SOLARIS

Ernst Pisch

17. Juli 2002

Text und Informationen in diesem Buch wurden mit größter Sorgfalt erarbeitet. Dennoch können Fehler nicht vollständig ausgeschlossen werden. Der Autor übernimmt keine juristische Verantwortung oder irgendeine Haftung für eventuell verbliebene Fehler und deren Folgen.

Inhaltsverzeichnis

Vorwort

Diese Zusammenstellung von diversen Themen rund um SOLARIS bzw. UNIX entstand ursprünglich, um meinen Arbeitskollegen und mir selbst eine Arbeitsunterlage zu schaffen. Nach und nach habe ich auch Dinge hinzugefügt, welche für einen UNIX-Vertrauten selbstverständlich sind, aber einem UNIX-Neuling beim Einstieg helfen sollen. Bei einigen Kapiteln existiert derzeit lediglich die Überschrift. Aus zeitlichen Gründen komme ich nur sehr langsam mit dem Schreiben weiter. Aus diesem Grunde habe ich mich dazu entschlossen das Werk - obwohl unvollständig - so zu veröffentlichen. Geplant habe ich vor allem Informationen zum Thema Netzwerk und Netzwerkadministration sowie mehr Tipps im Anhang.
Falls jemand Fehler entdeckt, wäre ich sehr dankbar, diese zu erfahren.

1 Grundbegriffe

1.1 Übersicht - Hardware

Bus-Systeme

SBUS bis 95 MB/s

PCI-Bus bis 132 MB/s; bei SUN: ’extended PCI-Bus’ mit doppelter Übertragungsrate

Workstations

1. Generation SBUS, 32-Bit HW + SW
IPC, SLC, SS2, IPX SPARC1

2. Generation SBUS, 32-Bit HW + SW
LX, SS4, SS5, SS10, SS20

3. Generation PCI-Bus, 64-Bit HW + SW
U2 (SBUS)
U5, U10 (IDE-Platten, Ultra SPARC-IIi)
U30, U60 (bis 2xCPU Ultra SPARC-II)

Server

1. Generation 3xx, 6xx

2. Generation 1000 (bis 4 x Super-SPARC), 2000 (bis 16 x Super-SPARC)

3. Generation S10, S60 (beide wie Workstation U10 bzw. U60, aber ohne Grafikkarte)
150, 250, 450 (kleine Server, Ultra SPARC-II)
3500, 4500, 5500, 6500 (große Server, Ultra SPARC-II, Hot-Plug Unterstützung)
10000, E10k (High End Server)

1.2 Fachausdrücke

Virtual Memory Operating System Virtueller Speicher ermöglicht Anwendungen mehr Speicher zu verwenden als physikalisch vorhanden ist. Dies wird durch temporäres Auslagern von Speicherbereichen auf Festplattenbereiche ermöglicht. Diese Festplattenbereiche werden **swap space** genannt.

Daemons Dämonen sind Programme, welche im Hintergrund laufen und Systemfunktionen wie z.Bsp. das Drucken steuern.

Host Ein Computersystem in einem Netzwerk

Host name Eindeutiger Name für ein System; jedes System im Netzwerk muss einen Hostnamen haben

IP address eine Nummer, welche im Netzwerk zur Identifikation der Maschinen dient

Client ein Host oder Prozess, welcher die Dienste eines Servers im Netzwerk nützt

Server ein Host oder Prozess, welcher Resourcen oder Dienste zur Verfügung stellt
z.Bsp.: file server, NIS server, print server, mail server, JumpStart server

Network eine Gruppe von Computern, welche so verbunden sind, dass sie kommunizieren können (z.Bsp. per Ethernet)

Multitasking ermöglicht den Ablauf von mehr als einem Prozess oder Programm zur selben Zeit

Multiuser ermöglicht es mehr als einem Anwender dieselben Systemressourcen zu benützen

Distributed processing ermöglicht die Benützung von Systemressourcen über ein Netzwerk hinweg

File system Hierarchie von Verzeichnissen, Unterverzeichnissen und Dateien

Shell Interface zwischen Anwender und Kernel

Kernel Kern des Betriebssystems - dessen Aufgaben sind: Verwaltung von Geräten, Speicher, swap space, Prozessen und Dämonen; Kommunikation zwischen Systemprogrammen und Hardware; führt Kommandos aus und reiht sie;

Diskless Client Workstation ohne Platte. Betriebssystem und Anwenderprogramme werden vom Netz geladen.

AutoClient Gleich wie 'Diskless Client' mit dem Unterschied, dass Betriebssystemteile (/root, /usr und swap) auf einer lokal vorhandenen Platte ge'cached' werden.

JavaStation Station, welche über Netz gebootet wird und **nur** wie ein Browser mit Java-Funktionalität verwendet werden kann. Sinn dieser Konstellation ist der geringe Administrationsaufwand.

Sun Ray™ Die Station beinhaltet nur die physikalischen Komponenten wie Tastatur, Maus, Bildschirm, Audio Ein- und Ausgabe. Applikationen laufen auf einem Server, nur die Darstellung erfolgt auf der Station.

Package Ein Software-Package ist eine Zusammenfassung von Dateien, Verzeichnissen, Installationsskripten und Beschreibungsdateien.

Runlevel Betriebsstatus des Systems. Z.Bsp. Multiuser-, Singleuser-Level; siehe Kapitel 3.2 ’Init Prozess’, Seite 10.

1.3 Übersicht der Systemtypen

System Typ	Lokale Dateisysteme	Lokale Swap	Netzwerk Dateisysteme	Netzwerk Belastung	Performance
Server	/, /usr, /home, /opt, /export/home, /export/root	ja	nein	hoch	hoch
Standalone System	/, /usr, /export/home	ja	nein	niedrig	hoch
Diskless Client	nein	nein	/, swap, /usr, /home	hoch	niedrig
Java Station	nein	nein	/home	niedrig	hoch
Auto Client System	cached /, cached /usr	ja	/var	niedrig	hoch

Liste von Servertypen:

Application Server Ermöglicht den Zugriff über Netz auf Programme wie z.Bsp. FrameMaker, Photoshop, Lotus Notes, ...

Boot Server Versorgt Diskless-Stations, Auto-Clients, Java-Stations und Install-Clients mit Bootparametern und ermöglicht das Booten über Netz.

Installation Server Stellt per NFS Installations-Software für Install-Clients zur Verfügung.

Database Server Bietet Datenbankzugriff über Netz.

Mail Server Stellt Mailtransfer von und zum lokalen Netzwerk zur Verfügung.

Home Directory Server Exportiert lokal lagernde Home-Verzeichnisse (üblicherweise */export/home*) für den Zugriff von Benutzern

License Server Beinhaltet einen Lizenzdämon, der die Verwendung der Lizenzen für System bzw. Applikationen verwaltet.

Print Server Stellt ein Drucksystem zur Verfügung. Drucker können lokal angeschlossen sein, oder im Netzwerk hängen.

Name Services Server Stellt einen Namensdienst wie z.Bsp. DNS, NIS, NIS+ oder FNS zur Verfügung.

2 Solaris Installation

Solaris 7 inkludiert:

- SunOS 5.7 operating system
- ONC+ (Open Network Computing) - inkludiert NFS, NIS+, XFN, RPC, ...
- CDE (Common Desktop Environment) - inkludiert OpenWindows

2.1 Software Cluster

Da nicht alle Anwender dieselben Anforderungen stellen, sind die Softwarepakete für die unterschiedlichen Bedürfnisse in sogenannte 'Software Cluster' zusammengefasst (siehe auch Kapitel 4.1, Seite 22).

SUNWreq Core; zwingend erforderliche Minimalkonfiguration;

SUNWuser EndUser; für Durchschnittsanwender; benötigt 438 MB (32Bit) bzw. 532 MB (64Bit) Festplattenplatz;

SUNWprog Developer; für Softwareentwickler; benötigt 716 MB (32Bit) bzw. 837 MB (64Bit);

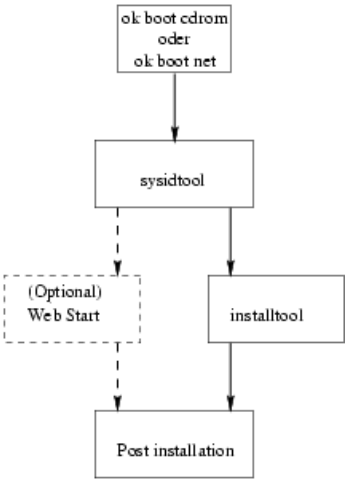
SUNWall Entire Distribution; komplette Solaris Installation; benötigt 787 MB (32Bit) bzw. 895 MB (64Bit);

SUNWxall Entire Distribution plus OEM support; komplette Solaris Installation mit zusätzlichen Paketen für Fremdprodukte; benötigt 801 MB (32Bit) bzw. 909 MB (64Bit);

2.2 HW Voraussetzungen

- SPARC- oder Intel-System
- Platte $\geq$1GB
- Hauptspeicher $\geq$64MB
- CD-ROM Laufwerk an SCSI-Bus oder Netzzugang zu einem 'JumpStart-Server'
- Monitor und Tastatur

2.3 Installationsablauf



Zunächst muss die Solaris Installations-CD eingelegt werden und im Boot-PROM mit dem Kommando:
`ok` **`boot cdrom`**
gestartet werden.

Im ersten Teil der Installation werden im Dialog einige Werte für die weitere Installation des Systems abgefragt:

- Language and Locale
- Host Name
- Network Connectivity (Anschluss an Netzwerk – ja/nein)
- Primary Network Interface (Auswahl der Netzwerkkarte – z.Bsp. hme0)
- IP Address
- Name Service (Auswahl des Name Service – NIS+, NIS, Other (damit ist DNS gemeint), None); Der Name Service kann auch später noch konfiguriert werden;
- Subnet (Teil eines Subnetzes – ja/nein)
- Netmask (Subnetmask, wenn Teil eines Subnetzes)
- Time Zone Information (die Eingabe kann über unterschiedliche Arten erfolgen: Geographische Region, Differenz zu GMT oder über eine Timezone-Datei)
- Local Date and Time
- Upgrade/Initial (falls sich schon ein Solaris-Betriebssystem am Server befindet, wird das erkannt und abgefragt, ob eine Neuinstallation oder ein Update gewünscht wird; Bei einem Update können anschließend auch zusätzliche Softwarepakete zur Installation ausgewählt werden)

- Select Language (Auswahl zusätzlicher Sprachvarianten – Englisch ist Standard)
- Allocate Client Services (es können an dieser Stelle die nötigen Ressourcen für Diskless- und Auto-Clients konfiguriert werden)
- Select Software (Auswahl von 64Bit Support und des Software-Clusters – Core System Support, End User System Support, Developer System Support, Entire Distribution, Entire Distribution plus OEM support)
- Select Disks (Auswahl der Systemplatte)
- File System and Disk Layout (Partitionierung der Platte)
- Mount Remote File Systems? (NFS-Dateisysteme zu mounten – ja/nein)
- Profile (nun wird noch zur Kontrolle eine Zusammenstellung aller zuvor eingegebenen Werte angezeigt)
- Reboot (an dieser Stelle kann gewählt werden, ob nach der Installation ein automatischer Reboot erfolgen soll oder nicht)
- Root Password (Eingabe des Root Passwortes)
- Power Management (Abfrage, ob automatischer Power Management Shutdown gewünscht ist oder nicht); Bei Servern sollte diese Frage unbedingt mit NEIN '*n*' beantwortet werden.

