Leibniz-Rechenzentrum, München

Einführung in die Systemverwaltung unter UNIX

Dr. Ernst Bötsch / Dr. Petra Eilfeld

5. Überarbeitete Auflage (September 2005)

http://www.lrz.de/services/schulung/admks/

Themengebiete: Übersicht

- Umfeld und allgemeine Aufgaben
- Stand-alone-Betrieb
- Netzaspekte
  - Name Service Switch (NSS)
  - Pluggable Authentication Modules (PAM)
  - Netzgrundlagen
  - Lightweight Directory Access Protocol (LDAP)
  - Network File System (NFS)
- Security-Grundlagen
- Workshop

Name Service Switch (NSS)
(Daniel Schedel)

Inhalt

- Motivation und Historie
- Funktion
- `/etc/nsswitch.conf`

NSS: Motivation und Historie (1)

- Problem: Wie kommt eine Anwendung zu den Daten in der Konfigurationsdatei ?
- Lösungen:
  - Die Anwendung liest die Datei und interpretiert den Inhalt *selbst*.
  - Die Anwendung verwendet Bibliotheksfunktionen, die dies für sie erledigen.
- Beispiele für Bibliotheksfunktionen:
  - `/etc/passwd`: `getpwent` (3), `getpwnam` (3), `getpwuid` (3)
  - `/etc/shadow`: `getspent` (3), `getspnam` (3)
  - `/etc/group`: `getgrent` (3), `getgrnam` (3), `getgrgid` (3)
  - `/etc/hosts`: `gethostbyname` (3), `gethostbyaddr` (3)

NSS: Motivation und Historie (2)

- Problem 2: Was machen die Bibliotheksfunktionen, wenn es neue Datenquellen gibt ?
- Lösungen zu Problem 2:
  - Die Bibliotheksfunktionen werden “aufgebohrt”, wobei die Datenquellen in einer festen Reihenfolge konsultiert werden.
  - Der Zugriff auf die Datenquellen wird von den Bibliotheksfunktionen entkoppelt; man kann konfigurieren, welche Datenquellen in welcher Reihenfolge ausgewertet werden.

    ⇨ Name Service Switch (NSS)

NSS: Funktion

- Umschalten zwischen verschiedenen Arten der Namensauflösung ohne Neu-Übersetzung oder Austausch von Programmdateien
- NSS stellt Wissen zur Verfügung, welche Benutzerkennungen bekannt sind, welcher Benutzer zu welcher Gruppe gehört u.v.m.
- Mit NSS Erhalt von Benutzer- und Systeminformationen aus verschiedenen Datenquellen möglich
- Für viele Datenquellen sind entsprechende Module verfügbar.
- Beispiel für Namensauflösungen: Login ➔ UID

NSS: `/etc/nsswitch.conf` (1)

- Beispiel:

```
passwd:   files  ldap
shadow:   files  ldap
group:    files  ldap
hosts:    dns [!UNAVAIL=return] files
```

- Erstes Feld definiert die (logische) Datenbasis
  - Sinnvolle Voreinstellungen für nicht explizit aufgeführte Datenbasen
- Rest der Zeile legt den Lookup-Prozess fest.
  - Syntax:

    Quelle (([(! ? *Status = Aktion*)+])? *Quelle*)*
  - Lookup-Code vom Modul `/lib/libnss_`*Quelle*`.X` bereitgestellt.

NSS: `/etc/nsswitch.conf` (2)

- Die Keywords bei *„Status“* und *„Aktion“* sind case-insensitive.
- Mögliche Werte für *„Aktion“:*

`return`	Ende der Namensauflösung
`continue`	Abfrage der nächsten Quelle

- Mögliche Werte für *„Status“*
  (voreingestellte *„Aktion“* in Klammern):

`SUCCESS`	Dienst OK, Name gefunden (`return`)
`NOTFOUND`	Dienst OK, Name nicht gefunden (`continue`)
`UNAVAIL`	Dienst dauerhaft nicht verfügbar (`continue`)
`TRYAGAIN`	Dienst temporär nicht verfügbar (`continue`)

Pluggable Authentication Moduls (PAM) *(Daniel Schedel)*

Inhalt

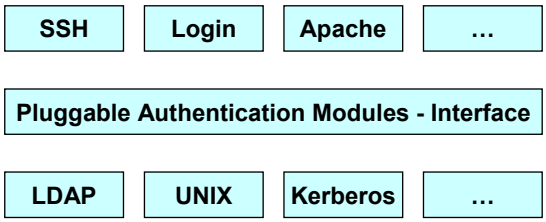
- Motivation und Ziele
- Architektur
- Aktionen und Rückgabewerte
- Passwörter
- Authentifizierung
- PAM und NSS

PAM: Motivation und Ziele

- Flexibler Mechanismus für Authentifizierung, Session, Passwort- und Account-Management
- Zentrale Bibliothek für Authentifizierung
- Konfiguration mittels Textdateien ➔ Anpassung der Konfigurationsdateien ohne Neucompilierung der Anwendung
- Modularer Aufbau für einfache Anpassung
- PAM trennt Anwendungs-Code von Authentifizierungs-Code durch Schnittstelle
- Administrator bestimmt Authentifizierungsmechanismus
- Methoden in PAM haben Zugriff auf Benutzerdaten und können sich des NSS-Mechanismus bedienen, um Informationen zu erhalten

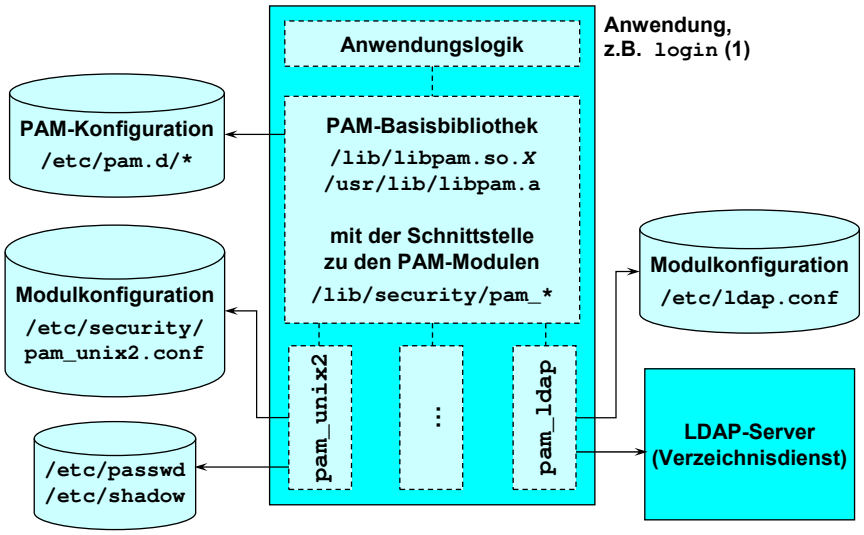
PAM: Modell

- PAM Schichtenmodell:



- Konfigurationsübersicht:
  - `/etc/pam.conf` Konfigurationsdatei für alle Anwendungen
  - `/etc/pam.d/*` Konfigurationsdateien der Anwendungen
  - `/lib/security/pam_*` Bibliotheksmodule
  - `/etc/security/*` Konfigurationsdateien der Module

PAM: Architektur



12

PAM: Syntax (1)

