Benutzer im Netzwerk

Login und Accountverwaltung in UNIX-Netzwerken

Sebastian Kolbe (sko@pool.math.tu-berlin.de)

Übersicht

- login / xdm / PAM / ssh
- Verteilte Dienste
  - NIS
    - NIS (YP) -Server/Client
  - RPC
  - NFS
    - Automount
- Shell

Login

- X-Oberfläche: XDM / XDMCP
- Remote:
  - sshd
  - telnetd / rlogind / rshd ...
- Authentication: login / PAM
- Linux-PAM: *Pluggable Authentication Modules*
  - Erweiterung für *login*

'*login*'

- UNIX-Dienst für interaktive Authentifizierung mit Username und Passwort
- Startet Session für Benutzer und stellt Systemvariablen ($HOME, $USER, ...) bereit
- Mailcheck (unterdrückt durch $HOME/.hushlogin)
- Dateien:
  /etc/login.defs,
  /etc/passwd, /etc/shadow | /etc/security/passwd
  /etc/motd, /etc/issue, /etc/nologin,
  (/etc/nsswitch.conf, /etc/pam*)

'*login*': Dateien (*Linux*)

- /etc/login.defs
  - Enthält Einstellungen zu "login-suite":
    - Max. Versuche beim Einloggen, Passwortändern...
    - Environment
    - Log-Dateien
    - Meldungen zum Einloggen: /etc/issue /etc/motd
    - Verhalten von 'su', passwd, ...
  - Viele Einstellungen werden von PAM überschrieben oder ersetzt!

'*login*': Dateien (*AIX*)

- /etc/security/user
  - Enthält globale Einstellungen und Definitionen pro User
    - Auf *NIS-Master Rechner* jeder Benutzer eingetragen!
    - Authentifikation (Art und Gültigkeit)
    - Einloggen: max. Versuche und Alterung des Passworts
    - Verhalten von: rlogin und telnet, su, passwd
    - Möglichkeit admin-Rechte einzuräumen (groups, roles)
- /etc/security/limits
  - Limitieren der Systemresourcen (Prozesse, Speicher, ...)
- /etc/security/login.cfg
  - Verfügbare Shells und Loginmeldungen

PAM (*Pluggable Authentication Modules*)

- Modulare Erweiterung von 'login' unter *Linux*, *Solaris*, *IRIX*, *HPUX*, *FreeBSD*, ...
- Teilt Authentifizierung in mehrere Teile (Module) auf und stellt Konfigurationen (/etc/pam.d/*) für Anwendungen bereit
- Module: /lib/security/pam_*.so
  - Authentifizierung (Passwörter, LDAP, Kerberos, NIS+, Challenge-Response, Chipkarten, usw.)
  - Environment (liest /etc/environment)
  - Limits (liest /etc/security/limits.conf)
  - ...

PAM (*Pluggable Authentication Modules*)