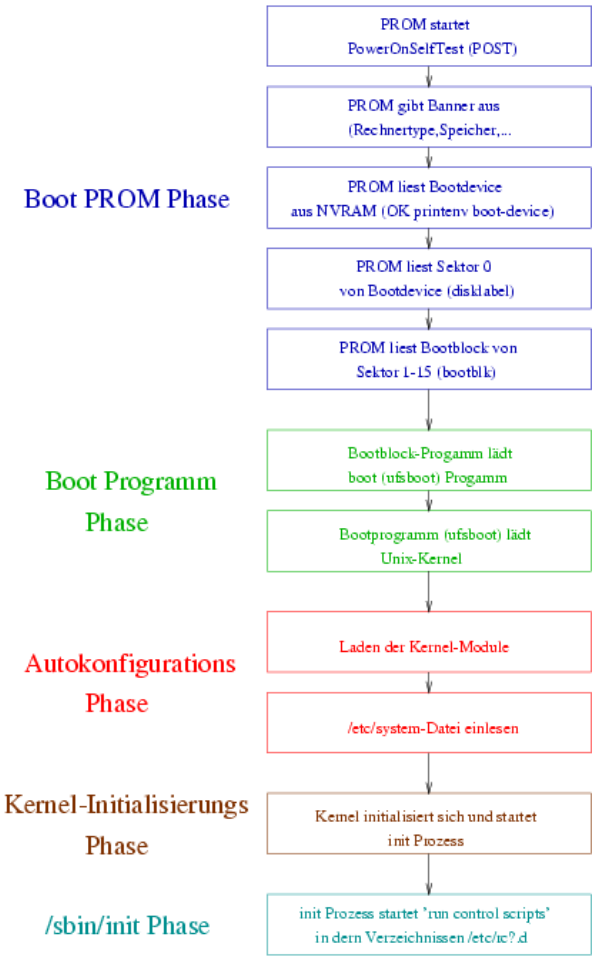
Die eigentliche Installation ist nun fertig. Jetzt sollten noch individuelle Einstellungen durchgeführt werden wie z.Bsp. Auswahl der Default-Shell (Bourneshell, Kornshell, C-Shell), editieren von /etc/hosts, Anpassen von .profile, /etc/default/login etc. sowie das Einrichten von Benutzerkennungen (siehe Kapitel 7, Seite 55).

Im Verzeichnis */var/sadm* befinden sich einige Unterverzeichnisse, welche die Installation betreffen:

- */var/sadm/system/logs/install_log* — In diese Datei wird das Protokoll der Installation geschrieben.
- */var/sadm/pkg* — Das Verzeichnis enthält Informationen zu allen installierten Softwarepaketen
- */var/sadm/softinfo* — In diesem Verzeichnis steht die Datei *INST_RELEASE*, welche die Version von Solaris enthält. Dies ist ein symbolischer Link auf die Datei */var/sadm/system/admin-/INST_RELEASE*.
- */var/sadm/install_data/install_log* — Das ist ein Link auf */var/sadm/system/logs/install_log*.
- */var/sadm/system* enthält einige wichtige Unterverzeichnisse (*admin*, *data* und *logs*) und Systeminformationen. Z.Bsp. enthält die Datei */var/sadm/system/admin/CLUSTER* den Softwarecluster, der während der Installation angegeben wurde. Die Datei */var/sadm/system/admin-/INST_RELEASE* enthält die installierte Solaris-Version.
- */var/sadm/install/contents* — Diese Datei enthält Informationen zu allen Dateien und Verzeichnissen, welche mit einem Softwarepaket installiert wurden.

3 Systemstart

3.1 Der Boot Vorgang



3.1.1 Boot PROM Phase

1. Das Bootprogramm auf dem PROM startet den Selbsttest (siehe Kapitel 3.5, Seite 21).
2. Es wird der sogenannte 'System Identification Banner' angezeigt. Dieser enthält einige wichtige Informationen wie Gerätetyp, Tastaturtyp, PROM Version, Speichergröße, PROM Seriennummer, Ethernet-Adresse und Host-ID.
3. Nun wird das Disk-Label aus dem Sektor 0 des Bootdevice gelesen. Welches das Bootdevice ist, steht in der Variable 'boot-device' im NVRAM (siehe Kapitel 3.4.4, Seite 20).
4. Jetzt wird der 'boot block' aus den Sektoren 1-15 des Bootdevice gelesen. Das ist ein Programm, welches in der Lage ist eine Datei aus einem ufs-Dateisystem zu laden. Es wird im Zuge der Installation mit Hilfe des Programmes 'installboot' auf das Bootdevice geschrieben.

3.1.2 Boot Programm Phase

1. bootblk lädt nun das eigentliche (secondary boot) Bootprogramm im root-Dateisystem. Beim 32-bit Solaris ist dies /*platform*/`uname -i`/ufsboot, beim 64-bit Solaris /*platform*/`uname -m`/ufsboot.
2. Erst jetzt wird der UNIX-Kernel, welcher aus einem HW-spezifischen und einem generischen Teil besteht, geladen.
   32-bit System: /*platform*/`uname -m`/kernel/unix und /kernel/genunix
   64-bit System: /*platform*/`uname -m`/kernel/sparcv9/unix und /kernel/sparcv9/genunix

3.1.3 Autoconfiguration Phase

Während des Systemstarts wird nach vorhandenen Geräten gesucht und benötigte Treiber hierfür können modular bei Bdarf nachgeladen werden. So wird zum Beispiel der Treiber für ein Magnetband erst beim ersten Zugriff geladen.

Der Kernel-Konfigurationsprozess kann durch die Datei **/etc/system** beeinflusst werden. Per Standard sind alle Zeilen in dieser Datei auskommentiert. Folgende Schlüsselwörter können verwendet werden:

moddir Der Suchpfad für Kernelmodule kann damit auf ein anderes Verzeichnis gelenkt werden

forceload erzwingt das Laden eines Moduls.

exclude Damit können bestimmte Module ausgeschlossen werden, welche sonst automatisch geladen würden.

rootdev ermöglicht die Angabe eines anderen root-Device.

set *variable=wert* um Kernel Parameter zu überschreiben

3.2 Der Init-Prozess

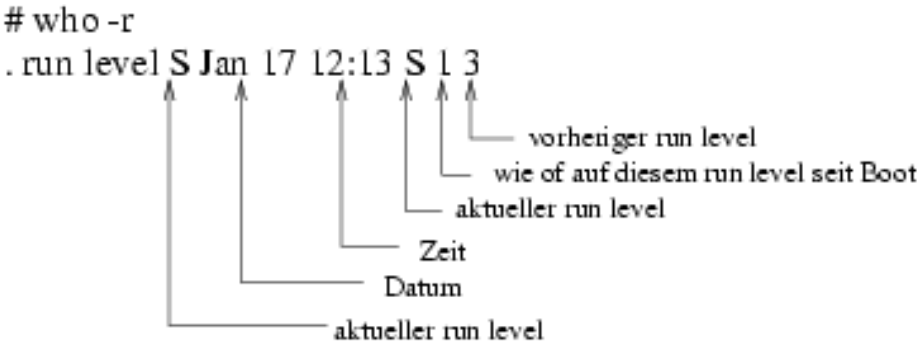
Nachdem der Kernel initialisiert wurde, wird als erster Prozess (Vaterprozess aller Prozesse) **/sbin/init** gestartet.

Seine Hauptaufgaben sind:

- Alle Prozesse zu starten, welche im hochgefahrenen System benötigt werden.
- Kontrolle bei Änderung des 'Runlevels' (gesteuert durch die Datei /etc/inittab).

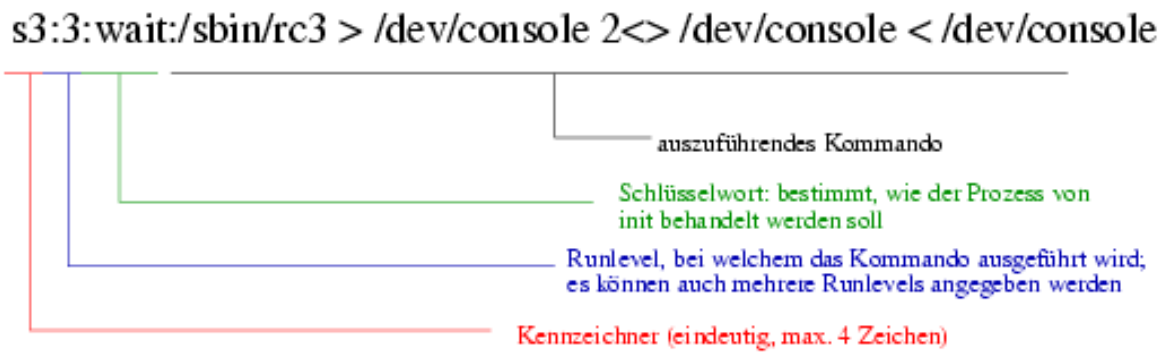
Run Level	Beschreibung
0	PROM Monitor
1	Single-user Modus; Dateisysteme ge'mounted'; User-Login nicht möglich
2	Multi-user Modus; keine Ressourcen freigegeben (NFS)
3	Multi-user Modus; Ressourcen freigegeben (Standard Runlevel bei Solaris)
4	-
5	Shutdown und (bei Sun-4m und Sun-4u) Strom aus
6	Reboot in den Standard Runlevel (3)
s, S	Single-user Modus; Dateisysteme nicht ge'mounted'; User-Login nicht möglich

Feststellen des aktuellen Runlevels:



Das Verhalten des init-Prozesses wird mit der Datei **/etc/inittab** gesteuert. Diese Datei bestimmt, welche Prozesse für welchen Runlevel gestartet werden sollen. Jede Zeile in dieser Datei hat 4 Felder, welche durch Doppelpunkt getrennt werden:

Die Bedeutung dieser Felder sind:



Im 3. Feld sind folgende Schlüsselwörter zulässig:

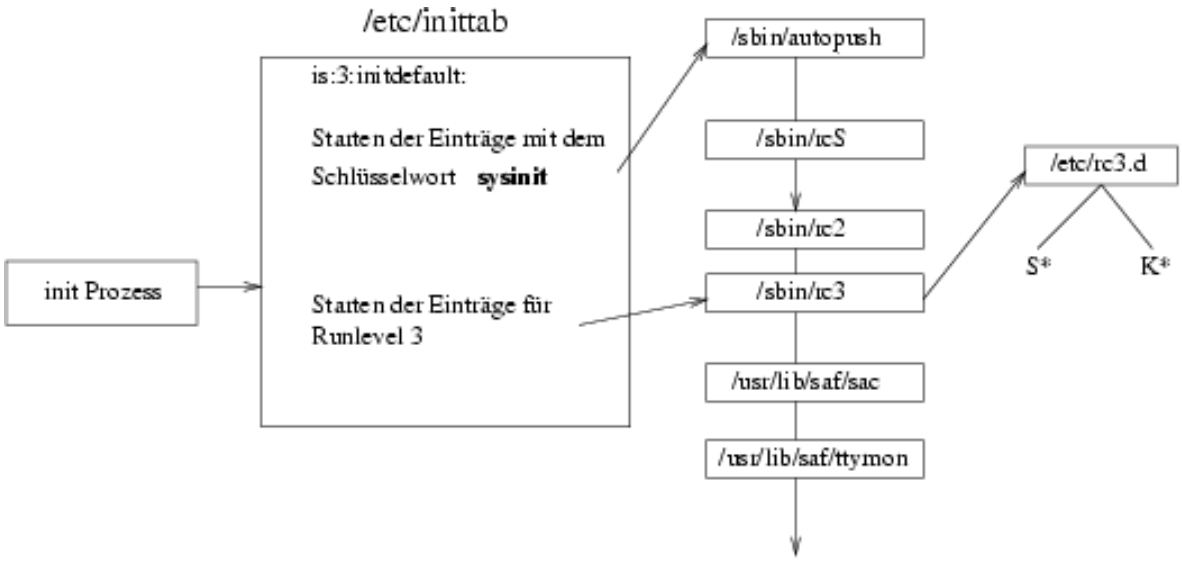
initdefault Gibt den Standard Runlevel an, in welchen das System startet. Dies ist bei Solaris Runlevel 3. Bei diesem Schlüsselwort darf im 2. Feld natürlich nur eine Ziffer stehen. Es ist auch nur ein Eintrag in der Datei zulässig.

respawn *init* startet den Prozess und überwacht, ob er läuft. Sobald der Prozess beendet wird, startet *init* den Prozess erneut.

powerfail Startet diesen Prozess, wenn *init* das Signal *power fail* erhält.

sysinit Startet den Prozess ohne Ausgabe auf die Konsole. Dies ist nötig für die ersten Prozesse während des Systemstarts. *Init* wartet auf die Beendigung des Prozesses.

wait *Init* startet den Prozess und wartet auf deren Beendigung.