Syntax: *Service Task Control Module-Path Module-Args*

- Dienstname (*„Service"*): `login`, `su`, `passwd`, `sshd`
- Aufgabenbereich (*„Task"*):
  - `auth` Benutzeridentifizierung
  - `session` Verwaltung der Sitzung, wie z.B. Limits, Rechte
  - `account` Verwaltung des Accounts
  - `password` Steuerung der Passwortänderung
- Steuerung der Abarbeitung (*„Control"*):
  - `required` Modul muss zwingend durchlaufen werden
  - `sufficient` Erfolg des Moduls gilt als ausreichend
  - `requisite`, `optional`

PAM: Syntax (2)

- Absoluter Pfad des Moduls (*„Module-Path"*)
- Spezifische Argumente für das Modul (*„Module-Args"*):
  - `debug` ⇨ Ausgabe der Diagnoseinformationen
  - `no_warn` ⇨ Keine Warnungsausgabe
  - `use_first_pass` ⇨ Verwendet Passwort des vorhergehenden Moduls
  - `expose_account` ⇨ Keine Informationen über Benutzerkonto
- Details: `pam` (8) = `pam.conf` (8) = `pam.d` (8)

PAM: Syntax (3)

Erweiterte Syntax von *„Control":* [(*Value = Action*)+]

- Mögliche Rückgabewerte (*„Value"*) des Moduls (u.a.):
  - `success`: Test erfolgreich
  - `ignore`: Dieses Modul ignorieren
- Aktionen (*„Action"*):
  - `ok`: Erfolg, weiter prüfen
  - `done`: Erfolg, zurück zur Anwendung
  - `bad`: Fehlschlag, weiter prüfen
  - `die`: Fehlschlag, zurück zur Anwendung
  - `ignore`: Ergebnis ignorieren, weiter prüfen
  - `reset`: Alle vorherigen Ergebnisse ignorieren, weiter prüfen

PAM: Passwörter

- Passwort Aging: `pam_unix`, `pam_unix2`
  - Modul für Shadow-Passwörter
  - Konfiguration über `chage` (1)
  - Mögliche Parameter:
    - Maximale / Minimale Gültigkeit eines Passworts
    - Datum der letzten Passwortänderung
    - Ablaufdatum des Accounts
- Passwort-History: `pam_unix`, `pam_unix2`
  - Alte Passwort-Hashes können gespeichert werden
  - User können alte Passwörter nicht mehr verwenden

PAM: Authentifizierung

- über `/etc/pam.d/login` Anforderungen für ein erfolgreiches Login steuerbar
- PAM-Konfigurationsbeispiel:

```
auth required /lib/security/pam_securetty.so
auth required /lib/security/pam_smb_auth.so
auth required /lib/security/pam_nologin.so
account required /lib/security/pam_pwdb.so
password required /lib/security/pam_cracklib.so
password required /lib/security/pam_pwdb.so
                  shadow nullok use_authtok
session required /lib/security/pam_pwdb.so
```


NSS und PAM: Zusammenfassung

- NSS nimmt Namensauflösungen vor
- PAM authentifiziert User
- Verzeichnisdienste
  - haben vollständige Informationen
  - können von PAM und NSS verwendet werden

Netzgrundlagen

Inhalt

- Grundbegriffe
- IP-Adressklassen
- Regeln für IP-Adressen
- Schichtenmodell (TCP/IP)
- Remote Procedure Call (RPC)
- Portmapper
- Vorbereitung der Netzanbindung
- Netzkonfiguration und -diagnose

Grundbegriffe: Ethernet-Adressen

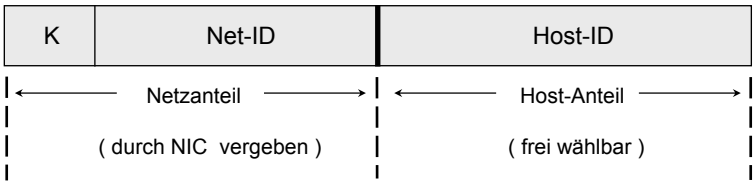
Basiskomponenten:

- Ethernet mit CSMA / CD
  Übergang auf Fast-Ethernet bzw. GB-Ethernet
- Ethernet-Adressen: weltweit eindeutig; 48 Bit, fest in HW eingetragen, pro Interface 1 Adresse
- Systemadministratoren haben (normalerweise) mit ET-Adressen nichts zu tun; nur bei Diskless-Boot, DHCP, etc. interessant
- Broadcast-Adresse: FF:FF:FF:FF:FF:FF
- Promiscuous Mode: abhören aller Pakete auf dem Netz (z.B. für Netzanalysatoren)

Grundbegriffe: IP-Adressen (1)

- für Administratoren interessant:
  IP-Adressen (Internet-Protocol) ⇨ `hosts` (4)

Aufbau einer IP-Adresse (32 Bit, ohne Sub-Netting):



K = Kennfeld der Adressklasse
Bsp.: 129.187.10.86 (dec) = 81.BB.0A.56 (hex)

ICANN : **I**nternational **C**orporation for **A**ssigned **N**ames and **N**umbers
(früher: **NIC**)

Grundbegriffe: IP-Adressen (2)

Achtung:

Die IP-Adresse identifiziert einen Rechner als *Objekt im Netz*.

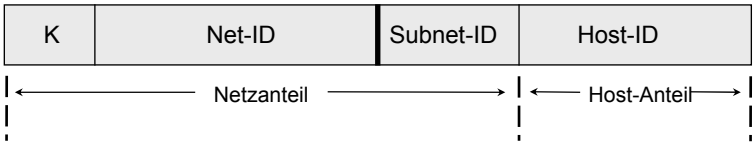
Konsequenzen:

- einfaches Routing (nur Netzanteil muß ausgewertet werden)
- mindestens 1 IP-Adresse für jedes (Sub-)Netz, an das der Rechner angeschlossen ist
- bei Wechsel in ein anderes (Sub-)Netz ändert sich die IP-Adresse

Grundbegriffe: Sub-Netting (1)

- Ein Netz kann mit Hilfe von Subnetzen strukturiert werden.

Aufbau einer IP-Adresse (32 Bit, mit Sub-Netting):



Die ersten n Bits der Host-ID werden als Subnet-ID interpretiert.
(bei Class-B-Adressen meist 1 Byte, d.h. 8 Bit)

Grundbegriffe: Sub-Netting (2)

Vorteile von Sub-Netting:

- Netzverkehr kann überwiegend lokal gehalten werden
  - ⇨ Last-Trennung
  - ⇨ Verkehrs-Separierung
- Administration wird übersichtlicher und einfacher
- Administration kann an organisatorischen Strukturen orientiert werden
- Fehler / Einbrüche können meist auf 1 Subnetz begrenzt werden.

Grundbegriffe: Die Netmask

- Die Netmask definiert Netzanteil (einschließlich Sub-Netting).
  ⇨ `ifconfig` (1M)

- Beispiele:
  - Class-B ohne Sub-Netting FF.FF.00.00
  - Class-B mit Sub-Netting (1 Byte) FF.FF.FF.00
  - Class C FF.FF.FF.00

Achtung:

Für die Host-ID stehen nur die nicht durch " FF " (d.h. durch binär " IIII IIII ") ausgeblendeten Adressen-Bytes zur Verfügung.

