinnvoll ist, wenn die Client-Rechner nach der Anmeldung per NFS Zugriff auf ihre Heimatverzeichnisse haben. Auf NFS soll an dieser Stelle jedoch nicht näher eingegangen werden.

5.3.8 Ändern des Passworts mit `kldap`

Das Ändern des Passwortes kann nun auf verschiedene Arten erfolgen.

Die sicher einfachste Möglichkeit ist die Verwendung des Programmes `kldap`. Dazu muss sich der Anwender mit seinem Namen (`RabeD`) und seinem aktuellen Passwort (`geheim`) am Verzeichnisdienst anmelden (siehe Abbildung 5.13). Durch die `access`-Definitionen des LDAP-Servers hat er das Recht sein eigenes Kennwort

zu verändern. Nachdem die Verbindung hergestellt wurde, kann der Anwender nun sein Kennwort einsehen (siehe Abbildung 5.14).

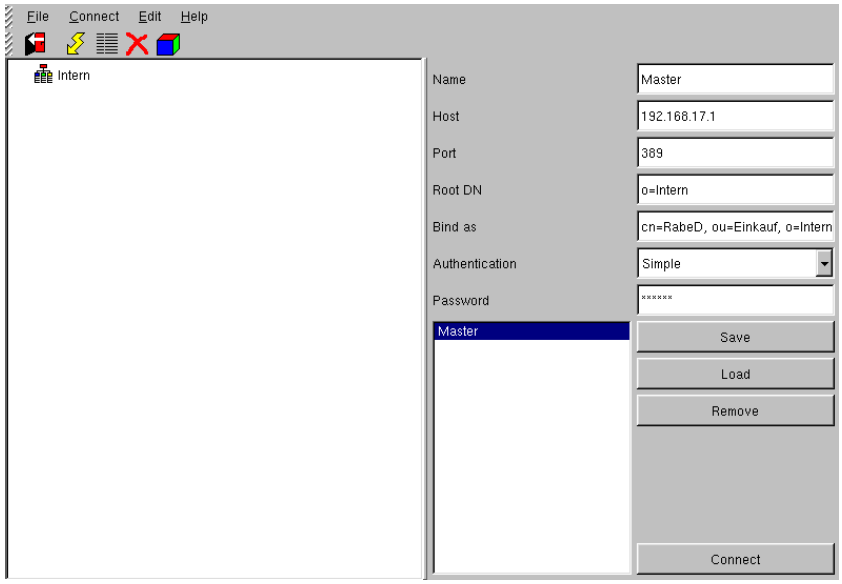

Abbildung 5.13: `kldap`: Anmelden am Verzeichnisdienst

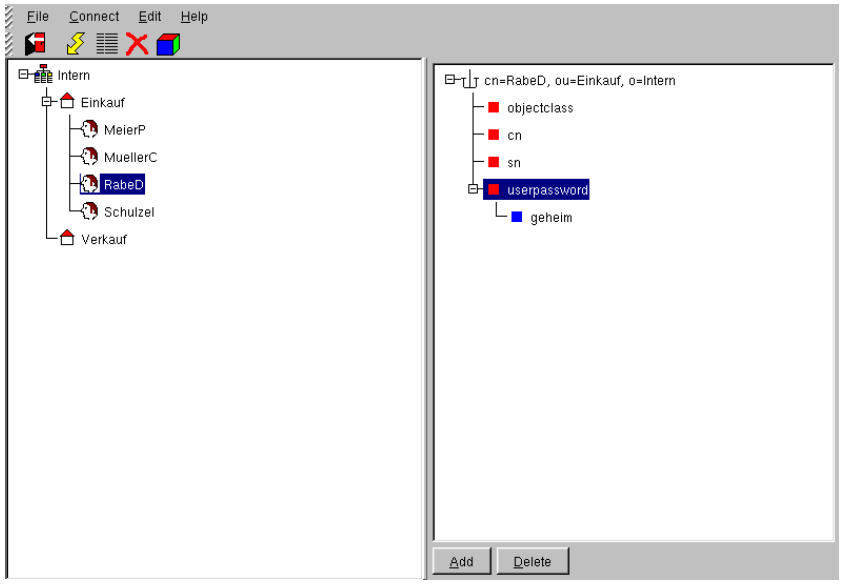

Abbildung 5.14: `kldap`: Einsehen des eigenen Kennwortes

Das Passwort wird in zwei Schritten geändert.

1. Das alte Passwort ist zunächst zu löschen. Dazu ist `geheim` zu markieren und anschließend DELETE anzuwählen.

2. Nun kann das neue Kennwort eingestellt werden. Dazu markiert man das Attribut `userpassword` und klickt ADD mit der Maus an. In dem dann erscheinenden Fenster kann das neue Kennwort eingegeben werden (zum Beispiel `secret`; siehe Abbildung 5.15).