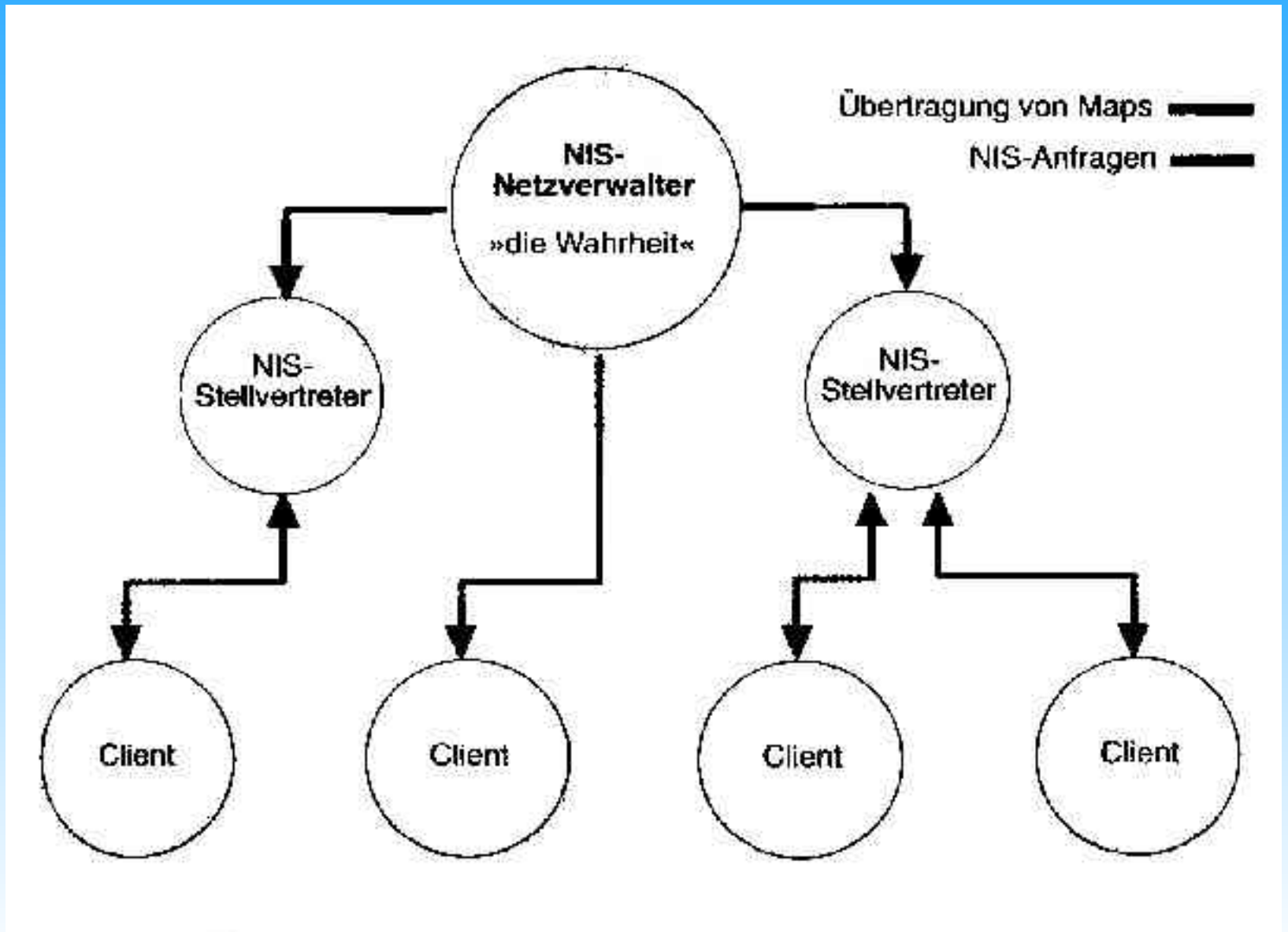
- PAM Konfiguration /etc/pam.d/*
  - Eine Datei pro Service, 'other' als default-Routine
    - Servicetyp: account, auth, session, password
    - 'Bedeutung': required, optional, requisite, sufficient
    - Modulname + Parameter

```
#%PAM-1.0
auth            requisite       pam_securetty.so
auth            sufficient      pam_rhosts_auth.so
auth            required        pam_unix.so nullok
auth            required        pam_nologin.so
account         required        pam_unix.so
password        required        pam_unix.so nullok use_authtok \
                                            obscure min=4 max=8
session         required        pam_unix.so
```

NIS

- *Network Information System* (aka *YP*)
- Informations- und Accountingdatenbank transparent zugänglich über in normale Systemaufrufe integrierte RPC-Funktionen
- Hauptsächlich zur Authentifizierung und Verteilung von Benutzerdaten benutzt
- System aus Masterserver und Replikationsservern (Slave-Server)
- Nur der Masterserver hat alle Informationen als reale Daten vorliegen (z.B. /etc/passwd)

NIS: Aufbau


Übertragung von Maps
NIS-Anfragen
NIS-Netzverwalter
»die Wahrheit«
NIS-Stellvertreter
NIS-Stellvertreter
Client
Client
Client
Client

Verwaltung von MAP-Dateien

- NIS-Server verwalten MAP-Dateien
  - passwd.byname, passwd.byuid, mail.aliases, hosts.byname, ...
  - Mapdateien werden unter den Servern regelmässig synchronisiert
- MAP-Dateien und Server sind NIS-Domänen zugeordnet: z.B. *lehrepool (Name ist frei wählbar)*
- Clients binden sich immer an Domäne und suchen sich selbst den richtigen Server dazu (via broadcast)
  - Siehe 'domainname'
  - Dateien: /etc/defaultdomain (linux) | /etc/rc.nfs (AIX)
  - Programme können Maps von anderen Domänen anfordern