Grundbegriffe: Kommunikation mit Rechnern

- Broadcast-Adressen:

  Um *alle* Rechner eines (Sub-)Netzes kontaktieren zu können, ohne die einzelnen IP-Adressen kennen zu müssen, benötigt man Broadcast-Adressen.
  - BSD (älter) Host-ID besteht nur aus " 0 " (Binär-Darstellung)
  - SysV (neuer) Host-ID besteht nur aus " I " (Binär-Darstellung)

- Der Port

  spezifiziert die ID einer *verteilten Anwendung* *innerhalb* eines Rechners ⇨ `services` (4)

Grundbegriffe: Internet-Namen

- weltweit eindeutiger, baumartiger Namensraum
- Überwachung durch NIC (USA)
- Spezifität der Namensbestandteile nimmt von links nach rechts ab
- Beispiele:

```
sun3.lrz-muenchen.de
planck.t30.physik.tu-muenchen.de
peanuts.pst.informatik.uni-muenchen.de
cs.ucl.ac.uk
cs.washington.edu
```


IP-Adressklassen

IPv4 (jetziges System)	Klasse A	Klasse B	Klasse C
Kennfeld	0 (1 Bit)	10 (2 Bit)	110 (3 Bit)
Wert des 1. Byte	1 – 126	128 – 191	192 - 223
Länge der Net-ID mögliche Zahl Netze	7 Bit 126	14 Bit 64 * 254 = 16 256	21 Bit 32 * 2542 = 129 024
Länge der Host-ID mögliche Zahl Rechner	24 Bit 254^3 = 16 387 064	16 Bit 254^2 = 64 516	8 Bit 254

Regeln für IP-Adressen (1)

Die Loopback-Adresse:

- ermöglicht einem Rechner, mit sich selbst direkt zu kommunizieren
  - ⇨ Standard: 127.0.0.1.
    - ☺ Vereinheitlichung:
      Rechner kann sich selbst genauso adressieren wie andere Rechner auch
    - ☺ Testzwecke:
      " Selbstversuch ", bevor Test auf's Netz ausgedehnt wird

Minimaleinträge ⇨ `/etc/hosts`

- alle eigenen Adressen (wegen Loopback-Adresse mindestens 2)
- alle wichtigen Rechner (z.B. *loghost*)

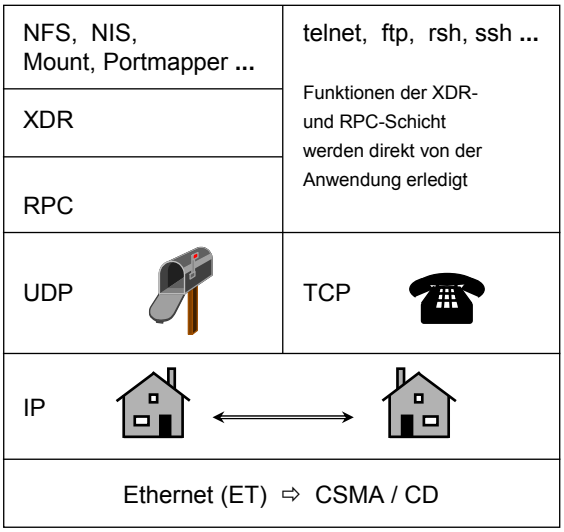
Regeln für IP-Adressen (2)

- 1. Byte des Netzanteils:
  - 0, 127 und 255 ⇨ Spezialzwecke (Broadcast, Loopback)
  - 224 - 254 ⇨ reserviert für zukünftige Nutzung

- Bytes der Net-ID, Subnet-ID und Host-ID dürfen nicht nur aus "0"en oder aus "1"en bestehen.
  Gilt auch für Teile von Bytes, deren eine Hälfte z.B. zur Net-ID, die andere zur Subnet-ID gehört.

- Die Host-ID muß mindestens 2 Bit lang sein (sonst Broadcast-Adresse !) .

Schichtenmodell



31

Schichtenmodell: Erläuterungen

Schicht	Funktionalität
NFS, telnet etc.	Eigentliche Anwendung
XDR	Umwandlung von Datentypen
RPC	Remote-Prozeduraufrufe
UDP / TCP	Kommunikation zwischen Anwendungen
IP	Kommunikation zwischen Rechnern
Ethernet	"gelbes Kabel" (Basis-HW + minimale SW)

Schichtenmodell: Unterschiede UDP ⇔ TCP

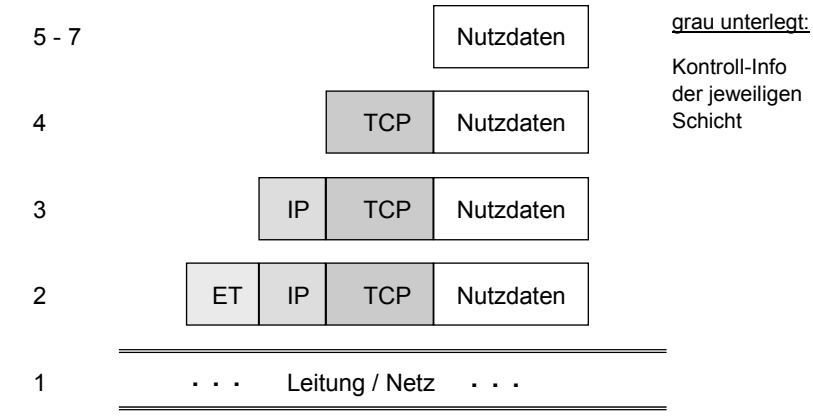
UDP

- ❒ unabhängige Pakete
- ❒ beliebige Reihenfolge
- ❒ Vervielfachung / Verlust möglich
- ❒ keine Leitungsproblem-Diagnose !
- ❒ keine Fehlerbehandlung ⇨ muß Anwendung selbst erledigen
- ❒ schneller, einfacher

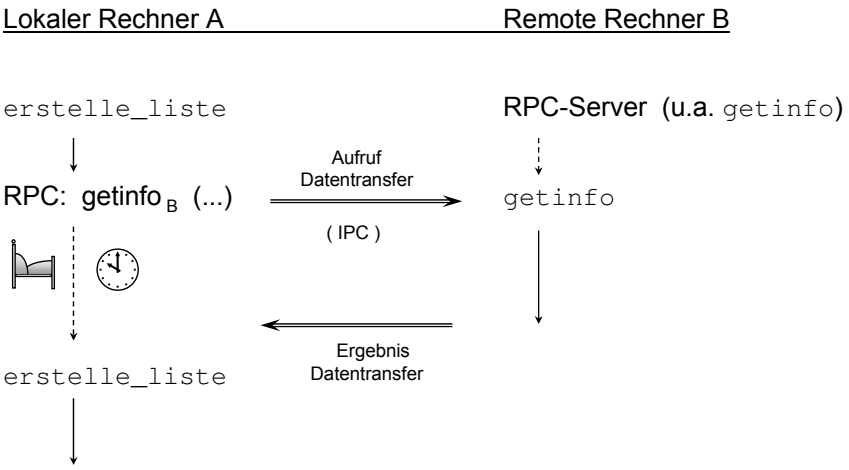
TCP

- ❒ stehende Verbindung
- ❒ Reihenfolge bleibt erhalten
- ❒ Pakete kommen genau 1 mal an oder Fehlermeldung
- ❒ Leitungszusammenbruch möglich, wird aber erkannt
- ❒ interne Fehlerbehandlung bzw. Meldung über Leitungszusammenbruch
- ❒ Overhead durch "Buchhaltung"

Schichtenmodell: Encapsulation



Remote Procedure Call (RPC)



Der Portmapper

- Zweck:
  Identifizierung verteilter Anwendungen innerhalb eines Rechners über Portnummern
  - ⇨ können statisch ("per Hand") oder dynamisch (Portmapper) vergeben werden
- Vorteile:
  - flexibel; schonender Umgang mit Ports
  - 1 Dienst kann kollisionsfrei von mehreren Servern erbracht werden
- Nachteile:
  - bei Portmapper-Ausfall ist kein RPC-Dienst erreichbar
  - nach Neustart des Portmappers Neustart aller RPC-Dienste nötig