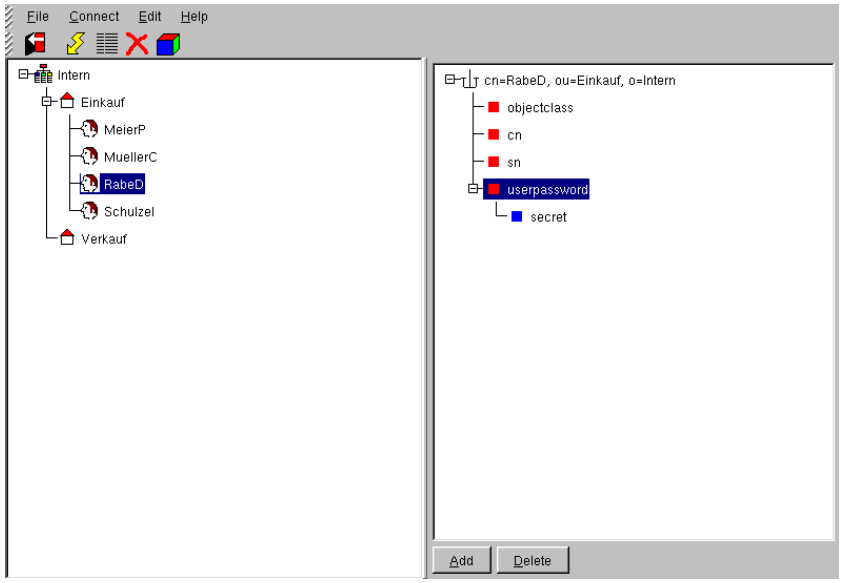

Abbildung 5.15: `kldap`: Einsehen des neuen Kennwortes

Das neue Kennwort ist nun aktiv und kann vom Anwender `RabeD` verwendet werden. Auf diese Weise ist es natürlich auch möglich, dass der Administrator von jedem beliebigen Client-Rechner aus die Kennwörter der Benutzer ändert.

5.3.9 Erhöhen der Sicherheit mit `ssh`

Die Kommunikation zwischen einem LDAP-Client und einem LDAP-Server erfolgt grundsätzlich unverschlüsselt. Das bedeutet, dass die auf diesem Wege übertragenen Daten im Klartext über das Netzwerk ausgetauscht werden. Somit fällt es potenziellen Angreifern sehr leicht, die Informationen mitzulesen. Durch das so genannte *Network Sniffing*, das zum Beispiel mit der Anwendung `ethereal` durchgeführt werden kann, können die zwischen Client und Server ausgetauschten Informationen eingesehen werden. Das bedeutet für das in diesem Abschnitt beschriebene Beispiel,

dass auch die Passwörter bei der Anmeldeprozedur im Klartext zum LDAP-Server übertragen werden, um sie dort zu prüfen. Deshalb ist es sinnvoll — wenn nicht sogar zwingend erforderlich — die Verbindung zu verschlüsseln.

Als Standard für eine gesicherte und verschlüsselte Datenübertragung hat sich unter Linux die Secure Shell (SSH) durchgesetzt. Mit dem gleichnamigen Kommando `ssh` ist es möglich, eine Verbindung von einem Rechner im Netzwerk zu einem anderen aufzubauen. Dabei werden die Daten durch die Verwendung von Schlüsseln sicher übertragen. Die so übertragenen Informationen sind nicht durch das *Network Sniffing* gefährdet.

Weitere Informationen zur Secure Shell wurden bereits in Abschnitt 3.3.6 auf der Seite 39 gegeben.

Damit die sichere Verbindung vom Client zum Server zustande kommt, muss dem Kommando `ssh` am Client der Hostname des Rechners übergeben werden, zu dem die Verbindung aufgebaut werden soll. Ferner kann ein Kommando angegeben werden, was dort zu starten ist. Der Aufruf hat demnach die Form:

```
ssh <Hostname> <Kommando>
```

Erfolgt dieser Aufruf, so wird nach dem Kennwort des Benutzers auf der entfernten Maschine gefragt. Auch diese Übertragung findet bereits verschlüsselt statt:

```
root@host2:~ # ssh host1 free
root@host1's password:
Filesystem      1k-blocks      Used Available Use% Mounted on
/dev/hda1          991936    849812     90920  90% /
root@host2:~ #
```

Im Beispiel wurde auf dem Rechner `host1` der Befehl `free` ausgeführt.

Neben dieser Nutzung ist es möglich, einen so genannten Tunnel zwischen den beiden Rechnern zu installieren. Dieser Tunnel dient dann als sicherer Datenübertragungskanal zwischen den Partnern. Durch den sicheren Tunnel können anschließend unsichere Kommunikationen wie zum Beispiel LDAP übertragen werden. Da es Außenstehenden nicht möglich ist, die Daten des Tunnels zu analysieren, kann niemand die Daten (LDAP) einsehen, die sich darin befinden.