NIS-Komponenten

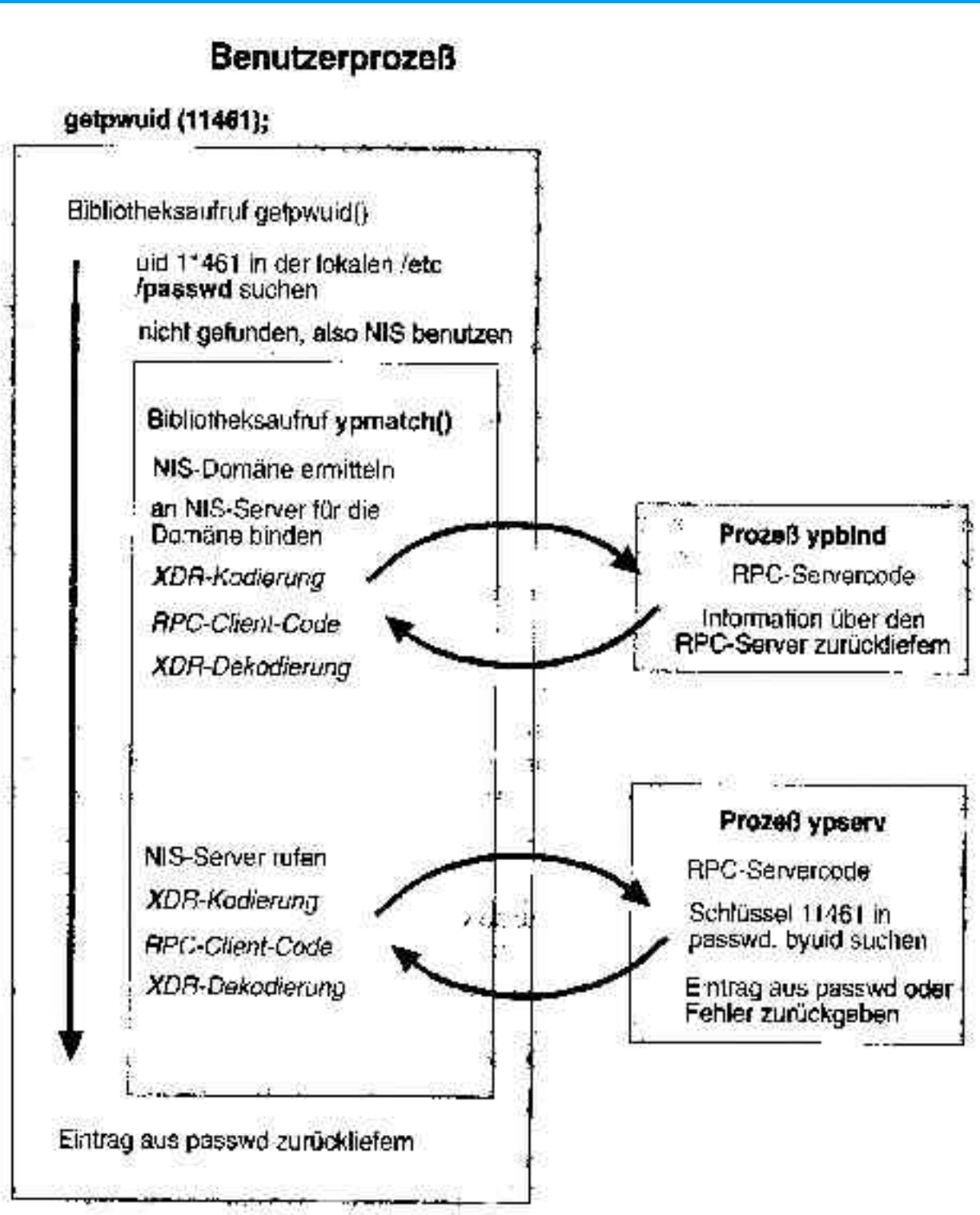
- Client
  - *libc* mit YP/NIS-Support
    (siehe auch /etc/nsswitch.conf)
  - ypbind stellt Verbindung von libc-Prozedur zum Server her
  - 'domainname', 'ypdomainname'
  - 'ypwhich', 'ypset', 'yppoll', 'yppush', 'ypxfr'
  - 'ypcat', 'ypmatch'
  - 'yppasswd', 'ypchfn', 'yp...' [usertools]

NIS-Komponenten

- Server
  - 'ypserv' als RPC-Dienst
  - Supportet alle Domänen in /var/yp/<domainname>
  - Unter /var/yp/<domainname> stehen alle Maps in einem Datenbankformat (libdbm) und mit Sortierung (.byname, byuid, ...)
  - Kurzbezeichnungen: passwd.byname == passwd (siehe /var/yp/nicknames)
  - /var/yp/Makefile baut aus realen Dateien dbm-Dateien
  - ypxfr kann ganze Maps vom Server holen und übertragen (z.B. zur Synchronisation der Slaveserver)