Vorbereitung der Netzanbindung (1)

Folgende Informationen bzw. Vergabestrategien müssen unbedingt vor der Netzanbindung vorliegen bzw. geklärt werden:

- Name und IP-Adresse des Rechners
- Zeitzone (Sun: MET; HP: MEZ-1MESZ; Cray: NFT)
- Rechnerausstattung bzw. -Funktion, z.B.:
  - Diskless-Server / -Client bzw. Stand-alone
  - Dataless-Server / -Client
  - ohne / mit LDAP / NIS
  - LDAP-Server / -Client
  - NIS-Master / -Slave / -Client

Vorbereitung der Netzanbindung (2)

Notwendige Informationen (Fortsetzung):

- LDAP-Server und evtl. erforderliche Zertifikate
- NIS-Domain
- Log-Host für `syslog` (3)
- Mail-Host zur Weiterleitung der Mail
- Internet-Domain
- Nameserver
  - ⇨ Primärer Server
  - ⇨ Backup-Server

Achtung: Bei allen Rechnern immer Name *und* IP-Adresse notieren

Netzkonfiguration

⇨ `hosts` (4), `hosts.equiv` (4) / `.rhosts` (4)

- Mindesteintrag für `/etc/hosts` (bei NIS nur während Boot):
  - `127.0.0.1  localhost`
  - *IP_Adress Host* `loghost`
- Rechnername in `/etc/hostname.le0`
  ⇨ setzen mit " hostname Host " in `/etc/rc.boot`
- IP-Adresse ⇨ `ifconfig` (1M)
- Netzmaske und Broadcast-Adresse in `/etc/rc.local`
  ⇨ " `ifconfig -a netmask + broadcast +` "
- Nameserver-Anbindung in `/etc/resolve.conf`:

```
domain  lrz-muenchen.de
nameserver  IP_Adress_1  #Host_1
nameserver  IP_Adress_2  #Host_2
```


Netzdiagnose

- Eigene Konfiguration ⇨ `ifconfig -a` (Sun)
  ⇨ `ifconfig interface` (HP)
- Erreichbarkeit ⇨ `ping` (1M)
- Routing ⇨ `netstat -rn`
- bestehende Kommunikationsverbindungen ⇨ `netstat -an`
- Ethernet-Adresse ⇨ `ping` *Host* ; `arp -a`
- Nameserver ⇨ `nslookup` (1M)
  - IP-Adresse ⇨ *Host* bzw. " `nslookup` *Host* " als Kommando
  - Name ⇨ `set q=ptr` und (z.B.)
    `86.10.187.129.in-addr.arpa` (IP-Adressbestandteile rückwärts !)

Lightweight Directory Access Protocol (LDAP) *(Daniel Schedel)*

Inhalt

- Motivation und Ziele
- Architektur und Modelle
- Sicherheitsaspekte: ACL, Verschlüsselung, Authentifizierung
- Distinguished Names und Relative Distinguished Names
- Klassen, Attribute, Schemas
- LDIF-Dateiformat
- Benutzerauthentifizierung
- Installation eines OpenLDAP-Servers
- Anpassen der Konfigurationsdateien und Directory ACLs
- Arbeiten mit LDAP
- Fazit

LDAP: Motivation (1)

- Problem: Verteilte Umgebungen
  - Zugriff auf Ressourcen werden notwendig
    - im WAN,
    - bei Projektpartnern
    - im Internet
  - Wunsch nach zentraler Datenhaltung
  - UNIX-Systeme benötigen umfangreiche Konfigurations-Datenbasen
    - /etc/passwd
    - /etc/shadow
- **Wichtig:** Datenbestände sollten auf allen Rechnern (weitestgehend) identisch sein

LDAP: Motivation (2)

- Lösung: Verzeichnisdienste
  - Verzeichnis:
    Auflistung von Informationen über Objekte in einer bestimmten Ordnung;
    Abfragen von Detailinformationen zu jedem Objekt
  - Verzeichnisdienst:
    Speicherung und Abruf von Informationen über Benutzer, Gruppen und diversen Konfigurationsdaten

LDAP: Ziele

- Die Ziele
  - Vereinheitlichte Datenhaltung
  - Zentrale Verwaltung der Informationen
  - Konsistenz in der Schnittstelle zum Benutzer
  - Konsistenz in den Richtlinien für das Netzmanagement
  - Konsistenz in den Security Policies
- werden erreicht durch:
  - einfaches Protokoll
  - genormte Schnittstellen
  - verteilte Architektur (Replikation)

LDAP: Architektur

- Kommunikation bei LDAP basiert auf TCP/IP
- Ressourcenschonend für Netzinfrastruktur
- LDAP speichert Informationen in Baumstruktur, wie z.B.:
  - Benutzer, Gruppen
  - Boot-Informationen
  - Mountpoints für Dateisysteme
- Verwendung von Objekten mit Eigenschaften zur Strukturierung (Klassen-Beziehung)

LDAP: Modelle (1)

- Informationsmodell
  - Directory Information Tree (DIT)
  - Einträge gehorchen Struktur:
    - Objektorientiert (Klassen, Attribute)
    - Erweiterbar
- Namensmodell
  - Relative Distinguished Name (RDN):
    Attribut, durch das sich ein Objekt eindeutig von allen Geschwisterobjekten unterscheiden muss.
  - Distinguished Name (DN):
    Identifiziert jedes Objekt im DIT eindeutig.
  - Regeln zur Generierung des DN aus RDNs
    (Kombination aus RDNs ergibt eindeutigen String)

LDAP: Modelle (2)

- Funktionsmodell

  Aktionen / Operationen, die LDAP zur Verfügung stellt
- Sicherheitsmodell
  - Authentifizierung (Binding)
  - Autorisierung (ACL)
  - Verschlüsselung während der Datenübertragung

LDAP: Informationsmodell (1)

- Directory Information Base (DIB):
  Alle Informationen in einem Verzeichnis
- DIB aus einzelnen Entries (Informationen über ein Objekt) aufgebaut
- Entries beschreiben Objekte (Instanzen einer Klasse) und bestehen aus Attributen.
- Jedes Attribut hat einen Typ mit einem oder mehreren Werten.

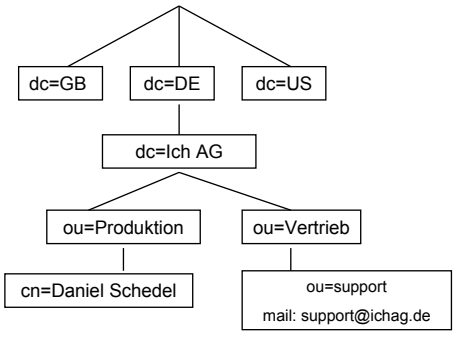
LDAP: Informationsmodell (2)

- Eigenschaften von Objekten abhängig von der Zugehörigkeit zu einer Klasse
- Klassen für Personen enthalten u.a. folgende Attribute:
  - **objectClass** (Klasse selbst), **sn** (Nachname) obligatorisch
  - **cn** (common name), **description** etc. optional
- Schemas: Definition von Klassen und Attributen
- Eigene Definition von Schemas möglich !