Standardmäßig wird vom Client `host2` eine TCP-Verbindung zum `host1` hergestellt, Port 389, um auf den LDAP-Server zuzugreifen. Falls ein Tunnel von `host2` zu `host1` eingerichtet wird, so erfolgt die TCP-Verbindung von `host2` an einen Port seines Rechners. Anschließend werden die Daten lokal vom Tunnel angenommen und dann gesichert zum `host1` gesendet.

Die Grafik 5.16 verdeutlich dieses Verfahren. Der Tunnel wird von `host2` zu `host1` eingerichtet. Falls der Rechner `host2` per LDAP seinen lokalen TCP-Port 7777 anspricht, werden die Daten durch den Tunnel an den Rechner `host1` und dort an den Port 389 transportiert.

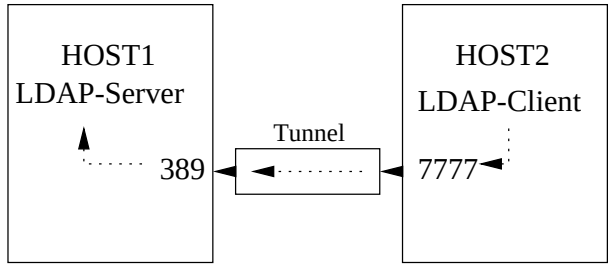

Abbildung 5.16: SSH-Tunnel

Damit dieses Verfahren funktioniert, sind zwei Konfigurationsschritte notwendig:

1. Als Erstes ist der SSH-Tunnel zu initialisieren. Dazu muss das am entfernten Rechner zu startene Kommando so lange aktiv sein, wie auch der Tunnel aktiv sein soll. Falls der Tunnel einen Tag aktiv sein soll, bedeutet dieses, dass das entfernte Kommando eine Laufzeit von einem Tag benötigen muss. Dies lässt sich am besten mit dem Kommando `sleep` realisieren. Es wartet die ihm übergebene Anzahl Sekunden ab, ohne eine Aktion auszuführen. Damit das Kommando 24 Stunden aktiv ist, muss es in der Form

   ```
   sleep 86400
   ```

   aufgerufen werden. Am `host2` würde dies so aussehen:

   ```
   ssh host1 sleep 86400
   ```

2. Nun muss dieser Aufruf um die Definition des Tunnels erweitert werden. Mit dem folgenden Aufruf sollen alle lokalen Anfragen an den TCP-Port 7777 sicher zum Rechner `host1` an den Port 389 übertragen werden:

   ```
   root@host2:~ # ssh -f -L 7777:host1:389 host1 sleep 86400
   root@host1's password:
   root@host2:~ #
   ```

   Durch die Option `-f` wird die Secure Shell nach der Eingabe des Passworts in den Hintergrund gestellt.

Nun ist der Tunnel aktiv und kann genutzt werden. Damit die LDAP-Anfragen nicht mehr direkt an den Rechner `host1`, Port 389, sondern an den lokalen Rechner `localhost`, Port 7777, gesendet werden, muss die Datei `ldap.conf` entsprechend geändert werden.

```
root@host2:~ # cat /etc/openldap/ldap.conf
```

```
#
# Konfigurationsdatei des LDAP-Clients
#
# ldap.conf
#
# Erstellt von Jens Banning
#

BASE    o=Intern
HOST    localhost
PORT    7777

SIZELIMIT       12
TIMELIMIT       15
DEREF           never

pam_login_attribute cn

root@host2:~ #
```

Damit besteht nun eine sichere, verschlüsselte Verbindung zwischen Client und Server.

Damit der Tunnel grundsätzlich beim Rechnerstart aktiviert wird, ist ein Shell-Skript zu erstellen. Im gewünschten Runlevel des Rechners muss ferner ein Start-Link auf das Skript erstellt werden. In diesem Beispiel besteht auch die Möglichkeit, den Start des SSH-Tunnels in dem Skript des NIS-Clients einzubauen.

Da die Eingabe des Kennworts bei der Initialisierung des Tunnels eher unerwünscht ist, besteht ferner die Möglichkeit, den öffentlichen Schlüssel des Quellrechners (`host2`) in einer bestimmten Datei des Zielrechners (`host1`) einzutragen:

1. Der öffentliche Schlüssel von `host2` befindet sich für den Benutzer `root` im Unterverzeichnis `.ssh/` seines Heimatverzeichnisses. Er ist in der Datei `identity.pub` abgespeichert.