1. Der *init* Prozess liest die Einträge der Datei */etc/inittab*
2. *init* sucht nach dem Eintrag *initdefault* um den gewünschten Runlevel festzustellen.
3. *init* führt die Prozesse aus, welche das Schlüsselwort *sysinit* haben (wie z.Bsp. */sbin/autopush*).
4. *init* startet das für den gewünschten Runlevel entsprechende 'run control script' (rc-Skript). Bei Solaris ist das normalerweise /sbin/rc3. Dieses Skript wiederum startet alle Skripte im Verzeichnis /etc/rc3.d welche mit dem Buchstaben 'S' (Großbuchstabe!) beginnen in alphanumerischer Reihenfolge. Die in diesem Verzeichnis befindlichen Dateien, welche mit dem Buchstaben 'K' (Großbuchstabe!) beginnen dienen dem Beenden der jeweiligen Prozesse und werden beim Wechsel in einen niedrigeren Runlevel benötigt.
5. Nun startet *init* noch alle anderen Prozesse, welche in der Datei */etc/inittab* für diesen Runlevel vorgesehen sind. Das sind üblicherweise zumindest die Portmonitore (ttymon).

Die in den Verzeichnissen */etc/rc?.d* befindlichen Dateien sind lediglich Links auf die tatsächlichen Shell-Skripte. Diese befinden sich alle im Verzeichnis */etc/init.d*. Jedes dieser Skripte kennt die Optionen *start* bzw. *stop*. Die Option *start* wird verwendet um den (die) Prozess(e) zu starten, *stop* wird verwendet um sie zu beenden. Die mit dem Buchstaben 'S' beginnenden rc-Skripte werden mit der Option *start*, die mit dem Buchstaben 'K' beginnenden rc-Skripte werden mit der Option *stop* aufgerufen.

Runscript Musterbeispiel

```
1  #!/bin/sh
2  #
3  # Beispielskript
4  #
5  # Author: Ernst Pisch      Datum letzter Änderung: 9.5.2000
6
7  # PID des Hintergrundprozesses feststellen:
8  pid=`/usr/bin/ps -e | /usr/bin/grep beispielprozess | \
9   /usr/bin/sed -e 's/^  *//' -e 's/ .*//'`
10 case $1 in
11 'start')
12         if [ "${pid}" = "" ]
13         then
14             # Hier wird der Prozess gestartet
15         fi
16         ;;
17 'stop')
18         if [ "${pid}" != "" ]
19         then
20                 /usr/bin/kill ${pid}
21         fi
22         ;;
23 *)
24         echo "usage: /etc/init.d/beispielscript {start|stop}"
25         ;;
26 esac
```

Die Zeilen 8 und 9 könnten u. U. von der Solarisversion abhängig sein! Das Suchmuster des *sed*-Kommandos muss zum Ausgabeformat des *ps*-Kommandos passen. Bei Bedarf muss man das von anderen rc-Skripten ’abschauen’.

3.3 Änderung des Runlevels

Der Wechsel zwischen Runlevels erfolgt durch das Kommando '*init*' (oder '*shutdown*').

init

Syntax:

init [-12356abcQqSs]

- **0** Wechsel in den PROM Monitor Level
- **1** Single-user Mode, Dateisysteme gemountet, User-Login nicht möglich
- **2** Multi-user Mode, keine NFS-Verzeichnisse freigegeben
- **3** Multi-user Mode, Ressourcen freigegeben
- **5** Shutdown + Strom aus
- **6** Shutdown + Reboot in Standard Runlevel (normalerweise 3)
- **a, b, c** Prozesse, welche in der Datei */etc/inittab* mit dem Runlevel *a,b* oder *c* gekennzeichnet werden, werden gestartet. Dies ermöglicht die Konfigurationvon individuellen Runlevels.
- **Q, q** *init* wird dazu veranlasst, die Datei */etc/inittab* erneut einzulesen und Prozesse zu starten bzw. zu stoppen
- **S, s** Single-user Mode

shutdown

shutdown dient ebenso dazu den Runlevel zu wechseln. Der Vorteil gegenüber dem Kommando **init** besteht darin, dass **shutdown** eine Meldung an alle angemeldeten User schickt.

Syntax:

shutdown [-y] [-g *sekunden*] [-i *runlevel*] [*Meldetext*]

- **-y** yes - es wird ohne Rückfrage ein Shutdown eingeleitet
- **-g** grace period - Anzahl Sekunden bevor der Shutdown beginnt
- **-i** init state - Angabe des Runlevels in den gewechselt werden soll

Meldetext Angabe eines Meldetextes, den alle angemeldeten User erhalten

Wird kein Schalter angegeben, wird in den Single-user Modus gewechselt.

Es befindet sich im Verzeichnis */usr/ucb/bin* ebenfalls ein Kommando *shutdown*, welches aber aus der ucb-Umgebung kommt, und andere Optionen kennt. Diese Variante wird aufgerufen, wenn das Verzeichnis */usr/ucb/bin* vor dem Verzeichnis */usr/sbin* in der Umgebungsvariable PATH eingetragen ist

halt

halt fährt das System in den Runlevel 0, aber ohne die rc0-Scripts zu durchlaufen!

poweroff

poweroff wechselt in den Runlevel 5 (d.h. die Maschine wird abgeschaltet bei Sun-4m und Sun-4u), durchläuft aber nicht die rc0-Scripte!

reboot

reboot führt einen Reboot durch, durchläuft aber ebenfalls nicht die rc0-Scripte! Es können auch Boot-Optionen mitgegeben werden, welche dann vom *boot* Kommando des PROM ausgewertet werden.

Bsp.: `# reboot -- -s` `# Reboot in den Single-user Modus`

wall

wall dient nicht direkt dem Wechsel des Runlevels, wird aber speziell in diesem Zusammenhang benötigt. Damit ist es möglich, angemeldeten Benutzern eine Nachricht auf den Bildschirm zu schicken.

Bsp.: `# wall -a messagefile` `# Inhalt der Datei 'messagefile' wird an alle Terminals (auch Pseudoterminals) geschickt`

Die Bedeutung des Kommandos schwindet aber immer mehr, da immer mehr Client-Server Anwendungen Verwendung finden. Solche Anwender können mit *wall* nicht benachrichtigt werden!

3.4 Boot PROM

Der BootPROM Chip sitzt am Systemboard. Die Daten werden in einem NVRAM gespeichert.

Revision-Nummern

1.x original SPARC boot PROM

2.x OpenBoot PROM (OBP)

3.x OBP mit downladbarer Firmware

3.4.1 Funktionen des Boot PROMs

- Power-on self-test (POST)
- initialisiert Boards
- generische Gerätetreiber - meldet Hardware-Konfiguration ans Betriebssytem
- User Interface an Konsole bzw. (wenn Konsole + Tastatur abgesteckt) an Terminal über 1. serielle Schnittstelle
- Datum / Uhrzeit
- MAC-Adresse
- Standardeinstellungen zu Bootdevice, Ein- Ausgabegerät, ...

3.4.2 Wichtige Tastenkombinationen

<STOP> während Power-On — kurzer POST (Power-on self-test); man gelangt ins Boot PROM User-Interface (ok Prompt); Der Wechsel in den OBP wird auch bei laufendem System durchgeführt. Deshalb ACHTUNG, denn alle Prozesse werden eingefroren -> niemals im Multiuser-Modus anwenden. Fortgesetzt wird das System durch das Kommando **go**.

<STOP>+<d> intensive Power-on self-tests

<STOP>+<n> NVRAM rücksetzen auf Standardwerte

<STOP>+<a> Unterbrechen des Systems und Wechsel in Boot PROM User-Interface (ok Prompt)

Korrekter Einstieg in OBP:

1. Reboot
2. ok **reset**
3. <STOP>+<a> direkt nach Selbsttest

Wenn kein 'reset' vorher gemacht wurde, kann es u.U. zu unerwünschten Reaktionen kommen, da kein definierter Zustand ist.

Unterbrechen eines hängenden Systems:

1. <STOP>+<a> drücken, um in den BootProm zu gelangen
2. ok **sync**

Dadurch wird das laufende System gestoppt und die im Speicher gepufferten Daten werden auf Platte geschrieben.

3.4.3 OBP Kommandos

words

Anzeige aller OBP-Kommandos

help

Syntax:

help [*command-name*]

sifting

Syntax:

sifting [*muster*]

sifting ist ähnlich dem grep unter Unix. Es werden alle OBP-Kommandos aufgelistet, welche das Suchmuster enthalten.
Bsp: Es sollen alle Kommandos aufgelistet werden, welche das Wort 'probe' enthalten.
ok **sifting** probe

banner

Ausgabe von Systeminformationen wie Maschinen-Typ, Prozessortyp, Speichergröße, Seriennummer, Ethernet-Adresse, Host-ID

show-post-results

Ausgabe der Testergebnisse des POST (Power On Self Test)

boot

Syntax:

boot [*device-name*] [- {a | r | s | v}]

- **a** Boot mit Dialog - fragt nach root- und swap-device sowie nach wichtigen Systemdateien
- **r** führt Rekonfiguration beim Boot aus, erzeugt Gerätedateien für alle erkannten Geräte und aktualisiert /etc/path_to_inst
- **s** bootet System in den SingleUsermodus
- **v** detaillierte Ausgabe während des bootens

Tipp: Sollte die Datei */etc/system* falsche Einträge enthalten und ein Booten verhindern, so kann mit Hilfe der Option '*-a*' gebootet werden. Bei der Abfrage nach 'Name of systemfile [etc/system]' gibt man dann einfach */dev/null* an.

printenv

Syntax:

printenv [*variable*]

Ausgabe von NVRAM-Parametern

setenv

Syntax:

setenv *variable wert*

Setzen von NVRAM-Parametern

go

Wurde das System mit der Tastenkombianation <STOP>+<A> gestoppt und in den OBP-Modus verzweigt, so kann mit dem Kommando **go** das System wieder fortgesetzt.

Es ist zu beachten, dass während des Zeitraumes im OBP-Modus das System stillsteht!

sync

Erzwingt das Zurückschreiben der Daten im Speicher auf Platte. Damit kann bei hartnäckigen Systemhängern eventuell ein größerer Schaden verhindert werden.

reset, reset-all

Rücksetzen in einen definierten Zustand. Das sollte immer vor dem Einstieg in den OBP und immer nach Änderungen im NVRAM durchgeführt werden. **reset-all** sollte unbedingt vor den Kommandos **probe-scsi, probe-scsi-all** oder **probe-ide** ausgeführt werden.

set-defaults

Rücksetzen der NVRAM-Parameter auf Standard

probe-scsi, probe-scsi-all

Führt HW-Erkennung der SCSI-Geräte durch. **probe-scsi** macht Erkennung am Onboard SCSI-Kontroller, **probe-scsi-all** macht Erkennung an allen SCSI-Kontrollern

probe-ide

Führt HW-Erkennung der IDE-Geräte durch.

show-devs

Zeigt Geräte an (siehe auch Seite 67).

show-disks

Zeigt installierte Platten an und ermöglicht es, die langen Namen in einen Zwischenspeicher zu kopieren. Daraus kann der Name später mit der Tastenkombination <CTRL>+<Y> an jeder beliebigen Stelle wieder eingefügt werden. Das ist sehr hilfreich in Verbindung mit dem Kommando *nvalias*.

nvalias, nvunalias

Mit **nvalias** können für Geräte Alias-Namen vergeben werden. Mit **nvunalias** können sie wieder gelöscht werden.