RPC (1)

- Beispiel für NIS-Anfrage:



RPC (2)

- *Remote Procedure Call* (auch SunRPC)
- Einbindung wie mit lokalen Daten
  - Lokales Programm weiß nichts von RPC-Aufruf
  - Programm wartet bis RPC-Code ein Ergebnis liefert (oder allgemeine Fehlermeldung)
- Teile des RPC-Subsystems:
  - Liste mit Programmnummern: /etc/rpc + NIS-Map
  - Portmapper
    - Registriert RPC-Dienste und liefert auf Anfrage zugehörige Portnummern der Dienste
    - Liegt immer auf Port 111 (sunRPC)
  - Dienstprogramme (z.B. 'rpcinfo')

RPC (3)

- Jeder RPC-Dienst wird festgelegt durch:
  - Programmnummer (siehe /etc/rpc)
  - Versionsnummer
  - Prozedurnummer
  - Protokoll (UDP oder TCP)
- Jeder Dienst enthält außerdem eine Null-Prozedur, die z.B. zu Testzwecken (siehe rpcinfo) benutzt wird
- Jeder RPC-Server bearbeitet eine Anfrage vollständig, bevor die nächste bearbeitet wird
- Datenaustausch mit plattformunab. XDR-Codierung
- Bei hohem Bedarf mehrere RPC-Server pro Dienst möglich (z.B. NFS)

NFS (1)

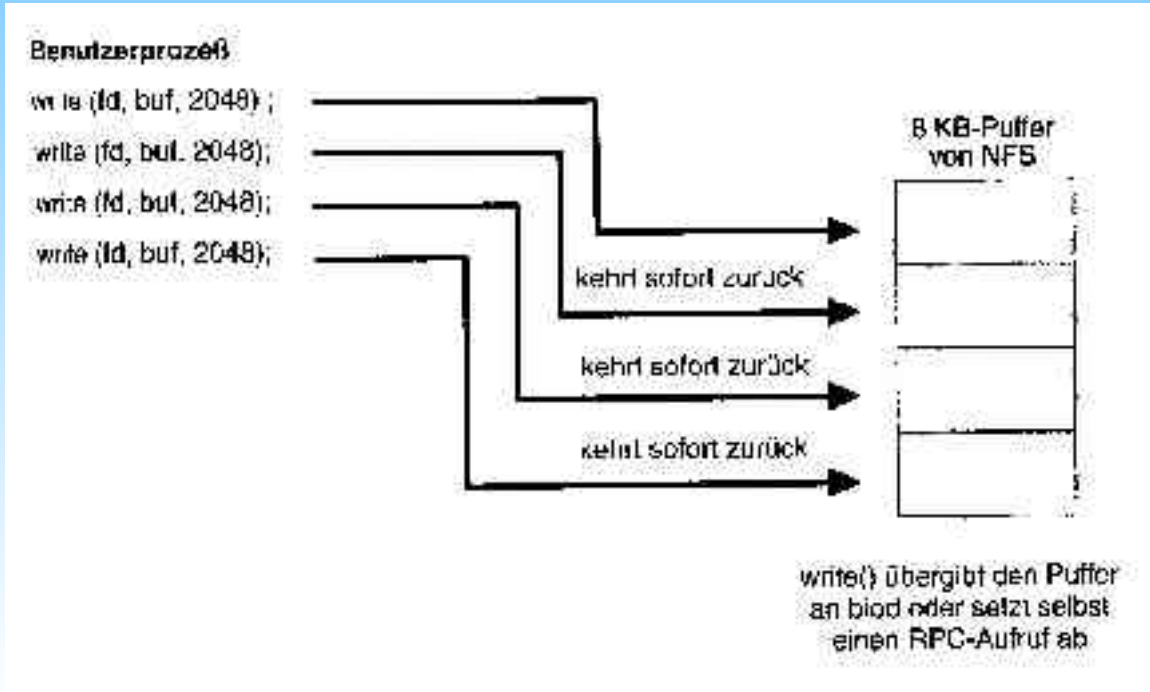
- *Network File System*
- Verteiltes Dateisystem auf RPC-Basis
  - Transparenz für Benutzer
  - Einfache Administration von Plattenplatz
  - Unterstützung durch prak. alle UNIX-Hersteller
- Zustandsfreies Protokoll
  - Jede NFS-RPC Anfrage beschreibt Operation vollständig
  - NFS-RPC Anfragen sind *idempotent*, d.h. es können mehrere gleichartige Anfragen nacheinander kommen. 'duplicate request cache', der gleiche Anfragen verwirft
  - Client wiederholt Anfragen solange, bis Operation abgeschlossen ist (siehe auch hard/soft-mount)