2. Dieser Schlüssel ist nun am Rechner `host1` im gleichen Verzeichnis an die Datei `authorized_keys` anzuhängen. Falls die Datei noch nicht existiert, kann sie neu erstellt werden. Der folgende Aufruf kopiert den öffentlichen Schlüssel von `host2` auf die entsprechende Datei am `host1`:

   ```
   root@host2:~/.ssh # scp identity.pub \
   # host1:/root/.ssh/authorized_keys
   root@host1's password:
   identity.pub   |   0 KB |   0.3 kB/s   | ETA: 00:00:00 | 100%

   root@host2:~/.ssh #
   ```

Falls die Secure Shell nun gestartet wird, so wird nicht nach dem Kennwort gefragt, da `host2` dem Rechner `host1` bekannt ist:

```
root@host2:~ # ssh -f -L 7777:host1:389 host1 sleep 86400
root@host2:~ #
```

Abschließend wird diese Zeile in das Startskript des NIS-Clients mit eingebunden. Die folgenden Zeilen zeigen lediglich den relevanten Ausschnitt des Skripts an:

```
root@host2:~ # cat /etc/rc.d/init.d/ypclient
case "$1" in
   start)
      /bin/ypdomainname &> /dev/null
      if [ $? -ne 0 -o -z "/bin/domainname" ]; then
         if [ -n "$YP_DOMAINNAME" ]; then
             /bin/ypdomainname "$YP_DOMAINNAME"
         else
             exit 0;
         fi
      fi

      echo -n "Starting ypbind and SSH-Tunnel"

      ssh -f -L 7777:host1:389 host1 sleep 86400
      startproc /usr/sbin/ypbind
…
root@host2:~ #
```

Der Start des NIS-Clients aktiviert somit zuvor den Tunnel.

```
root@host1:~ # /etc/rc.d/init.d/ypclient start
Starting ypbind and SSH-Tunnel                          done
root@host1:~ #
```

Weitere Informationen zur Secure Shell finden Sie in der Manual Page (`man ssh`) und in [6].

Pearson Education

ADDISON-WESLEY

6 Ausblick

Dieses Kapitel gibt einen kurzen Ausblick auf die weitere Entwicklung im Bereich der LDAP-Software unter Linux.

Die vorliegende Version der Software der OpenLDAP-Organisation bietet Administratoren von Netzwerken zurzeit ein sehr großes Spektrum an Möglichkeiten, wie die Verzeichnisdienste genutzt werden können; siehe das Beispiel der NFS-Verwaltung im Abschnitt 5.2. So ist im Prinzip jeder Administrator in der Lage, genau die Informationen im Directory Information Tree zu verwalten, die er dort benötigt. Auf diese Weise können neben dem Network File System (NFS) natürlich auch andere Netzwerkdienste wie das Dynamic Host Configuration Protocol (DHCP) oder das Network Time Protocol (NTP) realisiert werden. All diese Möglichkeiten setzen jedoch voraus, dass der LDAP-Administrator das Schema erweitert, um die gewünschten Informationen in irgendeiner Art und Weise abspeichern zu können.

Letztlich müssen dann die Konfigurationsdateien der eigentlichen, oben angesprochenen Dienste mit den LDAP-Informationen gefüllt werden. Dieses kann dadurch geschehen, dass, wie in diesem Buch gezeigt, selbst geschriebene Skripten der Linux Shell `bash` verwendet werden, die die Informationen der LDAP-Datenbank in die Dateien schreiben, die für den zu konfigurierenden Dienst relevant sind. Ferner können auch direkt Anwendungen in der Programmiersprache C geschrieben werden. Es ist jedoch generell notwendig, das Bindeglied zwischen LDAP und dem betreffenden Dienst (zum Beispiel NFS) selbst zu erstellen.

Die Weiterentwicklung der LDAP-Software unter Linux sollte es wünschenswerterweise ermöglichen, dass Netzwerkadministratoren bereits auf ein vordefiniertes Schema zurückgreifen können, das ihnen die Möglichkeiten gibt, alle in Netzwerken relevanten Dienste verschiedener Rechner zentral einzurichten. Dazu ist es unumgänglich, dass das grafische Anwendungsprogramm `kldap` dahin gehend erweitert wird, dass alle Objekte des Schemas erstellt und gepflegt werden können. Ferner müssen die Objekte natürlich mit einem aussagekräftigen Symbol versehen werden, damit die Navigation im Verzeichnisdienst erleichtert wird. Ferner ist es auch wünschenswert, Filter zu definieren, die die Anzeige einschränken. Auf diese Weise ist es zum Beispiel möglich, lediglich die Container und alle Benutzerobjekte einzusehen.

LDAP sollte vor allen Dingen nicht nur in der Lage sein, Konfigurationen für diverse Dienste aufzunehmen, sondern diese auch sofort umzusetzen. So muss es auch Administratoren, die nicht mit der Programmierung unter Linux vertraut sind, gelingen, die Dienste schnell und effizient aufzusetzen. Die oben angesprochenen Bindeglieder zwischen LDAP und den anderen Diensten sollten also bereits in der Software der OpenLDAP-Organisation integriert sein.