Bsp.: Die Platte mit dem Namen ’/pci@1f,0/pci@1,1/ide@3/disk@0,0’ soll den einfacheren Alias-Namen ’mydisk’ bekommen.

```
ok nvalias mydisk /pci@1f,0/pci@1,1/ide@3/disk@0,0
```

devalias

Listet alle Alias-Namen auf

nvedit, nvstore

nvedit ist ein einfacher Zeileneditor.

Wichtige Editor-Kommandos	
^C	Editor beenden
^U	aktuelle Zeile löschen
^B	ein Zeichen nach links
^F	ein Zeichen nach rechts
^P	eine Zeile nach oben
^N	eine Zeile nach unten
Delete	Zeichen links des Cursors löschen
Return	aktuelle Zeile abschließen und neue Zeile eröffnen

nvstore speichert die mit **nvalias** oder **nvedit** eingegebenen Werte ab. Anschließend muss ein **reset** erfolgen.

3.4.4 Wichtige NVRAM-Parameter

auto-boot? false — System bleibt nach dem Einschalten im OBP stehen; true — System bootet nach dem Einschalten von jenem Gerät, welches in der Variable **boot-device** hinterlegt ist;

boot-device Gerät von dem gebootet werden soll bzw. Reihenfolge der Geräte.Bsp.:
boot-device - disk net Bootversuch von Platte, falls das scheitert vom Netz
boot-device - disk mirror Bootversuch von 1. Platte, sonst von 'mirror'
('mirror' muss mittels **nvalias** der tatsächliche Name der Spiegelplatte zugewiesen werden)

diag-switch? false — (Standard); true — 'gesprächige' Ausgabe der Startuptests; ist *true* gesetzt, so wird von jenem Gerät geladen, welches in der Variable **diag-device** hinterlegt ist.

local-mac-address? false — (Standard) es wird die MAC-Adresse des NVRAM benützt; true — die MAC-Adresse der LAN-Karte wird verwendet

3.4.4.1 eeprom Kommando

Die NVRAM-Parameter können auch im laufenden System mit Hilfe des Kommandos **eeprom** verändert werden. Für die Verwendung des Kommandos benötigt man Superuser-Rechte.

Beispiel:

```
# eeprom "auto-boot?"=true
```

3.5 POST (Power-on self-test)

Jedes Systemboard besitzt eine eigene Firmware (kann geflasht werden) für den Selbsttest. Ebenso hat jedes Board LED's welche gelb leuchten, falls der Test fehlerhaft war.

Detaillierte Ausgaben erfolgen nur über die 1. serielle Schnittstelle - nicht auf die Konsole! —> Konsole und Tastatur abstecken und stattdessen ein Terminal an die serielle Schnittstelle hängen.

3.5.1 Kontrolle von POST

OBP (Open Boot PROM) Über Variablen im Boot PROM kann kontrolliert werden, ob intensiv getestet werden soll, ob 'verbose' eingeschaltet ist und ob die Temperaturüberwachung aktiviert ist.

diag-switch true — 'verbose' Ausgabe; false — 'verbose' ausgeschaltet

diag-level min — minimaler Test; max — intensiver Test

mfg-env CHAMBER — Temperaturüberwachung AUSGESCHALTET!; OFF — Temperaturüberwachung aktiv (normal)

Schlüsselschalter 'diagnostic position' — maximaler Testlevel wird aktiviert
'normal position' — der Test erfolgt laut Einstellungen des OBP

3.5.2 POST Board Statusmeldungen

Es wird für jeden Slot eine Meldung ausgegeben. Folgende Ergebnisse sind möglich:

Normal Test fehlerfrei

On-line/Failed mindestens eine Komponente des Boards fehlerhaft

Low Power Mode Test fehlerhaft, das Board wurde in den 'low power mode' geschaltet und kann im laufenden Betrieb getauscht werden

Not Installed es wurde in diesem Slot kein Board gefunden

Speicherfehler: Bei einem großen Server ist ein ECC-Fehler pro Tag akzeptabel!

4 Software Pakete und Patches

4.1 Software Package Administration

Alle Solaris Softwarepakete von SUN werden in Form des AT&T Package-Formates ausgeliefert.

Ein Package beinhaltet:

- Dateien, welche das Softwarepaket beschreiben
- Dateien, welche Abhängigkeiten und benötigten Plattenplatz beschreiben
- die zu installierenden Dateien und Verzeichnisse der jeweiligen Software
- Skripte, welche vor und nach der Installation bzw. Deinstallation laufen

Mit der grafischen Oberfläche von **admintool** kann Software installiert und deinstalliert werden. Die zugrundeliegenden Kommandos können aber auch von der Shell aus aufgerufen werden. Die wichtigsten Kommandos sind:

pkginfo

Syntax:

pkginfo [-x|-l] [-d *device*] [*pkg-name*]

-x zeigt alle wichtigen Paketinformationen wie Name, Architektur und Version an

-l zeigt alle Paketinformationen an

-d *device* Software-Paket befindet sich auf *device*

pkginfo zeigt Informationen zu den installierten Paketen an.

pkgadd

Syntax:

pkgadd [-d {*device*|*pathname*}] [*pkg-name*]

-d *device* Das Softwarepaket wird von *device* gelesen. *device* ist entweder der vollständige Pfadname oder ein Gerätename

pkgadd installiert ein Package-Softwarepaket von einem Datenträger oder einer Datei.

pkgrm

Syntax:

pkgrm *pkg-name*

pkgrm deinstalliert ein Softwarepaket. Es werden dabei eventuelle Abhängigkeiten berücksichtigt.

pkgchk

Syntax:

pkgchk [-l] [-p *pfadname*] [*pkg-name*]

-l listet alle zu einem Pakete gehörigen Dateien auf

-p *pfadname* Es wird nur *pfadname* überprüft.

pkgchk überprüft installierte Pakete und stellt fest, ob die Verzeichnisse und Dateien OK sind.

4.1.1 Dateien und Verzeichnisse der Package-Verwaltung

Datei oder Verzeichnis	Beschreibung
/var/sadm	Verzeichnis, welches Logdateien und Administrationsdateien enthält
/opt/*paketname*	Bevorzugter Installationspfad der Pakete
/opt/bin oder /opt/*paketname*/bin	Bevorzugter Installationspfad der Programmdateien der Pakete
/var/opt/*paketname* oder /etc/opt/*paketname*	Bevorzugter Pfad der Logdateien der installierten Pakete
/var/sadm/install/contents	Datei welche Daten zu allen installierten Paketen enthält

4.2 Umgang mit Patches

Patches sind Korrekturen oder Funktionserweiterungen zu vorhandenen Softwarepaketen. Jeder Patch hat eine README-Datei welche u. a. Funktion bzw. behobene Fehler beschreibt. Die Patches können übers Internet bei SUN unter `http://sunsolve.sun.com` oder per anonymous-ftp bei `ftp://sunsolve.sun.com` bezogen werden.

Bereits installierte Patches können mit den Kommandos **patchadd -p** (ab Solaris 2.6) oder **showrev -p** aufgelistet werden.

Patches werden als tar-Files in 3 verschieden komprimierten Formaten ausgeliefert und müssen mit dem jeweiligen Programm entpackt werden: Bei Suffix **.tar.Z** mit **uncompress** oder **zcat**, Suffix **.tar.gz** mit '**gzip -d**' oder **gzcat** und bei Suffix **.zip** mit **unzip.** Das Programm **gzip** ist nicht im Standard-Solaris enthalten, befindet sich aber auf den patch-CD's oder kann von `http://www.gzip.org/` geladen werden.

Die Installation eines Patches geschieht wie folgt (ab Solaris 2.6):

- komprimiertes Patchfile nach /tmp kopieren (z.Bsp.: 105030-01.tar.Z)
- **cd /tmp**
- komprimierte Datei auspacken
  zcat *./105030-01.tar.Z* | **tar xvf -**
  oder
  gzcat *./105030-01.tar.gz* | **tar xvf -**
- Lesen der README-Datei (unter anderem Kontrolle, ob passend für das System)
- **patchadd** *105030-01*
  Sollte nicht genügend Platz im Verzeichnis '/var/sadm/patch' vorhanden sein, kann das Kommando mit der Option '-d' aufgerufen werden. Es werden dann keine Dateien nach '/var/sadm/patch' kopiert. Aber ACHTUNG!! Es ist dann nicht mehr möglich, Patches wieder zu entfernen!

Patch-Cluster, wie z.Bsp. die sogenannten *Recommended Patches* für eine bestimmte Solaris Version, enthalten ein Paket von vielen Patches. Um diese in der richtigen Reihenfolge zu installieren, muss einfach das mitgelieferte Skript **install_cluster** gestartet werden.

Bei älteren Solaris Versionen ist anstatt *patchadd* das Kommando **installpatch** und anstatt *patchrm* das Kommando **backoutpatch** zu verwenden.

Entfernt können Patches mit dem Kommando '**patchrm** *patchnummer*' werden, falls nicht bei der Installation des Patches die Option '-d' angegeben wurde.

4.2.1 Was passiert während der Patch Installation?

Nach der Installation des Patches befindet sich im Verzeichnis /var/sadm/patch/*105160-05* (im Beispiel wurde der Patch 105160-05 installiert) ein Log-File mit dem Installationsprotokoll, die in der Abbildung dargestellten Dateien und alle benötigten Daten falls der Patch wieder entfernt werden soll.

Im Verzeichnis /var/sadm/pkg/*SUNWdtbas* (in unserem Beispiel das betroffene Paket) wird die Paketinformation 'pkginfo' entsprechend aktualisiert.

5 Systemsicherheit und Zugriffsrechte auf Dateien

5.1 Passwort Sicherheit

Jede Benutzerkennung soll ein Passwort erhalten, welches immer wieder geändert werden sollte. Das Passwort wird verschlüsselt in der Datei */etc/shadow* abgelegt. Diese Datei ist nur für die Kennung *root* lesbar.

UID - User Identifier

Jeder Benutzerkennung wird eine Nummer, die sogenannte **UID** zugeordnet. Diese Nummern stehen in der Datei */etc/passwd*.Die Nummern von 0 bis 99 sind reserviert für spezielle Systemkennungen und sollten nicht verwendet werden. Die UID ’0’ ist dem Superuser *root* zugeordnet und verleiht ’Allmacht’.

GID - Group Identifier

Die **GID** ist eine Nummer, die eine Gruppe von Benutzerkennungen kennzeichnet. Ebenso wie die UID steht die GID in der Datei */etc/passwd*. Ein User kann aber Mitglied von bis zu 16 Gruppen sein. Die GID, welche in der Datei /etc/passwd steht, gibt die *primary group* des Users an. Alle zusätzlichen ’Mitgliedschaften’ in anderen Gruppen (*secondary groups)*werden in der Datei */etc/group* geführt. Auch bei den GID’s sind die Nummern 0 bis 99 reserviert für Systemgruppen.

/etc/passwd Datei

Diese Datei ist eine Textdatei, welche alle lokalen Benutzerkennungen enthält und ist ein wesentlicher Bestandteil der Systemsicherheit. Jeder Eintrag enthält 7 Felder, die Felder sind durch das Zeichen ’:’ getrennt:

USERNAME:X:UID:GID:KOMMENTAR:HOME-VERZEICHNIS:LOGIN-SHELL

Felder:

Username Name der Benutzerkennung, mit dem sich der Anwender am System anmeldet

x Platzhalter für das verschlüsselte Passwort, welches in der Datei */etc/shadow* gespeichert ist. In älteren Systemen war das Passwort noch nicht in einer eigenen Datei abgelegt, sondern an dieser Stelle.

UID User Identifier - Nummer, die dieser Benutzerkennung zugeordnet ist

GID Group Identifier - Nummer der primären Gruppe, der der Benutzer zugeordnet ist

Kommentar beliebiger beschreibender Text, wie z.Bsp. Telefonnummer, Name, Zimmernummer etc.