LDAP: Namensmodell

- DN aus RDNs
- Durch Hierarchie im DIT repräsentiert

dc: Domain Name Component
ou: Organizational Unit
cn: Common Name



LDAP: DN / RDN (1)

- Analogien:
  - RDN ähnlich einem Dateinamen (Basename)
  - DN ähnlich dem absoluten Pfad zwischen der Wurzel eines Dateisystems und einer Datei
- RDN kann aus mehreren Attributen bestehen
  - Beispiel: `sn=Mueller + ou=Vertrieb`
- Folgende Sonderzeichen müssen mit „\“ gequotet werden:
  - Leerzeichen oder „#“ am Anfang
  - Leerzeichen am Ende
  - „,“, „+“, „"“, „\“, „<“, „>“, „;“

LDAP: DN / RDN (2)

- DN normalisieren:
  - Entfernen aller überflüssigen Leerzeichen in der „=-Umgebung“ bei allen RDNs
  - Kontrolle, ob alle Sonderzeichen gequotet sind
  - Entfernen aller überflüssigen Leerzeichen in der „+-Umgebung“ bei allen mehrwertigen RDNs
  - Entfernen aller überflüssigen Leerzeichen bei den RDNs
- Beispiel:
  - Vorher:
    `cn=gerald carter + ou=sales, dc=plainjoe ,dc=org`
  - Nachher:
    `cn=gerald carter+ou=sales,dc=plainjoe,dc=org`

LDAP: DN / RDN (3)

- String-Repräsentation ist case-preserving
- Zwei DNs werden case-insensitive verglichen
- Verhalten beruht auf Anpassungs- und Syntaxregeln von Attributtypen
- Dennoch zur Sicherheit auf konsistente Schreibweise achten !

LDAP: Sicherheitsmodell (1)

- Wichtige Sicherheitskonzepte bei LDAP:
  - **Geheimhaltung** der Kommunikation zwischen Client und Server durch Transport-Layer-Security (TLS)
  - **Authentifizierung** des Client und/oder Servers bei der Bind-Operation
  - **Autorisierung** durch Access Control Lists (ACLs) beim Datenbestand
- Protokollierung
- Unterstützung des TCP-Wrappers bei OpenLDAP
  - ⇨ Anpassung erforderlich: `/etc/hosts.allow` und/oder `/etc/hosts.deny`

LDAP: Sicherheitsmodell (2)

- "Netgroups" (für Fortgeschrittenere):
  - Netgroups erlauben, Geräte und/oder Benutzer für administrative Zwecke in eine Gruppe zusammenzufassen
  - Konzept der "Netgroups" von NIS übernommen
  - Vor Eintragen von Netzgruppen die `ou=netgroup` erstellen
  - `/etc/ldap.conf` anpassen (`nss_base_netgroup` Parameter)
  - In Eintrag in `/etc/nsswitch.conf`:
    `netgroup: ldap`

LDAP: Klassen, Attribute, Schemas (1)

- LDAP-Schema: alle möglichen Typen von Einträgen mit deren Attributen
- Einträge im Schema: Klassen
- Klassen hierarchisch aufgebaut
  - Wurzel des Baumes: `top`
- Schema: Definition von Klassen und/oder Attributen
- Standardschemas (`/etc/openldap/schema/*`):
  - `core.schema` (obligatorisch)
  - `cosine.schema` (optional)
  - `nis.schema` (optional)

LDAP: Klassen, Attribute, Schemas (2)

- Klassen und Attribute
  - Einfache Vererbung (Keyword „SUP“)
  - Optionaler Kommentar (Keyword „DESC“)
- Klasse:
  - Keyword „objectclass“
  - Struktur eines Objekts
  - Besteht aus Attributen
  - Liste obligatorischer Attribute (Keyword „MUST“)
  - Liste optionaler Attribute (Keyword „MAY“)

LDAP: Klassen, Attribute, Schemas (3)

- Arten von Klassen:
  - Basisklassen (Keyword „STRUCTURAL“)
    - Modellierung von Objekten der „realen Welt“ (Personen, Geräte,…)
    - Mindestens ein Attribut obligatorisch: RDN
  - Hilfsklassen (Keyword „AUXILIARY“)
    - Erweiterung einer Basisklasse um zusätzliche Attribute
  - „Abstrakte Klassen“
    - Nutzung nur als „Urahnen“ in der Vererbung (Beispiel: „top“)

LDAP: Klassen, Attribute, Schemas (4)

- Attribut
  - Keyword „`attributetype`“
  - Definierter Datentyp / Kodierungsregel (Keyword „`SYNTAX`“) zum Speichern und Übertragen von Werten eines Attributtyps
  - Ein (Keyword „`SINGLE-VALUE`“) oder mehrere Werte (Default) gleichen Typs pro Attribut
  - Regeln für Vergleiche
    - Gleichheit (Keyword „`EQUALITY`“)
    - Substrings (Keyword „`SUBSTR`“)

LDAP: LDIF (1)

LDAP Data Interchange Format (LDIF)

- Standardisiertes textbasiertes Format für die Datensätze
- Primäre Verwendung: Datenaustausch, Backup, Manipulation der Datensätze durch externe Programme
- Besteht aus Folge von Attributen und Wertepaaren:
  - Schlüsselfeld:

```
dn: cn=Daniel Schedel,dc=admks,dc=lrz-muenchen,dc=de
```

  - Klasse(n):

```
objectClass: person
```

  - Attribute der Klasse(n):

```
sn: Schedel
cn: Daniel Schedel
userPassword: {CRYPT}xxxxxxx
```

LDAP: LDIF (2)

- Sammlung von Einträgen voneinander durch Leerzeilen getrennt
- Syntax:
  - Kommentare beginnen mit „**#**“
  - „**:**“ und ein Leerzeichen trennt den Attributnamen vom Wert
- Migration-Tools für Konvertierung in LDIF-Format (**`http://www.padl.com/tools`**)

LDAP: Benutzer / Benutzerauthentifizierung

- Verwendung von LDAP-Einträgen als Quelle für Daten zur Authentifizierung von Benutzern
- Einbinden der Benutzer-Accounts und den dazugehörigen Konfigurationsinformationen in das LDAP-Schema
- Migration der aktuellen Benutzeraccount-Daten aus Standarddateien bzw. aus NIS (PADL-Software: Migration Tools)
- Klassen für Benutzer-Accounts:
  - `posixAccount: /etc/passwd`
  - `shadowAccount: /etc/shadow`

LDAP: Authentifizierung über PAM

Zwei zusätzliche OpenSource-Module für Benutzerauthentifizierung notwendig:

- `nss_ldap`:
  - Modul für NSS
  - Beispielkonfiguration von `/etc/nsswitch.conf`:

```
passwd: files ldap
shadow: files ldap
```

- `pam_ldap`:
  - Modul für PAM
  - Beispielkonfiguration:

```
auth       sufficient  /lib/security/pam_ldap.so
account    sufficient  /lib/security/pam_ldap.so
password   sufficient  /lib/security/pam_ldap.so
```


LDAP: OpenLDAP-Installation

- OpenSource-Implementierung von LDAP
- Basiert auf einer Entwicklung von der University of Michigan
- Quelle: `http://www.openldap.org/`
- Fertige Pakete bei den meisten Linux-Distributionen vorhanden
- OpenLDAP-Installation aus Quellen:
  - Distributionspaket beschaffen und entpacken
  - `./configure`
  - Mit `make depend` und `make` Abhängigkeiten erzeugen und compilieren
  - `make install`