Es ist auch wünschenswert, wenn es direkt per LDAP möglich ist, das Verhalten der zu konfigurierenden Dienste zu prüfen und zu beeinflussen. Falls zum Beispiel ein

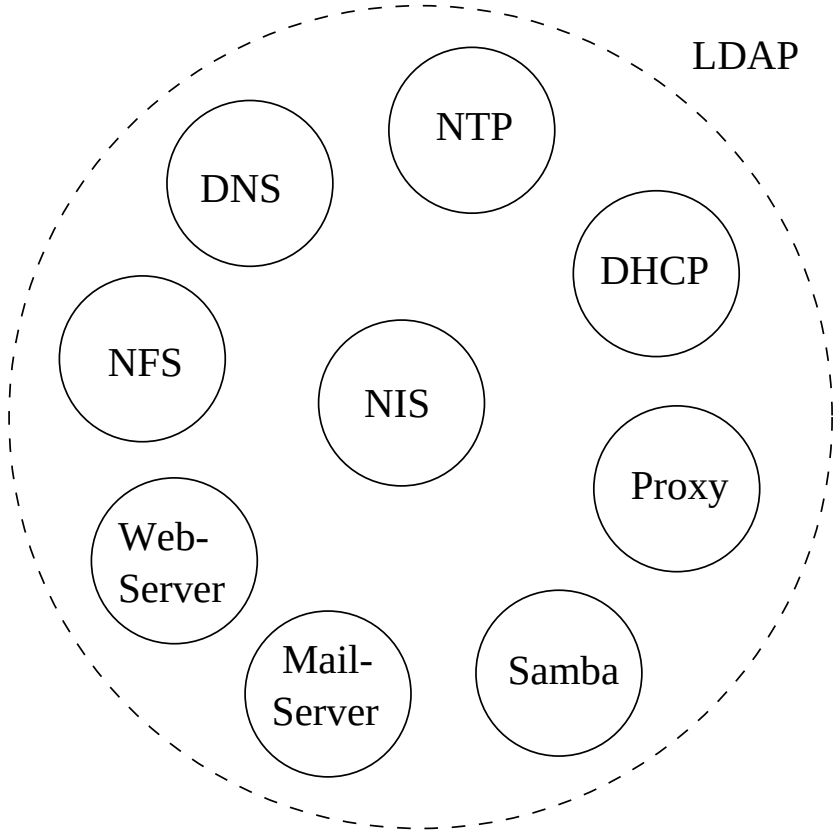

Abbildung 6.1: Integration anderer Dienste in LDAP

DHCP-Server auf einem entfernten Rechner per LDAP eingerichtet wird, so könnte dieser Server durch Ändern eines LDAP-Attributes sofort gestartet und gestoppt werden. Ferner könnte eine andere Eigenschaft anzeigen, ob der Dienst zurzeit aktiv ist.

Die in Abbildung 6.1 angegebenen Dienste realisieren wichtige Netzwerkfunktionen und sollten daher in einem Verzeichnis direkt zu konfigurieren sein.

DNS Das Domain Name System enthält eine Zuweisung von Namen zu IP-Adressen.

NTP Über NTP werden Zeit-Quellen im Netzwerk (auch im Internet) definiert, die Clients zum Justieren ihrer lokalen Zeit verwenden können.

NFS Das Network File System dient dazu, Teile des lokalen Dateisystems anderen Rechnern zur Verfügung zu stellen.

NIS Mit dem Network Information Service ist die zentrale Verwaltung von Benutzern, Gruppen usw. möglich.

DHCP Über das Dynamic Host Configuration Protocol können Rechner im Netzwerk dynamisch mit IP-Adressen und weiteren netzwerkspezifischen Parametern konfiguriert werden.

Samba Der Samba-Dienst ermöglicht es Microsoft Windows-Clients, über deren Netzwerkumgebung auf den Linux-Rechner zuzugreifen. Samba und NFS sind verwandte Dienste unterschiedlicher Protokolle.

Mail-Server Das Versenden von E-Mail wird in Netzwerken durch die Konfiguration von Mail-Servern ermöglicht.

Web-Server Web-Server dienen zur Verbreitung von HTML-Dokumenten.

Proxy Proxy-Server können die aus dem Internet abgerufenen Informationen zwischenspeichern und protokollieren.

Für die so definierten Dienste ist es natürlich sehr wichtig, dass eine gesicherte Übertragung zwischen LDAP-Server und -Client stattfindet.

Pearson Education

ADDISON-WESLEY

Glossar

ACL Access Control List; dient zum Beschreiben der Rechte an einem Objekt.