Home-Verzeichnis Verzeichnisname (absoluter Pfadname) in dem sich der Benutzer nach dem Login befindet.

Login-Shell Name des Programmes (meist eine Shell) welches nach der Anmeldung gestartet wird. Das kann auch eine Anwendung sein, sodass der Benutzer keine Möglichkeit hat dieses Programm zu verlassen ohne sich damit auch gleich abzumelden.

/etc/shadow Datei

/etc/shadow enthält das verschlüsselte Passwort und einige Passwort relevanten Daten. Nur der Superuser hat Zugriff auf diese Datei. Jeder Eintrag enthält 8 Felder, welche durch das Zeichen ':' voneinander getrennt sind:

USERNAME:PASSWORT:LETZTE ÄNDERUNG:MINIMUM:MAXIMUM:WARNUNG:INAKTIV:ABLAUF

Username Name der Benutzerkennung (Login-Name)

Passwort verschlüsseltes Passwort (13 Stellen); Hat das Passwort nicht 13 Stellen, so ist diese Kennung gesperrt und ein Login ist nicht möglich.

Letzte Änderung Anzahl der Tage zwischen dem 1.1.1970 und dem Tag der letzten Passwortänderung.

Minimum Minimale Anzahl an Tagen die verstreichen muss, bevor das Passwort wieder geändert werden darf.

Maximum Maximale Anzahl an Tagen, die verstreichen darf, ohne dass das Passwort geändert werden muss.

Warnung Anzahl der Tage vor Ablauf des Passwortes wenn bei der Anmeldung eine Warnmeldung ausgegeben werden soll. Danach erscheint bei jeder Anmeldung eine Warnung.

Inaktiv Anzahl an Tagen, die ein Benutzer keine Anmeldung durchzuführen braucht ohne gesperrt zu werden. Wird der Zeitraum ohne Anmeldung überschritten, wird die Kennung gesperrt.

Ablauf Datum, an dem die Benutzerkennung ungültig wird. Nach diesem Datum ist die Kennung gesperrt.

/etc/group Datei

Die Datei enthält die Namen aller Gruppen und deren Mitglieder. Jede Zeile enthält 4 durch das Zeichen ':' getrennte Felder:

GRUPPENNAME:PASSWORT:GID:BENUTZERKENNUNG[,BENUTZERKENNUNG]

Gruppenname Name der Benutzergruppe

Passwort vorgesehen für ein Gruppenpasswort; dieses Feld wird bei Solaris nicht verwendet

GID Gruppenidentifier - Nummer der Gruppe;

Benutzerkennung Liste aller Benutzerkennungen (durch Komma getrennt) welche dieser Gruppe als *secondary group* zugehören.

Superuser Kennung

Die Benutzerkennung mit der UID 0 wird **Superuser** oder **root user** genannt. Diese Kennung hat keinerlei Einschränkungen und sollte deshalb nur verwendet werden, wenn nötig.

Solche Tätigkeiten sind:

- Shutdown
- Datensicherung und Datenrestaurierung
- Mounten und unmounten von Dateisystemen
- Benutzeradministration

sysadmin Gruppe

Mitglieder der Gruppe **sysadmin** (GID = 14) haben die Möglichkeit diverse Administrationstätigkeiten durchzuführen, welche dem Normalbenutzern nicht ermöglicht werden. Das betrifft vor allem die Verwendung von **admintool**.

Unter anderem kann diese Gruppe User, sowie Drucker konfigurieren und entfernen. Es ist dieser Gruppe aber nicht möglich die betroffenen Dateien direkt zu editieren. Die Rechte erhalten diese nur mit Hilfe des admintool.

id Kommando

Damit kann festgestellt werden, welche UID und GID eine Benutzerkennung besitzt, sowie welchen Gruppen er angehört.

Syntax:

id [-a] [username]

Die Option *-a* gibt auch socondary groups aus.

su Kommando (Switching Users)

Mit dem Kommando **su** kann man die Rechte eines anderen Users erlangen.

Syntax:

su [-] [username]

Ohne Angabe eines Usernamens wird *root* angenommen. Bei Angabe der Option '-' werden nicht nur die Rechte des Users angenommen, sondern es wird auch */etc/profile* und dessen *.profile* durchlaufen.

su ist eine Möglichkeit für den Superuser die Rechte einer gesperrten Kennung (z.Bsp. bin, lp, ...) zu bekommen.

Welche Rechte die Kennung hat, kann mit dem Kommando '*id*' ermittelt werden. Der Name der aktuellen Kennung kann mit '**whoami**', der Name der ursprünglichen Kennung mit '**who am i**' festgestellt werden.

Um wieder zur ursprünglichen Kennung zurückzukehren, muss die aktuelle Shell beendet werden (mit *^D* oder *exit*).

5.2 Administration von Eigentum

Eine neue Datei oder ein neues Verzeichnis wird Eigentum desjenigen, der sie/es erzeugt hat. Nur der Superuser hat die Möglichkeit den Eigentümer zu ändern. Siehe dazu auch das Kapitel 5.5 auf Seite 37. Die Änderung erfolgt durch 2 Kommandos:

chown

chown ändert den Eigentümer der Datei bzw. des Verzeichnisses. Nur *root* kann dies tun.

Syntax:

chown [-R] {*user*|*UID*}[:*gruppe*|*GID*] *filename*

Mit der Option '*-R*' können Eigentümer rekursiv auch in Unterverzeichnissen gesetzt werden.

Anstatt des Usernamens kann auch dessen UID angegeben werden. Zusätzlich ist es möglich zusammen mit dem Eigentümer auch den Gruppenamen bzw. die GID festzulegen.

Beispiele:

Verzeichnis 'dir' und alle Unterverzeichnisse und Dateien erhalten den Eigentümer 'owner'
chown -R owner dir

Datei 'file' erhält den Eigentümer 'owner' und die Gruppe 'group'
chown owner:group file

Mehrere Dateien (readme.txt und install.txt) erhalten den Eigentümer 'pische'
chown pische readme.txt install.txt

chgrp

chgrp ändert die Gruppenzugehörigkeit der Datei bzw. des Verzeichnisses. Nur *root* hat diese Möglichkeit.

Syntax:

chgrp [-R] {*gruppe*|*GID*} *filename*

Mit der Option '*-R*' können Gruppen rekursiv auch in Unterverzeichnissen gesetzt werden.

Anstatt des Gruppennamens kann auch dessen GID angegeben werden.

5.3 /etc/default Verzeichnis

Das Verzeichnis */etc/default* beinhaltet 3 Dateien, welche die Sicherheit des Systems beeinflussen.

/etc/default/passwd Datei

Die Datei kennt 3 Variablen, welche Passwort-Eigenschaften betreffen:

MAXWEEKS Anzahl Wochen nach denen das Passwort geändert werden muss.

MINWEEKS Anzahl Wochen die mindestens verstreichen muss bevor das Passwort geändert werden darf

PASSLENGTH Anzahl der Zeichen die ein Passwort haben muss. Gültige Werte sind: 6, 7 und 8. Werte kleiner 6 werden als 6 gewertet, Zahlen größer 8 werden als 8 gewertet.

Sind die den Variablen *MAXWEEK* und *MINWEEK* entsprechenden Felder der Datei /*etc*/*shadow* belegt, haben die Werte in der Datei /etc/shadow Vorrang.

/etc/default/login Datei

Es gibt darin eine Menge von Variablen welche die Systemsicherheit beeinflussen. Die wichtigsten davon sind:

CONSOLE Lautet der Eintrag '*CONSOLE=/dev/console*', so kann sich der Superuser nur an der Konsole anmelden. Aber Achtung: Es ist möglich, sich irgendwo als Normaluser anzumelden und dann mittels *su* Kommando die Rechte des Superusers zu bekommen.
Ist die Variable auf Null ('*CONSOLE=*') gesetzt, so kann sich der Superuser nirgends anmelden. Es bleibt nur der Wechsel mit dem Kommando *su*.

PASSREQ Wenn '*PASSREQ=YES*' gesetzt ist, können sich nur Benutzer anmelden, die ein Passwort besitzen. Kennungen ohne Passwort können sich nicht anmelden.

/etc/default/su Datei

Die Variablen dieser Datei beeinflussen das Protokollieren von Userwechsel mittels Kommando '*su*'.

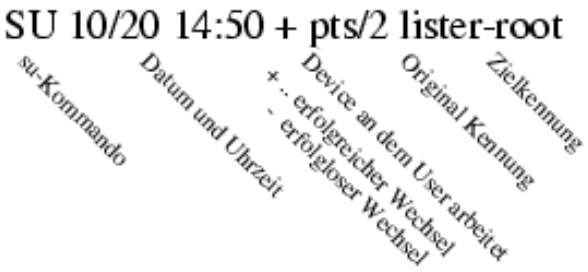
SULOG Pfad der Logdatei. Alle *su* - Aufrufe werden hier protokolliert.

CONSOLE z.Bsp.: *CONSOLE=/dev/console;* Das Protokoll kann damit zusätzlich an ein Device (im Beispiel die Konsole) geschickt werden.

SYSLOG Wenn *SYLOG=YES* gesetzt ist, wird auch noch per syslog protokolliert (z.Bsp. an einen anderen Rechner)

Es gibt noch die beiden Variablen **PATH** und **SUPATH**, welche die PATH-Variable für Normaluser bzw. Superuser voreinstellen.

Die su-Logdatei ist eine Textdatei und hat folgenden Aufbau:



5.4 Überwachung von Systemzugriffen

Zur Kontrolle der angemeldeten User stehen folgende Kommandos zur Verfügung:

who

who zeigt die zur Zeit angemeldeten User, dessen Gerätedevice und den Zeitpunkt des Anmeldens an.

Syntax:

who [-b] [-H] [-t] [-u] [-q]

- **-b** Datum und Uhrzeit des letzten Reboot
- **-H** oberhalb der Information eine Kopfzeile (Beschreibung der Felder) ausgeben
- **-t** Letzte Änderung der Systemuhr
- **-u** wer ist noch angemeldet (es wird der Inhalt der Datei */etc/adm/utmp* ausgewertet)?
- **-q** Kurzliste und Anzahl der angemeldeten Benutzer

Beispiele:

```
pisch@sunlicht:/export/home/pie # > who -b
system boot Mai 23 08:34
pisch@sunlicht:/export/home/pie # > who -Hu
NAME     LINE    TIME         IDLE PID COMMENTS
weinhapp console Mai 23 08:38 0:02 776 (:0)
root     pts/3   Mai 23 08:39 0:02 857 (:0.0)
root     pts/4   Mai 23 08:39 0:13 860 (:0.0)
root     pts/5   Mai 23 08:39 0:03 887 (:0.0)
pisch    pts/6   Mai 23 08:51 .    4552 (193.16.214.129)
pisch@sunlicht:/export/home/pie # > who -q
weinhapp root root root pisch
# users=5
```

whodo

whodo listet die laufenden Programme von angemeldeten Benutzer auf.