LDAP: OpenLDAP-Konfigurationsdateien

- Konfigurationsdateien für den LDAP-Server:
  - `/etc/openldap/slapd.conf`
  - `/etc/openldap/schema/*`
- Konfigurationsdatei für LDAP-Clients:
  - `/etc/openldap/ldap.conf`
- Konfigurationsdateien für `nss_ldap` und `pam_ldap`
  - `/etc/ldap.conf`
  - `/etc/ldap.secret`
  - `/usr/share/doc/packages/pam_ldap/*`
- SuSE – Boot-Mechanismus
  - `/etc/sysconfig/openldap`

LDAP: `/etc/openldap/slapd.conf` (1)

- `slapd` ist der LDAP-Server (standalone)
- `slapd.conf`: Zentrale Konfigurationsdatei
- Allgemeine Regeln / Syntax:
  - Mit „**#**“ beginnende Zeilen und Leerzeilen werden ignoriert
  - Parameternamen und dazugehörige Werte werden mit Leerzeichen oder Tab getrennt
  - Bei mehrfach vorkommenden Parametern „gewinnt“ der zuletzt gesetzte Wert
  - `slapd.conf` sollte nur für diejenige Kennung lesbar sein, unter der der `slapd`-Dämon gestartet wird (oft `root`) !

LDAP: `/etc/openldap/slapd.conf` (2)

`slapd.conf` besteht aus zwei Abschnitten:

- Globaler Abschnitt
  - Steht am Anfang der Datei und schließt mit Beginn der ersten Datenbasis-Direktive ab
  - Beispiele für globale Einstellungen:
    - Schemas: `nis.schema, core.schema …`
    - Referrals: „Ausweichserver“, falls Anfrage nicht beantwortet wird
    - `pidfile` und `argsfile`: für den laufenden Betrieb
- Datenbasis-Backend
  - Folgt im Anschluss an den globalen Abschnitt
  - Mehrere Datenbasen definierbar

LDAP: Directory-ACLs (1)

- Sehr flexibel und mächtig in der Implementierung
- Einfache Syntax:

```
access to   Was
   by   Wem   Wie
   by   Wem   Wie
```

- Wer bekommt Zugriff?
  - `self` Der DN des bereits verbundenen Benutzers
  - `anonymous` Nicht authentisierte Benutzer
  - `users` Authentisierte Benutzer
- Access Levels („*Wie*“)

  `write, read, search, compare, auth, none`

LDAP: Directory ACLs (2)

- Reihenfolge der ACLs zählt:
  Die erste passende Regel ist ausschlaggebend !
  ➔ spezifischere ACLs **müssen vor** allgemeineren ACLs stehen !
- Beispiel für ACLs:

```
# Attribut  "userPassword"  nur zur
# Authentifizierung, erlaubt Ändern
# des eigenen Passworts.
access to attrs=userPassword
    by self write
    by *    auth
# Erlaubt Leserecht für alle.
access to *
    by *    read
```


LDAP: Directory ACLs (3)

- Beispiel für ACLs:
  Achtung ! Sicherheitsproblem !

```
# Erlaubt Leserecht für alle.
access to *
    by *    read
# Attribut  "userPassword"  nur zur
# Authentifizierung, erlaubt Ändern
# des eigenen Passworts.
access to attrs=userPassword
    by self write
    by *    auth
```

  Welche **fatalen** Auswirkungen hätten diese ACLs ?

LDAP: `/etc/openldap/ldap.conf`

- Konfiguration von LDAP-Clients und Vorgaben für die OpenLDAP-Bibliotheken und -Utilities
- Meist Verwendung des Internetdomain-Namens als Namen (`dc`)
- Beispiele:
  - Standardmäßig abgefragter Teilbaum:
    `BASE dc=lxkursN,dc=admks,dc=lrz-muenchen,dc=de`
  - Standardmäßig abgefragter Server:
    `HOST lxkursN.lrz-muenchen.de`
  - `PORT 389`
  - Maximale Anzahl an abzufragenden Objekten:
    `SIZELIMIT 50`

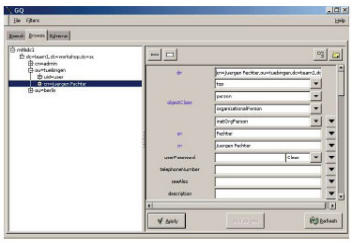
LDAP: `/etc/ldap.conf`

- Festlegung der Konfiguration für PAM- und NSS-Module
- Wichtigste Parameter:
  - IP Adresse des LDAP-Servers:
    `host 192.168.3.1`
  - Wurzel des Directory-Trees:
    `base dc=lxkursN,dc=admks,dc=lrz-muenchen,dc=de`
  - Server-Port:
    `port 389`
  - `ldap_version 3`
  - `timelimit 30`
  - `pam_filter objectclass=account`
  - `pam_password`

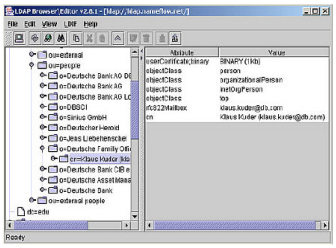
LDAP: Arbeiten mit LDAP (1)

- Verwendung von LDAP-Browsern:

LDAP Client GQ



LDAP Java-Client



DIT nach Anlegen einer neuen Person

- Obige Beispiele von LDAP-Clients nur eine Auswahl

LDAP: Arbeiten mit LDAP (2)

- Nützliche Kommandozeilenbefehle:
  - `ldapsearch` Datensätzen suchen
    - Beispiel:

```
ldapsearch -W \
   -D "cn=Daniel Schedel,dc=lrz-muenchen,dc=de"
```

  - `ldapadd` Datensätzen hinzufügen
  - `ldapmodify` Einträge ändern
  - `ldapdelete` Einträge löschen

LDAP: Arbeiten mit LDAP (3)

- „`slap*`“–Kommandos für Verwaltungsaufgaben:
  - `slapadd` Einfügen von Einträgen aus LDIF-Datei ins LDAP-Verzeichnis
  - `slapcat` Entnahme von Einträgen aus LDAP-Verzeichnis

Achtung! *Verwendung von „`slap*`“-Kommandos bei laufendem Server kann zum Verlust der Datenbasis führen !*

- verschiedene Schnittstellen für C, Perl, … vorhanden

LDAP: Fazit

- Objektorientierte Datenmodellierung
- Offener Standard ermöglicht Herstellerunabhängigkeit
- LDAP bietet hohe Sicherheit durch Zugriffskontrolle und Authentifizierung
- Daten können von verschiedenen Anwendungen verwendet werden
- Viele Einsatzmöglichkeiten von LDAP:
  - Kontaktdateninformationsdienste (z.B. elektronisches Telefonbuch)
  - Authentifizierungsdienst
  - Verzeichnisdienste im Bereich Digital Libraries
  - Content Management Systeme

Network File System (NFS)

Inhalt

- Probleme in verteilten Umgebungen
- Motivation
- Architektur
- Besonderheiten
- Betriebsphasen
- Planung eines NFS-Verbunds
- Bestandteile von NFS
- `/etc/exports`
- Mount-Optionen
- Der Automounter

NFS: Probleme in verteilten Umgebungen

- Administratoren: (fast) gleiche Behandlung aller Rechner
- Benutzer: einheitliche Sicht auf die Daten von jedem Rechner aus
- Probleme bei manueller Lösung:
  - Daten müssen oft und meist umständlich kopiert werden
  - Inkonsistenzen bei den Daten auf den einzelnen Rechnern sind praktisch unvermeidlich.
  - Platzverschwendung, da im schlimmsten Fall auf jedem Rechner eine Kopie der Daten gehalten werden muß (und zwar eine aktuelle !)