Algorithmus Verfahren.

ASCII American Standard Code for Information Interchange.

Attribut Eigenschaft eines Objekts.

bash Bourne Again Shell.

Baum Struktur der Directory-Service-Datenbank.

bind Ist die Kommunikation zwischen LDAP-Client und -Server zustande gekommen, so gelten beide Seiten als verbunden (english bind).

bin Binary; bezeichnet ein binäres Attribut.

Blattobjekt Objekt in einer Baumstruktur, das keine weiteren Objekte enthalten kann.

Broadcast Senden von Daten an alle Teilnehmer im Netzwerksegment.

ces Case exact String. Attribut, das zwischen Groß- und Kleinschreibung unterscheidet.

cis Case ignore String. Attribut, das nicht zwischen Groß- und Kleinschreibung unterscheidet.

cn ⇒ Common Name.

Common Name Attribut, das den Namen eines Blattobjekts enthält.

Containerobjekt Objekt, das seinerseits weitere Objekte enthalten kann.

Context Position im Directory Information Tree.

Country Containerobjekt; steht für ein Land.

CPU Central Processing Unit.

c ⇒ Country.

C Programmiersprache, mit der z. B. neue Zugriffsfunktionen auf den Verzeichnisdienst realisiert werden können.

DAP Directory Access Protocol. Ursprüngliches Protokoll, das für die Kommunikation mit dem Directory Service verwendet wurde.

DHCP Dynamic Host Configuration Protocol; dient zum dynamischen Zuweisen von IP-Adressen im Netzwerk.

Directory Service Englische Bezeichnung für Verzeichnisdienst.

DIT Directory Information Tree. Englische Bezeichnung für die Baumstruktur des Verzeichnisdienstes.

DNS Domain Name System; dient zum Auflösen von Rechnernamen in deren IP-Adressen.

dn Distinguished Name; gibt die Position eines Objekts im Verzeichnisdienst an.

DS ⇒ Directory Service.

Ergebnis-Code Datenpaket, das vom LDAP-Server zum LDAP-Client geschickt wird, in dem das Ergebnis der Anfrage mitgeteilt wird.

FQDN Fully Qualified Domain Name; vollständiger DNS-Name eines Rechners.

FTP File Transfer Protocol; dient zur Übertragung von Dateien im Netzwerk.

Funktion Schnittstelle einer Programmiersprache für den Zugriff auf einen Verzeichnisdienst.

Gateway Schnittstelle zwischen einem Anwendungsprotokoll wie HTTP und LDAP.

HTML HyperText Markup Language; Dateiformat von Dokumenten auf Internetseiten.

HTTP HyperText Transfer Protocol; dient zum Übertragen von HTML-Seiten im Internet.

Inkrementelle Replizierung Übertragung nur der geänderten Daten vom Master-Server zum Replica-Server.

Intervall der Replizierung Intervall, in dem die Daten vom Master- zum Replica-Server übertragen werden.

IP Internet Protocol; dient zur Kommunikation und zur Adressierung in Netzwerken.

KDE K Desktop Environment; grafische Oberfläche für Linux.

Komplette Replizierung Übertragung aller Daten vom Master- zum Replica-Server.

LAN Local Area Network. Netzwerk, das sich z. B. auf ein Firmengelände beschränkt.

LDAP Lightweight Directory Access Protocol; dient zur Kommunikation mit dem Verzeichnisdienst.

LDBM LDAP Database Format; das Datenbankformat für LDAP.

LDIF LDAP Directory Interchange Format. Textformat, mit dem LDBM-Dateien beschrieben werden können.

MAC Media Access Control.

MAN Metropolitan Area Network. Netzwerk, das sich z. B. auf eine Stadt beschränkt.

Multicast Senden von Daten an mehrere Empfänger im Netzwerk.

Namensraum Aufteilung des ⇒ DITs in Container- und Blattobjekte und deren Benennung.

Network Sniffing Mithören von Kommunikationen im Netzwerk.

NFS Network File System; dient dazu, Teile des Dateisystems anderen Rechnern im Netzwerk zur Verfügung zu stellen.

NIS Network Information System; dient z. B. zur zentralen Verwaltung von Benutzer- und Gruppenkonten im Netzwerk.

NTP Network Time Protocol; dient zur Zeitsynchronisierung.

Objektklasse Typ eines Objekts.

Objekt Eintrag in einem Verzeichnisdienst.

OID Object Identifier; nummerische Kennung eines Objekts.

OpenLDAP Freie und offene LDAP-Implementierung.

Organizational Unit Containerobjekt für eine organisatorische Einheit.

Organization Containerobjekt für eine Organisation.

ou ⇒ Organizational Unit.

o ⇒ Organization.

PAM Pluggable Authentication Modules.