Syntax

whodo [-l]

- **-l** Lange Ausgabe

Beispiele:

```
pisch@sunlicht:/export/home/pie # > whodo
Die 23 Mai 2000 09:07:28 MET DST
sunlicht
console      weinhapp  8:38
?            776    0:00 Xsession
?            786    0:00 fbconsole
pts/2          823    0:00 sdt_shell
pts/2          825    0:00 dtsession
?            847    0:02 dtwm
?            848    0:00 dtterm
pts/5          887    0:00 dtksh
pts/4          860    0:00 dtksh
pts/3          857    0:00 dtksh
pts/2         4508    0:00 sh
pts/2         4509    0:07 dtfile
pts/2         4512    0:00 dtfile
pts/2          836    0:00 ttsession
?            822    0:00 dsdm
pts/3        root      8:39
pts/4        root      8:39
pts/5        root      8:39
pts/6        pisch     8:51
pts/6         4552    0:00 dtksh
pts/6         5311    0:00 whodo
pisch@sunlicht:/export/home/pie # > whodo -l
9:07am  up 0 day(s), 0 hr(s), 33 min(s)  5 user(s)
User     tty           login@  idle   JCPU   PCPU  what
weinhapp console       8:38am    29     14      2  -dtksh
root     pts/3         8:39am    17                -
root     pts/4         8:39am    35                -
root     pts/5         8:39am     4                -
pisch    pts/6         8:51am                      whodo -l
pisch@sunlicht:/export/home/pie # >
```

finger

finger liefert etwas mehr Information als who, wie im Beispiel ersichtlich ist.

```
pisch@sunlicht:~ > finger pisch
Login: pisch                            Name: Ernst Pisch
Directory: /export/home/pische          Shell: /bin/ksh
Office:  ICP ITS Innsbruck Tel. -67459
On since Sun Apr 16 10:29 (MEST) on :0,    idle 10:01, from console
On since Sun Apr 16 11:36 (MEST) on pts/0 from :0.0
Mail last read Sun Apr 16 20:24 2000 (MEST)
Plan:
Urlaub: 19.4. - 21.4.2000
8.7. -  5.8.2000
```

Die Informationen der Felder '*Name:*' und '*Office:*' liest '*finger*' aus dem Kommentarfeld der Datei '*/etc/passwd*'. Der Text von '*Plan:*' stammt aus der Datei '*$HOME/.plan*'.

last

last liefert nicht nur Daten zu noch angemeldeten Usern, sondern auch zu früheren Logins. Außerdem sind Systemstarts ersichtlich.

Es werden Usernamen, Gerätedevice, Rechnername (bei Login von einem anderen Rechner aus) Zeitpunkt des Login und Logoff, sowie die Gesamtzeit der Verbindung ausgegeben.

Syntax:

last [*username* | reboot]

username Login- bzw. Logout-Aktivitäten des Benutzers '*username*'

reboot Liste der System Reboots

```
pisch@sunlicht:~ > last
pisch     pts/1        149.202.162.77   Mon Apr 17 10:12   immer noch angemeldet
weinhapp  dtremote     whl:0            Fri Apr 14 10:43 - 11:20  (00:36)
pisch     pts/2        149.202.162.77   Thu Apr 13 11:21 - 14:08  (02:46)
pisch     pts/1        149.202.162.77   Thu Apr 13 10:52 - 14:07  (03:15)
weinhapp  console      :0               Tue Apr 11 10:31 - 15:18  (04:47)
mic       ftp          sunpc            Tue Apr 11 10:24 - 10:39  (00:15)
zarzera   pts/1        149.202.160.111  Mon Apr 10 18:18 - 18:21  (00:03)
zarzera   pts/1        149.202.160.111  Mon Apr 10 18:16 - 18:17  (00:00)
zarzera   pts/2        149.202.160.111  Mon Apr 10 18:01 - 18:01  (00:00)
zarzera   pts/2        149.202.160.111  Mon Apr 10 17:59 - 18:00  (00:01)
zarzera   console      :0               Mon Apr 10 17:18 - 17:22  (00:04)
zarzera   console      :0               Mon Apr 10 17:16 - 17:18  (00:02)
mic       ftp          sunpc            Mon Apr 10 17:15 - 17:30  (00:15)
zarzera   pts/1        149.202.160.111  Mon Apr 10 17:14 - 18:16  (01:01)
zarzera   pts/1        149.202.160.111  Mon Apr 10 17:14 - 17:14  (00:00)
root      console      :0               Mon Apr 10 17:11 - 17:16  (00:04)
```

logins

Syntax:

logins [-m] [-u]

-u Nur User mit UID > 99 werden aufgelistet.

-m Es werden auch Secondary Group Name und Secondary Group ID angezeigt

logins dient der Auflistung der konfigurierten Login-Kennungen. Die Liste enthält:

- Name der Benutzerkennung
- UID (user id)
- Kommentarfeld der Datei /etc/passwd
- Name der primary group
- GID (primary group id)

- Secondary group names
- Secondary group ID's
- Home Directory

Beispiele:

```
pisch@sunlicht:/export/home/pie # > logins
root            0       other           1       Super-User
smtp            0       root            0       Mail Daemon User
daemon          1       other           1
bin             2       bin             2
sys             3       sys             3
adm             4       adm             4       Admin
uucp            5       uucp            5       uucp Admin
nuucp           9       nuucp           9       uucp Admin
ppp             10      uucp            5       Solstice PPP 3.0 pppls
listen          37      adm             4       Network Admin
lp              71      lp              8       Line Printer Admin
testusr         1001    staff           10
user1           1002    other           1       User fuer Testzwecke
nobody          60001   nobody          60001   Nobody
noaccess        60002   noaccess        60002   No Access User
nobody4         65534   nogroup         65534   SunOS 4.x Nobody
pisch@sunlicht:/export/home/pie # > logins -mu
testusr         1001    staff           10
user1           1002    other           1       User fuer Testzwecke
staff           10
nobody          60001   nobody          60001   Nobody
noaccess        60002   noaccess        60002   No Access User
nobody4         65534   nogroup         65534   SunOS 4.x Nobody
pisch@sunlicht:/export/home/pie # >
```

5.5 Read, Write, Execute Permission und umask

Siehe dazu auch das Kapitel 5.2 auf Seite 30. Unix unterscheidet bezüglich Zugriffsrechten auf Dateien 3 unterschiedliche Benutzergruppen:

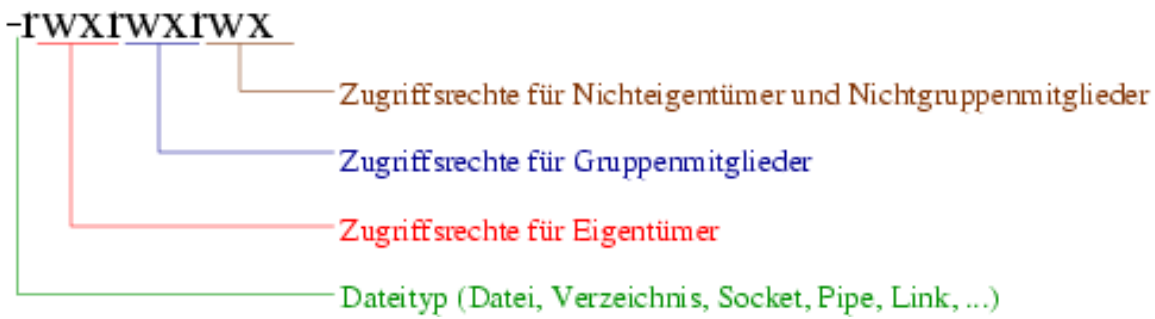
owner Eigentümer der Datei

group Benutzerkennungen können in Gruppen zusammengefasst werden. Welche Rechte ein Benutzer auf eine Datei hat, die einem anderen Benutzer derselben Gruppe hat, wird durch die 'group-permission' bestimmt.

other 'Rest der Welt' bzw. alle Benutzer, welche weder Eigentümer noch Mitglied der Gruppe des Eigentümers sind.

Die Zugriffsrechte jeder Datei können mit Hilfe des 'ls'-Kommandos mit der Option '-l' aufgelistet werden:

```
# ls -l
-rw-r--r-- 1  root     root    2479 Apr 9 02:50  README
drwxr-xr-x 6  gastuser users   1024 Feb 23 10:09 gastuser
drwxr-xr-x 27 pische   develop 3072 Apr 9 02:29  pische
#
```



Die oben dargestellten Rechte können auch als 3-stellige Oktalzahl dargestellt werden:

Bsp.: r w x r - x - - x entspricht 751

		Owner	Group	Other
r	Lesen	4	4	0
w	Schreiben	2	0	0
x	Ausführen	1	1	1
		7	5	1

umask Filter

umask legt die Standard-Zugriffsrechte beim Erzeugen von Dateien oder Verzeichnissen für einen Benutzer fest. Der Standardwert von umask ist 022. Der Wert ist als Maske mit Oktalwerten zu verstehen. Die erste Stelle steht für den Eigentümer, die zweite für die Gruppe und die dritte für alle anderen.

Um die tatsächlichen Zugriffsrechte zu erhalten, muss für Verzeichnisse der umask-Wert von 777 (oktal) bzw. für Dateien der umask-Wert von 666 (oktal) abgezogen werden. Beim Standardwert für die umask (022) ergibt sich damit für Verzeichnisse 755 und für Dateien 644.

Einige sinnvolle Werte für umask fasst folgende Tabelle zusammen:

umask Wert	Datei		Verzeichnis	
022	644	r w -r - - r - -	755	r w x r - x r - x
027	640	r w - r - - - - -	750	r w x r - x - - -

Der Wert von umask wird mit dem Kommando **umask** gesetzt.

Syntax:

umask [*Oktalwert*]

Bsp.: # **`umask 027`**

5.6 setuid und setgid Permissions

Es gibt Situationen in denen ein normaler User auf geschtützte Systemdateien Schreibzugriff benötigt. Das ist zum Beispiel der Fall, wenn ein User sein Passwort ändern will. Das Passwort wird in die Datei */etc/shadow* eingetragen. Darauf hat ein User nicht einmal Leserecht und noch weniger Schreibrecht. Ähnliches gilt für Druckspool-Dateien. Wenn ein User einen Druckauftrag absetzt, so muss dieser eingespoolt werden. Der User darf aber keine Schreibrechte darauf besitzen, da er sonst ja Aufträge anderer User verändern oder löschen könnte.

Lösung für dieses Problem bieten die beiden Zugriffsbits *setuid* und *setgid*.:
Zugriff wird den Usern nur kontrolliert durch ein Programm gestattet. Im Falle der Datei */etc/passwd* ist dies das Programm */bin/passwd*. Wenn ein User das Programm */bin/passwd* aufruft, so erhält der User über dieses Programm die Rechte der Benutzerkennung *root*.

Beispiel:

```
$ ls -l /etc/shadow /bin/passwd
-rw-------   1 root     sys       1166 Apr 16 20:41 /etc/shadow
-rwSr-Sr-x   1 root     sys      27456 Nov 13 12:29 /bin/passwd
```

setuid- bzw. setgid-Bit bei ausführbaren Programmen

Das s-Bit des Eigentümers setzt *setuid Permissions*, das s-Bit der Gruppe setzt *setgid Permissions*. Das bedeutet: Wird das Programm von jemandem aufgerufen, so läuft das Programm mit der effektiven UID (bei setuid) bzw. mit der effektiven GID (bei setgid) des Eigentümers dieses Programmes. Für unser Beispielprogramm *passwd* bedeutet das: Das Programm läuft mit den Eigentümerrechten von root und den Gruppenrechten von sys.

ACHTUNG: Wird das s-Bit als Großbuchstabe angezeigt, so weist das auf eine fehlerhafte Rechtevergabe hin! Es wurde das s-Bit gesetzt, aber nicht das execute-Bit (x-Bit), was natürlich keinen Sinn macht!

setgid-Bit bei Verzeichnissen

Ist das s-Bit der Gruppe (setgid) bei einem Verzeichnis gesetzt, so erhalten die darunter erzeugten Dateien dieselbe Gruppe wie das Verzeichnis. Das s-Bit des Eigentümers (setuid) hat bei Verzeichnissen keine Auswirkung!

Diese Eigenschaft findet Verwendung bei freigegebenen Verzeichnissen, auf die verschiedenste Anwender Zugriff haben sollen. Man kann so z.Bsp. User, die auf dieses Verzeichnis Zugriff haben sollen, in dieselbe secondary group aufnehmen. Damit sind Dateien, die im Homeverzeichnis erzeugt werden geschützt (diese erhalten ja die primary group des Users), im Share-Verzeichnis erzeugte Dateien können aber von allen Mitgliedern der entsprechenden secondary group gelesen bzw. bearbeitet werden.