  ⇨ Lösung durch ein automatisches System erforderlich

Ziel:zentrale Datenhaltung mit ortstransparentem Zugriff

NFS: Motivation

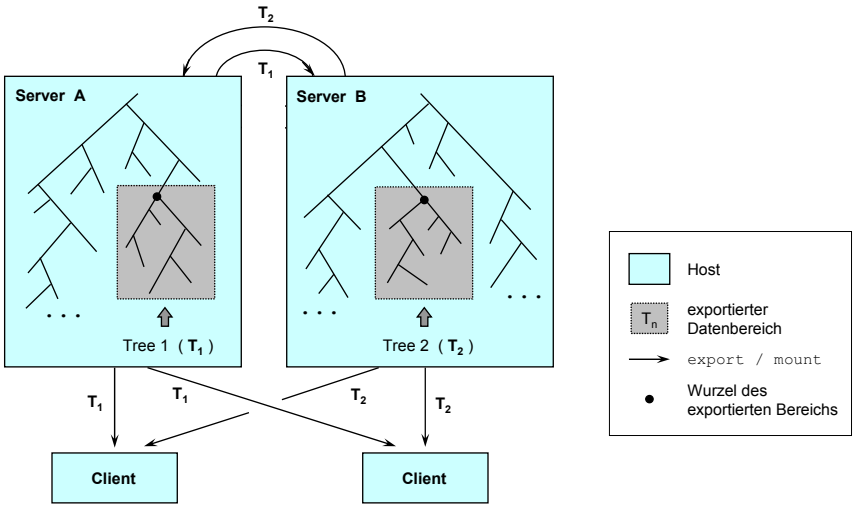
NFS war eines der ersten verteilten Filesysteme und setzte sich aus folgenden Gründen rasch durch:

- ❒ "richtiger" Zeitpunkt, d.h. ausreichender Bedarf
- ❒ "richtige" Plattform (Suns waren damals wie heute weit verbreitet)
- ❒ günstiger Preis
- ❒ einfache und übersichtliche Funktionalität und Strukturierung
  ⇨ Implementierung relativ problemlos
- ❒ Offenlegung der Schnittstellen; geringe Lizenzgebühren

NFS: Architektur

- ❒ Zweck:
  Ortstransparente Dateizugriffe über Netz auf einen *zentralen* Datenbestand unter *Beibehaltung* der UNIX-Filesystem-Semantik
- ❒ Unterschiedliche Funktionen:
  - Server: exportiert Bereiche von lokalen Platten und stellt sie für Zugriffe über Netz zur Verfügung
  - Client: greift auf exportierten Bereich zu
  - Ein Rechner kann bzgl. eines Bereichs nicht gleichzeitig Server und Client sein.
- ❒ Zustandsloses Protokoll:
  Jede Operation muß abgeschlossen werden, bevor es weitergeht.
  ⇨ relativ einfache Fehlerbehandlung

NFS-Konfiguration: Beispiel



81

NFS: Besonderheiten (1)

- NFS und Mount sind unterschiedliche Subsysteme und verwenden deshalb unterschiedliche Protokolle.
- NFS unterstützt keine Special Files.
- Locking durch Zusatzsystem mit Zustandsinformation

  ⇨ `statd` (1M), `lockd` (1M)
- Client puffert kurzfristig (Default):
  - Dateiattribute: min. 3, max. 60 s
  - Directories: min. 30, max. 60 s
  - Daten: Übertragung in 8 KB-Blöcken

NFS: Besonderheiten (2)

Sicherheitsproblem:

Berechtigungen auf der Basis UID / GID, nicht Kennung / Gruppe !

⇨ Abbildung „Kennung → UID“ bzw. „Gruppe → GID“ sollte auf allen Rechnern gleich sein !

Bsp.: Server: User `hugo` mit UID 327
Client: User `clara` mit UID 327

- ❒ Prüfung der Berechtigungen erfolgt auf UID-Basis auf dem *Server*
- ❒ Abbildung „User → UID“ erfolgt auf dem *Client*

⇨ `clara` hat auf dem Client die NFS-Rechte von `hugo` !

NFS: Besonderheiten (3)

Häufigste Fehlersituation:

Fehlermeldung "`Stale NFS File Handle`" auf einem Client, wenn lokal ein Directory (seltener eine Datei) geöffnet ist,

- ❒ das nicht mehr existiert (i.a. weil es gelöscht wurde)
  - ⇨ `umount` oder schlimmstenfalls Reboot
- ❒ dem die Export-Berechtigung entzogen wurde (i.a. versehentlich)
  - ⇨ Berechtigung wieder erteilen oder Reboot

NFS: Betriebsphasen

- Planung des NFS-Verbundes
- Konfiguration der Server und Clients
- Zyklus

Server	Client
Bereich exportieren	
	Bereich mounten
	div. Datenzugriffe
Export aufheben	`umount` des Bereichs

t

- Wartung:
  - Fluktuation (Rechner, Platten)
  - Trouble Shooting

Planung eines NFS-Verbundes

- möglichst wenige Server und exportierte Bereiche
  - ⇨ Fehleranfälligkeit steigt überproportional mit der Gesamt-Komplexität (besonders bei statischen Mounts)
- Für Server ist weniger die CPU- als die Gesamt - I / O - Leistung ausschlaggebend (Platten + Bus + Netz) .
- möglichst kein Benutzebetrieb auf Servern
  - ⇨ Performance
  - ⇨ Security
- auf Servern möglichst wenige Zugangsberechtigte
  - ⇨ Security

NFS-Bestandteile: Dateien

- `exports` (5) bzw. bei Solaris `dfstab` (4)
  - ⇨ Konfiguration für den Export-Befehl ⇨ Server
- `xtab` (5)
  - ⇨ aktuell gültige Export-Berechtigungen ⇨ Server
- `mtab` (4)
  - ⇨ aktuell gültiger Export-Status ("was ist wo gemountet ?") ⇨ Server
- `fstab` (5) bzw. bei Solaris `vfstab` (4)
  - ⇨ Konfiguration für den `mount`-Befehl ⇨ Client
- Konfigurationsdateien für den Automounter ⇨ Client
- `mtab` (5) bzw. bei Solaris / HP `mnttab` (4)
  - ⇨ aktuell gemountete Filesysteme ⇨ Client

NFS-Bestandteile: Dämonen

- `nfsd` (1M)
  - ⇨ bearbeitet die Zugriffe vom Client ⇨ Server
- `mountd` (1M)
  - ⇨ beantwortet `[u]mount`-Wünsche des Clients ⇨ Server
  - ⇨ richtet sich nach der Konfiguration in `xtab` (5)
  - ⇨ mainteniert `rmtab` (4)

NFS-Bestandteile: Kommandos

- `exportfs` (8) bzw. bei Solaris `[un]share[all]` (1M)
  ⇨ (de-)exportiert Dateibereiche; mainteniert `xtab` (5)
- `showmount` (1M)
  ⇨ Anzeige gemounteter und exportierter NFS-Dateibereiche
- `automount` (1M)
  ⇨ automatisches, dynamisches `[u]mount` (bedarfsabhängig)
- `mount` (1M) bzw. bei Solaris zusätzlich `mountall` (1M)
  ⇨ explizites mounten und Anzeige aller gemounteten Bereiche
- `umount` (1M) bzw. bei Solaris zusätzlich `umountall` (1M)
  ⇨ explizites unmounten
- `nfsstat` (1M)
  ⇨ Statistiken über die NFS-Nutzung

`/etc/exports` (1)

Syntax von `/etc/exports` (5)

- *Directory* [- *Option* [, *Option*] ...]
- Leerzeilen oder Strings nach "`#`" werden als Kommentare interpretiert

Syntax von `dfstab` (4)

- `share -F nfs` [- *Option* [, *Option*] ...] *Directory*

/etc/exports (2)

Gängige Optionen:

- `ro`
  - ⇨ nur lesender Zugriff (Default: auch schreibend)
- `rw=` *Host* [: *Host* ...]
  - ⇨ nur aufgeführte Rechner dürfen schreiben (Default: alle)
- `root=` *Host* [: *Host* ...]
  - ⇨ nur aufgeführte Rechner dürfen als Root zugreifen; bei allen anderen werden Root-Zugriffe wie „nobody“-Zugriffe behandelt
- `access=` *Client* [: *Client* ...]
  - ⇨ Nur aufgeführte Rechner dürfen zugreifen (Default: alle). *Client* kann ein Rechner oder eine Netgroup sein.