Partitionierung Aufteilen der X.500-Datenbank in mehrere Teile, die dann auf unterschiedlichen Servern abgelegt werden können.

PID Process Identification; nummerische Kennung eines Linux-Prozesses.

Primfaktor Teiler einer natürlichen, ganzen Zahl, bei dem es sich um eine Primzahl handelt.

Primfaktorzerlegung Verfahren zum Bestimmen der Primfaktoren, das bei Verschlüsselungsverfahren zum Einsatz kommt.

Primzahl Natürliche, ganze Zahl, die nur durch 1 und durch sich selbst teilbar ist.

Protocol Analyzing Analysieren von Protokollen im Netzwerk.

Protokollkonverter Schnittstelle zwischen LDAP und einem Anwendungsprotokoll wie HTTP.

Protokoll Sprache, über die sich zwei Anwendungen im Netzwerk verständigen können.

Querverweis Verbindet die Partitionen eines DS-Baums.

rdn Relative Distinguished Name; gibt die relative Position eines Objekts im Verzeichnisdienst an.

referral Verbindet die Partitionen eines DS-Trees.

Regel Beschreibt, wo welches Objekt/Attribut stehen kann bzw. muss.

Replizierung Speichern der gleichen X.500-Datenbank auf mehreren Servern.

RFC Request for Comments; Dokumentation aller in Netzwerken festgelegten Standards.

Root Oberster Eintrag der Directory-Service-Datenbank.

Routine Schnittstelle einer Makrosprache zum Zugriff auf einen Verzeichnisdienst.

RPM Red Hat Package Manager. Standardformat für Software-Pakete unter Linux.

RSA Verschlüsselungssystem, das nach seinen Entwicklern Rivest, Shamir und Adleman benannt ist.

Runlevel Konfiguration des Linux-Rechners.

Schema Struktur eines Verzeichnisdienstes mit allen möglichen Einträgen und Eigenschaften.

scp Secure Copy; dient zur sicheren Übertragung von Dateien im Netzwerk.

Shellskript Verarbeitungsdatei in der Linux-Shell `bash`.

slapd Standalone LDAP Daemon; realisiert den LDAP-Server.

slurpd Standalone LDAP Update Replication Daemon; realisiert die Replizierung.

SMTP Simple Mail Transfer Protocol; dient zum Senden von E-Mails.

sn Surname; Attribut eines Personen-Objekts.

ssh Secure Shell; dient zum sicheren Fernwarten anderer Rechner im Netzwerk.

Subtree Alle Objekte unterhalb eines Containers.

Suffix Beschreibt den Subtree, dessen Daten ein LDAP-Server in seiner Datenbank gespeichert hat.

TCP Transmission Control Protocol; dient zur kontrollierten Kommunikation in IP-Netzwerken.

Teilbaum Alle Objekte unterhalb eines Containers.

Telnet Terminal Emulation Protocol; dient zum Fernwarten von Rechnern.

tel Telephone Number String; Attribut-Typ, der eine Telefon-Nummer beschreibt.

Tree Baumartige Struktur der Directory-Service-Datenbank.

UDP User Datagram Protocol; dient zur unkontrollierten Kommunikation in IP-Netzwerken.

ud LDAP User Directory; Anwendungsprogramm zum Bearbeiten von Benutzer-Objekten.

UFN User Friendly Notation; dient zur benutzerfreundlichen Angabe von Objekten.

Unicast Senden von Daten im Netzwerk an exakt einen Empfänger.

URL Uniform Resource Locator; dient z. B. zum Lokalisieren eines LDAP-Elements.

Verzeichnisdienst Verteilte Datenbank zum Festhalten von Informationen und zum Konfigurieren diverser Dienste.

vi Visual Editor; dient zum Editieren von Textdateien.

WAN Wide Area Network. Netzwerk, das sich über mehrere Standorte erstreckt.

Wurzel Oberster Eintrag der Directory-Service-Datenbank; wird auch als Root bezeichnet.

WWW World Wide Web.

X.500 Standardaufbau von Verzeichnisdiensten.

YaST Yet another Setup Tool; wird in SuSE-Linux zur Installation und Administration verwendet.

YP Yellow Pages. YP war die ursprüngliche Bezeichnung des NIS-Dienstes, die jedoch aus rechtlichen Gründen nicht verwendet werden durfte.