Setzen von setuid-Bit

```
# chmod 4755 setuid-program
```

oder

```
# chmod u+s setuid-program
```

Setzen von setgid-Bit

```
# chmod 2755 setgid-program
```

oder

```
# chmod g+s setgid-program
```

5.7 Sticky Bit

Im Verzeichnis */tmp* ist es allen Usern erlaubt Dateien zu erzeugen, umzubenennen oder zu löschen. Dazu muss das Verzeichnis Schreibrechte für alle vergeben haben. Es muss aber verhindert werden, dass User fremde Dateien umbenennen oder löschen können. Das erreicht man durch das Setzen des *sticky-Bit* (*t-Bit*).

Ist ein Verzeichnis allgemein beschreibbar und das sticky-Bit ist gesetzt, so können Dateien in diesem Verzeichnis nur unter mindestens einer der folgenden Bedingungen umbenannt oder gelöscht werden:

- Der User ist Eigentümer der Datei
- Der User ist Eigentümer des Verzeichnisses
- Der User hat Schreibrechte auf diese Datei
- Der User ist Superuser (root)

Setzen des sticky-Bits

```
# chmod 1777 Verzeichnis
```

oder

```
# chmod o+t Verzeichnis
```

Früher, in Zeiten des akuten Speichermangels, wurde dieses Bit für ausführbare Programme genützt. Das Bit wurde für häufig verwendete Programme gesetzt, um dem System anzuzeigen, dass es resident im Speicher lagern sollte. Heute hat diese Bit für Programme oder auch sonstige Dateien keine Bedeutung mehr.

5.8 Übersicht - Zugriffsrechte

Zugriffsrecht	Datei	Verzeichnis
r - - - - - - - -	Leserecht für Eigenümer	erlaubt Eigentümer opendir() und readdir(); damit ist dieAuflistung der Dateinamen, aber nicht deren Eigenschaften möglich
- - - r - - - - -	Leserecht für Gruppenmitglied	erlaubt Gruppenmitgliedern opendir() und readdir()
- - - - - - r - -	Leserecht für Nichteigentümerund Nichtgruppenmitglieder	erlaubt Nichteigentümern und Nichtgruppenmitgliedern opendir()und readdir()
- w - - - - - - -	Schreibrecht für Eigentümer	erlaubt Eigentümer anlegen, umbenennen und löschen von Dateienin diesem Verzeichnis
- - - - w - - - -	Schreibrecht für Gruppenmitglieder	erlaubt Gruppenmitgliedern anlegen, umbenennen und löschenvon Dateien in diesem Verzeichnis
- - - - - - - w -	Schreibrecht für Nichteigentümerund Nichtgruppenmitglieder	erlaubt Nichteigentümern und Nichtgruppenmitgliedern anlegen,umbenennen und löschen von Dateien in diesem Verzeichnis
- - x - - - - - -	Ausführrecht für Eigentümer	erlaubt dem Eigentümer die Anzeige von Dateieigenschaften, denWechsel in das Verzeichnis und das Öffnen bzw. Ausführen vonDateien im Verzeichnis
- - - - - x - - -	Ausführrecht für Gruppenmitglieder	erlaubt dem Gruppenmitglied die Anzeige von Dateieigenschaften, den Wechsel in das Verzeichnis und das Öffnen bzw. Ausführen vonDateien im Verzeichnis
- - - - - - - - x	Ausführrecht für Nichteigentümerund Nichtgruppenmitglieder	erlaubt Nichteigentümern und Nichtgruppenmitgliedern die Anzeigevon Dateieigenschaften, den Wechsel in das Verzeichnis und das Öffnenbzw. Ausführen des Programmes im Verzeichnis
- - s - - - - - -	ausführender Prozess erhälteffektive UID	SUID-Bit ist für Verzeichnisse nicht relevant
- - - - - s - - -	ausführender Prozess erhälteffektive GID	wird eine Datei in diesem Verzeichnis erzeugt, erhält sie dieselbeGruppe wie das Verzeichnis
- - - - - - - - t	früher: Programm wird resistent in Speicher geladen;jetzt nicht mehr relevant	Dateien in diesem Verzeichnis können nur von dessen Eigentümer,dem Eigentümer des Verzeichnisses oder vom Superuser gelöscht werden.Bsp.: chmod 1777 /tmp

5.9 Access Control Lists (ACLs)

Zusätzlich zu den in UNIX üblichen Zugriffsrechten ’read, write, execute’ für ’owner, group, other’ unterstützt Solaris beim ufs-Dateisystem eine darüber hinaus gehende Rechtevergabe.

Ob einer Datei diese zusätzlichen Rechte vergeben wurden, sieht man mit dem Kommando ’ls -l Datei’. Im Falle der Verwendung von ACL wird das Zeichen ’+’ anschließend an die üblichen Rechte angezeigt.

Bsp: # **ls -l testdatei.txt**

```
-rwxr-----+ 1 root other 154 Nov 11 11:12 testdatei.txt
```

ACLs werden mit dem Kommando **setfacl** gesetzt bzw. verändert und mit dem Kommando **getfacl** abgefragt.

setfacl

Syntax:

setfacl [-r] -s ACL-Einträge Datei . . .

setfacl [-r] -md ACL-Einträge Datei . . .

setfacl [-r] -f ACL-Datei Datei . . .

-m setzt oder ändert eine ACL

-s ersetzt die komplette ACL durch eine neue ACL

-d löscht ACL Einträge

-r errechnet ACL Rechte neu

ACL Eintrag	Bedeutung
u[ser]::perms	Rechte (permissions) für Eigentümer
g[roup]::perms	Rechte für die Gruppe des Eigentümers
o[ther]::perms	Rechte für Nichteigentümer und Nichtgruppenmitglieder
m[ask]:perms	ACL-Maske. Setzt maximal erlaubte Rechte für Nichteigentümer
u[ser]:uid:perms	Rechte für einen bestimmten User
g[roup]:gid:perms	Rechte für eine bestimmte Gruppe

Werden die Rechte nicht in Oktalform, sondern symbolisch angegeben, müssen diese vollständig angegeben werden (also z.Bsp. *rw-* anstatt *rw*).

Beispiele:

```
# setfacl -m user:ssa20:6 ch3.doc      # Lese- und Schreibrecht für den User ssa20
# setfacl -d user:ssa20:6 ch3.doc                          # ACL-Eintrag löschen
# setfacl -m group:other:5 file2   # nur Lese- und Ausführrecht für Gruppe 'other'
# setfacl -s user::rw-,group::r--,other:---,mask:rw-,user:ssa20:rw- ch1.txt
# Eigentümerrecht:read/write;Gruppenrechte:read only; other:none;
# Benutzer 'ssa20':read/write;mask permissions: read/write ->
# kein User oder Gruppe hat Ausführrecht;
```

getfacl

Syntax:

getfacl [-ad] *Datei* ...

-a Zeigt Dateiname, Eigentümer, Gruppe und ACL-Einträge der Datei bzw. des Verzeichnisses

-d Zeigt Dateiname, Eigentümer, Gruppe und Standard ACL-Einträge für das Verzeichnis

6 Prozess Kontrolle

6.1 Überwachung von Prozessen

ps (Prozess Status) Kommando

Das Kommando **ps** listet Prozesse auf. Die Information dazu erhält das Kommando aus dem Pseudodateisystem */proc*. Der Umfang der Ausgabe hängt nicht nur von der Angabe der Optionen ab, sondern auch davon, ob ein normaler User oder der Superuser das Kommando aufruft. Der Superuser erhält eine Liste der Prozesse inklusive der Optionen mit denen die Prozesse gestartet wurden.

Syntax:[1]

ps [-e] [-f] [-p *PID*]

- **-e** Auflistung aller Prozesse
- **-f** 'full listing'; gibt Benutzer, PID, PPID, Startzeit, Terminal, verbrauchte Zeit und Name aus
- **-p *PID*** Ausgabe von Prozessinformationen des Prozesses mit der *PID*

PID (Prozess-Identifier) ist die Nummer des Prozesses
PPID (Parent Prozess-Identifier) ist die Nummer des Vaterprozesses. Das ist meistens jener Prozess, der diesen Prozess erzeugt hat. Bei Dämon-Prozessen, wird *init* zum Vaterprozess (PID=1) gemacht, damit der eigentliche Vaterprozess beendet werden kann.

kill Kommando

Mit Hilfe des **kill** Kommandos kann man Prozessen ein Signal schicken. Meist macht man dies um einen Prozess zu beenden. Das muss aber nicht immer sein. Zum Beispiel kann man manche Dämon-Prozesse (inetd, syslogd,...) durch senden des *HUP*-Signales dazu veranlassen, sich erneut zu initialisieren.

Das *kill* Kommando ist Bestandteil der Shell, existiert aber auch als eigenes Kommando.

Syntax:

kill [-l] [-SIGNAL] PID [PIDs]

-l Liste aller Signale, welche das Betriebssystem unterstützt

Als Signal kann entweder die Zahl des jeweiligen Signals oder dessen Name (Option *-l*) angegeben werden. Wird kein Signal angegeben, so wird Signal 15 (TERM) verwendet. Eine kurze Beschreibung der Signale kann der Datei */usr/include/sys/signal.h* entnommen werden. In Verbindung mit dem Kommando *kill* sind eigentlich nur die Signale *HUP, TERM* und *KILL* von Bedeutung.

HUP ist jenes Signal, das ein Prozess erhält, wenn ein Terminal die Verbindung zum System beendet. Auf dieses Signal reagieren einige Dämonen mit einem Neustart. Das wird z.Bsp. für *syslogd*, *cron* und *inetd* verwendet, nachdem man deren Konfigurationsdateien verändert hat. Dadurch wird die neue Konfiguration *scharf*.

KILL beendet einen Prozess *gewaltsam* (Programme können dieses Signal nicht abfangen und sich definiert beenden).

Hilfreich ist auch manchmal das Kommando **fuser**. Damit können jene Prozesse ermittelt werden, die eine bestimmte Datei geöffnet haben. Mit der Option **-k** können diese Prozesse auch gleich beendet werden.

Beispiel:

```
# fuser -k /dev/pts/1       # beenden aller Prozesse, welche 'an /dev/pts/1 hängen'
# fuser -u /var/adm/messages     # Auflistung aller Prozesse, die /var/adm/messages
geöffnet haben
```

Zombie Prozesse

Es kommt manchmal vor, dass Prozesse, obwohl sie eigentlich schon beendet wurden (z.Bsp. mit dem Kommando *kill*), hartnäckig als sogenannte *Zombies* oder *[defunct]* bestehen bleiben. Wie kommt es dazu, hat das negative Auswirkungen?

Ein Prozess entsteht, indem ein Programm den Systemaufruf *fork()* benützt. Normalerweise ist der Vaterprozess am Exit-Status (Rückgabewert beim Beenden eines Prozesses) des Sohnprozesses interessiert und wartet mittels Systemaufruf *wait()* auf dessen Beendigung. Endet der Sohnprozess durch *exit()* oder durch ein Signal, das er erhält, so wird der Vaterprozess benachrichtigt und alles läuft OK. Hat der Vaterprozess den *wait()* Aufruf nicht durchgeführt (Programmierfehler oder sonst ein Grund) und der Sohnprozess beendet sich, so kann der Exitstatus nicht an den Vaterprozess zurückgemeldet werden, da dieser nicht darauf wartet. Der Kernel gibt alle Ressourcen des Sohnprozesses frei, muss sich aber dessen Exit-Status weiterhin merken solange der Vaterprozess existiert (es könnte ja sein, dass dieser irgendwann doch noch *wait()* aufruft und den Exit-Status benötigt). Der Sohnprozess bleibt solange als Zombieprozess bestehen, bis der Vaterprozess entweder den Exit-Status mittels *wait()* erwartet, oder sich selbst beendet. Nach der Beendigung des Vaterprozesses ist es auch nicht mehr nötig, dass sich der Kernel den Status merkt.