/etc/exports: Beispiel

```
#############################################################
## Beispiel für /etc/exports(5)
## Home Directories ##########################################
/mnt/home1h           -access=suns
/mnt/home2h/users     -access=suns
## Verweis-System fuer Home Directories #####################
/mnt/home2h/home      -ro,access=suns
/mnt/home5h/home      -ro,access=suns
## Lizenz-SW ################################################
/mnt/sw/OS.413        -ro,access=suns:admks
/mnt/sw/imsl-2.0      -ro,access=suns
## OpenSource-SW ############################################
/mnt/sw/PD            -access=suns:admks,rw=afs1,root=afs1
/mnt/sw/install       -access=srvsuns:usrsuns,root=afs1
/mnt/sw/src           -access=srchosts,rw=afs1:sun1,root=afs1
## Sonstiges ################################################
/mnt/home7h/intern    -access=suns:admks,rw=afs1,root=afs1
```

NFS: Mount-Optionen (1)

Gängige Mount-Optionen, die auch für NFS anwendbar sind

(Default jeweils **`fett`**)

- **`rw`** / `ro`
  - ⇨ read / write bzw. read-only
- **`suid`** / `nosuid`
  - ⇨ Set-UID und Set-GID aktivieren / deaktivieren

NFS: Mount-Optionen (2)

NFS-spezifische Mount-Optionen (1)

(Default jeweils **`fett`**)

- **`fg`** / `bg`
  - ⇨ Wiederholungsversuche im Vordergrund / Hintergrund
- `noquota`
  - ⇨ keine Kontingentüberprüfung auf dem Client
    (unabhängig vom Verhalten des Servers)

NFS: Mount-Optionen (3)

NFS-spezifische Mount-Optionen (2)

(Default jeweils **`fett`**)

- **`hard`** / `soft`
  - ⇨ falls Server nicht antwortet, "endlos" weiter versuchen bzw. Fehlermeldung

  NFS-Bereich entspricht im Fehlerfall
  - bei `hard` einer *sehr langsamen* Platte
  - bei `soft` einer *defekten* Platte

- `intr`
  - ⇨ Bei Hard-Mounts Abbruch von Prozessen möglich, die wegen Server- oder Netzproblemen hängengeblieben sind.

NFS: Der Automounter (1)

Zweck

- erfüllt im Prinzip dieselbe Funktionalität wie `mount` (1M) bzw. `umount` (1M) ⇨ `automount` (1M)
- mountet Bereiche dynamisch, d.h. erst bei Bedarf
- unmountet, wenn eine bestimmte zeitlang nicht zugegriffen wurde

Vorteile

- System wird Fehler-robuster; bei nicht gemounteten Bereichen kann kein „`Stale NFS File Handle`“ auftreten.
- mit Hilfe von NIS zentral und konsistent konfigurierbar

NFS: Der Automounter (2)

Konfiguration

- Konfiguration ist mit Dateien und/oder LDAP- bzw. NIS-Maps möglich
- Es gibt 3 Arten von Konfigurationsdateien (Automounter-Maps):
  - Master-Map ⇨ Verzeichnis aller Maps mit Mountpoints
  - Direct-Map ⇨ Mountpoints durch absolute Pfade spezifiziert
  - Indirect-Map ⇨ Mountpoints sind Unterverzeichnisse in einem ebenfalls vom Automounter verwalteten Map-spezifischen Verzeichnis
- Achtung: Automounter - " Map" ≠ NIS - " Map"

Automounter-Maps: Beispiel für `auto.master`

```
#######################################################
##  Automounter-Map  "auto.master":  Map of maps
#######################################################
# mount     map-name        default options
# point

/-          auto.direct     -ro,soft,noquota,intr
/homesys    auto.home       -ro,soft,noquota,intr
/nfs        auto.nfs        -ro,soft,noquota,intr
```

Automounter-Maps: Beispiel für `auto.direct`

```
#########################################################
##  Automounter-Map  "auto.direct":
##       Direct map for "/home"
#########################################################

# mount   options                location
# point

/cluster  -ro,soft,noquota,intr  sunserver:/mnt/sw/cluster
/home     -ro,soft,noquota,intr  sunserver:/mnt/home0h/home \
                                 sunserver:/mnt/home1h/home
/sw       -ro,soft,noquota,intr  sunserver:/mnt/sw2/sw-links
```


Automounter-Maps: Beispiel für `auto.home`

```
#########################################################
##  Automounter-Map  "auto.home":
##       Indirect map for home directories in  "/homesys"
#########################################################

# mount   options     location
# point

home0     -rw,intr    sunserver:/mnt/home0h/localusers
home1     -rw,intr    sunserver:/mnt/home1h/users
home2     -rw,intr    sunserver:/mnt/home0h/users
home5     -rw,intr    sunserver:/mnt/home3g
home50    -rw,intr    afs1:/mnt/home1h
largetmp  -rw,intr    sun5:/mnt/data/&
```

Automounter-Maps: Beispiel für `auto.nfs`

```
#############################################################
##  Automounter-Map  "auto.nfs":
##       Indirect map for software packages
#############################################################
# mountpoint    options                 location
# Area for generation/installation ##########################
build.gnu       -rw,soft,noquota,intr     sun3:/usr/local/build/gnu
cluster.build   -rw,soft,noquota,intr     sunserver:/mnt/sw3/&
cluster.src     -rw,soft,noquota,intr     sunserver:/mnt/sw/&
# Kurse #####################################################
kurs.adm        -ro,soft,noquota,intr     sunserver:/mnt/home1h/&
kurs.os         -ro,soft,noquota,intr     sunserver:/mnt/home1h/&
# Anonymous FTP archives ####################################
pub             -rw,soft,noquota,intr     sunserver:/mnt/home7h/ftp/&
public          -ro,soft,noquota,intr     \
        hpsystem2:/usr/proj/archive/isar/.mntpts/tum.info-pub1
public2         -ro,soft,noquota,intr     \
        hpsystem2:/usr/proj/archive/isar/.mntpts/tum.info-pub2
# License-SW ################################################
OS.413          -ro,soft,noquota,intr     sunserver:/mnt/sw/&
imsl-2.0        -ro,soft,noquota,intr     sunserver:/mnt/sw/&
# OpenSource-SW #############################################
PD              -rw,soft,noquota,intr     sunserver:/mnt/sw/&
PD2             -rw,soft,noquota,intr     sunserver:/mnt/sw2/&
```