Pearson Education

ADDISON-WESLEY

Literaturverzeichnis

[1] Ph. D. Timothy A. Howes, Mark C. Smith, Gordon S. Good

Understanding and Deploying LDAP Directory Services, Macmillan Technical Publishing USA, 1999, ISBN 1-57870-070-1

[2] Jens Banning

DHCP unter Linux, Addison-Wesley, 2000, ISBN 3-8273-1731-2

[3] Christian Huitema

IPv6 Die neue Generation, Addison-Wesley, 2000, ISBN 3-8273-1420-8

[4] Jochen Hein

Linux Systemadministration, Addison-Wesley, 1999, ISBN 3-8273-1510-7

[5] Michael Kofler

Linux. Installation, Konfiguration, Anwendung, Addison-Wesley, 1999, ISBN 3-8273-1475-5

[6] Dr. Bernhard Röhrig

Linux im Netz, Computer & Literatur, 1999, ISBN 3-932311-61-2

[7] Peter Samulat

Linux-Server im kommerziellen Netzwerk, Addison-Wesley, 2000, ISBN 3-8273-1631-6

[8] Helmut Herold

Linux/Unix Grundlagen, Addison-Wesley, 1999, ISBN 3-8273-1435-6

[9] J. Postel

RFC 0768: User Datagram Protocol, 1980

[10] J. Postel

RFC 0791: Internet Protocol, 1981

[11] J. Postel

RFC 0793: Transmission Control Protocol, 1981

[12] P. Barker, S. Kille

RFC 1274: The COSINE and Internet X. 500 Schema, 1991

[13] S. E. Hardcastle-Kille

RFC 1275: Replication Requirements to provide an Internet Directory using X.500, 1991

[14] S. E. Hardcastle-Kille

RFC 1279: X.500 and Domains, 1991

[15] C. Weider, J. Reynolds

RFC 1308: Executive Introduction to Directory Services Using the X.500 Protocol, 1992

[16] C. Weider, J. Reynolds, S. Heker

RFC 1309: Technical Overview of Directory Services Using the X.500 Protocol, 1992

[17] S. Hardcastle-Kille, E. Huizer, V. Cerf, R. Hobby, S. Kent

RFC 1430: A Strategic Plan for Deploying an Internet X.500 Directory Service, 1993

[18] T. Howes

RFC 1558: A String Representation of LDAP Search Filters, 1993

[19] P. Barker, S. Kille, T. Lenggenhager

RFC 1617: Naming and Structuring Guidelines for X.500 Directory Pilots, 1994

[20] W. Yeong, T. Howes, S. Kille

RFC 1777: Lightweight Directory Access Protocol, 1995

[21] T. Howes, S. Kille, W. Yeong, C. Robbins

RFC 1778: The String Representation of Standard Attribute Syntaxes, 1995

[22] S. Kille

RFC 1779: A String Representation of Distinguished Names, 1995

[23] S. Kille

RFC 1781: Using the OSI Directory to Achieve User Friendly Naming, 1995

[24] A. Young

RFC 1798: Connection-less Lightweight X.500 Directory Access Protocol, 1995

[25] T. Howes, M. Smith

RFC 1823: The LDAP Application Program Interface, 1995

[26] T. Howes, M. Smith

RFC 1959: An LDAP URL Format, 1996

[27] T. Howes *RFC 1960: A String Representation of LDAP Search Filters*, 1996

[28] F. Yergeau *RFC 2044: UTF-8, a Transformation Format of Unicode and ISO 10646*, 1996

[29] S. Kille

RFC 2164: Use of an X.500/LDAP directory to support MIXER address mapping, 1998

[30] T. Genovese, B. Jennings

RFC 2218: A Common Schema for the Internet White Pages Service, 1997

[31] S. Kille, M. Wahl, A. Grimstad, R. Huber, S. Sataluri

RFC 2247: Using Domains in LDAP/X.500 Distinguished Names, 1998

[32] M. Wahl, T. Howes, S. Kille

RFC 2251: Lightweight Directory Access Protocol (v3), 1997

[33] M. Wahl, A. Coulbeck, T. Howes, S. Kille

RFC 2252: Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions, 1997

[34] M. Wahl, S. Kille, T. Howes

RFC 2253: Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names, 1997

[35] T. Howes

RFC 2254: The String Representation of LDAP Search Filters, 1997

[36] T. Howes, M. Smith

RFC 2255: The LDAP URL Format, 1997

[37] M. Wahl *RFC 2256: A Summary of the X.500(96) User Schema for use with LDAPv3*, 1997

[38] F. Yergeau

RFC 2279: UTF-8, a Transformation Format of ISO 10646, 1998

[39] S. Kille

RFC 2293: Representing Tables and Subtrees in the X.500 Directory, 1998

[40] S. Kille

RFC 2294: Representing the O/R Address hierarchy in the X.500 Directory Information Tree, 1998

[41] L. Howard

RFC 2307: An Approach for Using LDAP as a Network Information Service, 1998

[42] A. Grimstad, R. Huber, S. Sataluri, M. Wahl

RFC 2377: Naming Plan for Internet Directory-Enabled Applications, 1998

[43] Manual Pages

[44] HOWTOs

[45] FAQs

Stichwortverzeichnis