NFS (2)

- 'nfsd'
  - bearbeitet RPC-Anfragen
  - der größte Code-Teil ist im Betriebsystemkern enthalten
  - Meist mehrere Instanzen vorhanden, die Anfragen nach einem Warteschlangenprinzip beantworten
  - 'lookup' Operationen auf Dateien oder Verzeichnisse liefern 'file handles':
    - Meist aus Inode, Gerätenummer und Zeitstempeln erzeugte ID
    - Dient dem Client zum Zugriff auf Dateisystemobjekt
    - Nur nfsd kann dem 'file handle' eine reales Objekt zuordnen
    - "stale NFS file handle": Inode (auf dem Server) ist ungültig; Der Client versucht aber noch darauf zuzugreifen. (reboot!)

NFS (3)

- Client
  - Dateizugriff mit vom Server vergebenen 'file handle'
  - 'biod' optimiert Serverzugriff durch Caching und Zusammenfassen mehrerer Operationen
  - Gepuffertes lesen und schreiben ermöglicht asynchrones I/O



NFS (4)

- Benötigte Dienste:
  - rpc.mountd [server] Mount-Operationen
  - rpc.nfsd [server] führt RPC-Anfragen aus
  - biod [client] stellt blockorientierte I/O zur Verfügung
  - rpc.lockd, rpc.statd Dateisperren (locking) über NFS
  - /etc/exports [server] Exportliste
  - /var/lib/nfs/{xtab,rmtab,etab,sm} [server] Statusdateien
  - exportfs, mount, showmount, nfsstat
  - rpc.quotad [server] Quotainformationen abfragen
  - automountd [client]

NFS (5)

- Benutzung
  - Exportieren von Dateisystemen
    - Jedes Verzeichnis kann nur einmal exportiert werden (!)
    - Nur echte Teilmengen eines Dateisystems sind möglich
    - Nur lokale Verzeichnisse können exportiert werden
    - root-Zugriff (uid == 0) wird standardmäßig auf nobody (bzw. anonuid) umgesetzt
  - mount-Optionen
    - 'hard': (default) Alle RPC-Operationen werden wiederholt bis der Server eine Bestätigung sendet. Prozesse bleiben "hängen"
    - 'soft': Wenn eine Zeitüberschreitung im RPC-Teil stattfindet, wird eine Fehlermeldung zurückgegeben
    - 'intr': RPC-Operationen sind unterbrechbar mit Signalen
    - 'retrans', 'timeo': Wiederholungen und Timeout

Automounter

- Ohne Automounter müssten alle Benutzerverzeichnisse auf allen Rechnern ständig gemountet sein!
- Automounter mountet automatisch Dateisysteme (meist NFS), wenn ein Prozess dieses Verzeichnis anfordert
  - Erfordert ein 'Magic'-Verzeichnis (z.B. /homes)
  - Wenn ein Verzeichnis angefordert wird (z.B. mit chdir()) wird es automatisch erzeugt
  - Der Prozess wird gestoppt, bis mount-Befehl fertig ist
  - Automounter prüft regelmäßig, ob Verzeichnis noch in Benutzung ist, sonst: umount
  - Kernelunterstützung (autofs)

Automounter

- Konfiguration
  - Zuordnung von 'Magic-Directory' und Mountpoints
  - weitere Mountoptionen
  - Datei: /etc/auto.master
  - NIS-Map auto.master
    - Incl.: automounts.home und automounts.net

Shell

- 'login', 'xterm', 'sshd', ...
  startet Kommandointerpreter (Shell)
  - Shell wird aus /etc/passwd (bzw. NIS-Map) gelesen
  - Shell wird im *login-Modus* aufgerufen
- /usr/local/bin/usershell
  - Standardshell für Pool-Benutzer
  - Prüft, ob Rechner für Benutzer freigegeben ist
    - Kriterium:Rechnername ($HOSTNAME)
      und Displayname ($DISPLAY)
  - Startet dann Shell
    Default: /usr/local/bin/tcsh

Quellen

- Hal Stern, "*Verwaltung von UNIX-Netzwerken mit NFS und NIS*", O'Reilly, 1998
- Tobias Klein: "*Linux-Sicherheit*", dpunkt.verlag, Heidelberg, 2001
- Marc Heuse, "Flexible Authentifizierung mit PAM", c't, 2000/26, Heise Verlag, Hannover
- *Man*-Pages zu 'mount', 'automountd', 'login' (und weiteren)

Ende