Das heißt, dass man Zombie-Prozesse nur loswird, indem man deren Vaterprozesse beendet. Ist der Vaterprozess für den Betrieb so wichtig, dass er nicht beendet werden darf, so ist es kein Problem mit den Zombies zu leben. Beim nächsten Reboot verschwinden diese. Treten Zombies massenhaft auf, so ist anzunehmen, dass ein Programmierfehler vorliegt.

pgrep und pkill Kommando

Die beiden Kommandos **pkill** und **pgrep** sind eine Kombination aus grep und ps bzw. grep und kill. Es können damit gezielt bestimmte Prozesse mittels Suchmuster gefunden werden.

Syntax:

pgrep [-U *UID*] [-f] [-t *Terminal*] [-l] [-G *GID*] [-n] *Suchmuster*

pkill [-U *UID*] [-f] [-t *Terminal*] [-l] [-G *GID*] [-n] *Suchmuster*

- **-U** Suche auf UserID einschränken
- **-G** Suche auf GroupID einschränken
- **-f** Suche mittels regulärem Ausdruck (unterstützt mächtige, aber u.U. komplizierte Suchmuster)
- **-t** Suche auf Terminal einschränken
- **-l** listet Prozessnamen auf
- **-n** listet den jüngsten Prozess auf

Beispiele:

```
# pgrep -l mail                    # liste alle Prozesse auf, dessen Name mail beinhaltet
399 sendmail
942 dtmail
# pkill dtmail
# pkill -U bevw dtmail # liste Prozesse auf, welche dem User bevw gehören und im Namen dtmail vorkommt
```

Der Prozessmanager

Die grafische Oberfläche (CDE) von Solaris bietet auch eine Kontrolle der Prozesse per *Mausklick*. Dieses Programm befindet sich im Folder *Tools* mit dem Namen *Find Process*. Es kann damit elegant die gewünschte Ausgabe aller Prozesse konfiguriert werden. Außerdem können Prozesse per Maus ausgewählt und beendet werden.

6.2 Zeitgesteuerte Prozesse

at Kommando

Das **at** Kommando dient dazu, Programme zu einem bestimmten Zeitpunkt zu starten.

Syntax:

at [-m] [-r *job*] [-f *programm*] *zeit* [*datum*]

- **-m** Sendet nach Beendigung ein Mail an den User
- **-r** entfernt einen Job aus der Warteschlange, welcher vorher mit *at* eingefügt wurde
- **-f** Programm bzw. Skript, welches ausgeführt werden soll
- **zeit** Uhrzeit zu der Programm ausgeführt werden soll (genaue Syntax siehe Online-Manual bzw. Beispiele unten)
- **datum** Datum, an dem das Programm ausgeführt werden soll

Beispiele:

```
# at 8:45PM                                                    # Eingabe eines Jobs
at>find /export/home/rimmer -name core -exec rm {} \;
at>[Control-D drücken, um Eingabe zu beenden]
commands will be executed using /bin/ksh
job 891550478.a at Thu Apr 2 13:45:28 1999
#

# atq                                                          # Ausgabe der Job-Warteschlange
Rank Execution Date     Owner      Job                  Queue JobName
1st    Apr 2, 1999 13:54   rimmer     891550478.a    a          stdin
#

# at -r 891550478.a                                  # entfernen des Jobs aus der Warteschlange
```

Es kann verhindert werden, dass Benutzer zeitgesteuerte Jobs in die Warteschlange eintragen können. Dies geschieht durch Eintrag der Benutzerkennung in die Datei */etc/cron.d/at.deny*.

Der cron Prozess

cron ist ebenfalls für die zeitgesteuerte Ausführung von Programmen gedacht. Allerdings dient dieses Programm zur wiederholten Programmausführung und nicht - wie **at** - für einmalige Aktionen.

Das Programm eignet sich ideal für nächtliche Sicherungen, Aufräumungsarbeiten (z.Bsp. Logdateien sichern und löschen), Abarbeitung von Batch-Jobs etc.

Die Steuerung von **cron** erfolgt durch die sogenannten *crontab*-Dateien. Diese können für jeden Benutzer eigens konfiguriert werden. Das Bearbeiten dieser Dateien erfolgt mit Hilfe des Kommandos **crontab**. Die Dateien befinden sich im Verzeichnis */var/spool/cron/crontabs/* und tragen den Namen der jeweiligen Benutzerkennung.

Format der crontab - Datei:

Minuten (0-59)	Stunden (0-23)	Tag im Monat (1-31)	Monat (1-12)	Wochentag (0-6), So = 0	Kommando
10	3,21	*	*	1-5	/usr/lib/newsyslog

Die Werte der Felder können in folgenden Formaten angegeben werden:

n Die Programmausführung erfolgt genau zu diesem Zeitpunkt (Minute, Stunde, etc.)

n,p,q Mehrere Zeitpunkte (z.Bsp.: 3,21 — Ausführung um 3 Uhr und um 21 Uhr)

n-p Alle Werte von n bis p (z.Bsp.: 8-17 — Ausführung von 8 bis 17 Uhr)

***** Alle Werte (z.Bsp.: jeden Monat)

Ob Benutzer berechtigt sind, solche cron-Prozesse auszuführen, wird durch die beiden Dateien */etc-/cron.d/cron.allow* und */etc/cron.d/cron.deny* definiert. Existiert die Datei *cron.allow*, so können nur jene Benutzer cron-Jobs ausführen, die darin eingetragen sind. Existiert *cron.allow* nicht, so überprüft **cron**, ob dem Benutzer durch den Eintrag in die Datei *cron.*deny die Ausführung verweigert wird. Existiert keine der beiden Dateien, so ist nur der Superuser *root* berechtigt, cron-Jobs zu starten. Standardmäßig ist es den Benutzern *daemon, bin, smtp, nuucp, listen, nobody* und *noaccess* nicht erlaubt, cron-Jobs zu starten.

Syntax:

crontab [-u *User*] *Datei*

crontab [-u *User*] { -l | -r | -e }

-u Angabe des Users, dessen crontab-Datei bearbeitet werden soll

-l auflisten der crontab-Datei

-e editieren der crontab-Datei

-r löschen der crontab-Datei

Datei Inhalt der Datei wird als crontab-Datei übernommen

Beispiel (crontab von root)

```
1 #ident   "@(#)root       1.14     97/03/31 SMI"    /* SVr4.0 1.1.3.1
      */
2 #
3 # The root crontab should be used to perform accounting data collection.
4 #
```

```
# The rtc command is run to adjust the real time clock if and when
# daylight savings time changes.
#
10 3 * * 0,4 /etc/cron.d/logchecker
10 3 * * 0   /usr/lib/newsyslog
15 3 * * 0 /usr/lib/fs/nfs/nfsfind
1 2 * * * [ -x /usr/sbin/rtc ] && /usr/sbin/rtc -c > /dev/null 2>&1
0 20 * * 1 /opt/SUNWexplo/runexplorer
```

Im obigen Beispiel wird das Programm *logchecker* jeden Sonntag und Donnerstag um 3^{10}h ausgeführt.

Ein häufiger Fehler beim Erstellen von cron-Jobs ist, dass Umgebungsvariablen (PATH, etc.) nicht den erwarteten Wert haben.

6.3 syslog

syslog dient dazu, Meldungen des Systems über den *syslogd* Dämon zu verteilen. Der Systemadministrator hat die Möglichkeit, diese Meldungen durch entsprechende Konfiguration der Datei */etc/syslog.conf* an verschiedene Ziele zu leiten.

Es bestehen folgende Möglichkeiten, Meldungen zu senden:

- Schreiben der Meldung in eine Log-Datei
- Weiterleiten von Meldungen an eine Benutzerkennung
- Ausgabe der Meldungen auf die System-Konsole
- Weiterleiten von Meldungen an den *syslogd*-Dämon eines anderen Rechners im Netz

/etc/syslog.conf - Konfigurationsdatei

Ein Konfigurationseintrag in der Datei */etc/syslog.conf* besteht aus zwei durch Tabulatoren getrennte Felder.[2] Die beiden Felder werden *selector* und *action* genannt. Der *selector* besteht wiederum aus den beiden Teilen *facility* und *level*, welche durch einen Punkt '.' voneinander getrennt sind. *facility* beschreibt die Kategorie der die Meldung zuzuordnen ist, *level* gibt die Wichtigkeit der Meldung an. Das Feld *action* bestimmt das Ziel der Meldung (wohin wird die Meldung geschickt).

Ein Konfigurationseintrag entsprich also folgendem Muster:

facility.level action # **ACHTUNG!** Die beiden Felder müssen durch Tabulatoren voneinander getrennt werden!

Je Zeile können auch mehrere *Selector Fields* angeführt werden, welche dann durch ein Semikolon ';' (keine Leerzeichen oder Tabulatoren!) getrennt werden. Das würde dann so aussehen:

facility.level;facility.level action

Für die einzelnen Felder (facility, level und action) stehen folgene Möglichkeiten zur Verfügung:

Selector Field

Facility:

user Die Meldung wurde durch einen User-Prozess generiert. Das ist die Standardeinstellung für alle Programme, welche in dieser Liste sonst nicht vorkommen.

kern Meldung, die der Kernel erzeugt.

mail Das Mail-System

daemon System Dämonen, wie z.Bsp. *in.fptd* und *telnetd*

[2] Die Trennung der Felder **muss** unbedingt durch Tabulatoren und darf nicht durch Leerzeichen erfolgen! Die Datei wird durch den Makroprozessor *m4* verarbeitet und dieser verlangt zwingend Tabs als Trennzeichen.

auth Das Authorisierungssystem (*login*, *su* und *getty*)

lpr Drucksystem

news Network News System

uucp UNIX to UNIX copy (UUCP) System; (veraltet) verwendet unter Solaris nicht den syslog-Mechanismus

cron *cron* und *at* (*crond*-Dämon)

local[0-7] reserviert für lokale Verwendung

mark Dies ist eine Zeitmarke, die *syslogd* selbst generiert wird. Der Sinn dieser Meldungen ist u.A., dass damit nach einem unbeaufsichtigtem Absturz nachvollzogen werden kann, bis wann das System noch OK war.

***** Alle *facilities* ausgenommen der *mark facility*

Level:

emerg Panic Situationen. Diese werden normalerweise an alle angemeldeten User geschickt.

alert Zustände, die sofort behoben werden sollten, wie z.Bsp. inkonsistente Systemdateien.

crit Kritische Meldungen wie Festplattenfehler.

err andere Fehler

warning Warnungen

notice Meldungen, die keinen Fehler darstellen, aber ev. doch eine spezielle Bedienung erfordern.

info Informationsmeldungen

debug Meldungen von Programmen, welche normalerweise nur dann ausgegeben werden, wenn 'debugging' eingeschaltet ist.

none In Verbindung mit der '*'-Facility, um eine bestimmte Facility aus der Regel auszuklammern. Bsp.: **.debug;mail.none* betrifft alle Meldungen des Levels 'debug' ausgenommen jener, welche vom Mailsystem generiert wurden.

Action Field

/filename Absoluter Pfadname einer Logdatei

@host Hostnamen oder IP-Adressen von Rechnern muss das Zeichen '@' vorangestellt werden. Meldungen werden an den syslogd-Dämon dieses Rechners weitergeleitet.

user1, user2 Die Benutzer 'user1' und 'user2' erhalten eine Meldung, sofern sie angemeldet sind.

***** Alle angemeldeten Benutzer erhalten die Meldung.

syslogd - Dämon

Der Hintergrundprozess *syslogd* wird beim Hochfahren des Systems in Runscriptfile *'/etc/rc2.d/S74-syslog'* gestartet. Der Prozess kann manuell durch den Aufruf dieses Skriptes gestoppt bzw. gestartet werden:

```
# /etc/init.d/syslog start          # Starten von syslogd
# /etc/init.d/syslog stop           # Stoppen von syslogd
```

Nach jeder Änderung in der Konfigurationsdatei /etc/syslog.conf muss der syslogd-Dämon gestoppt und erneut gestartet werden, da die Konfigurationsdatei nur beim Start ausgelesen wird!

Beispiel der Datei /etc/syslog.conf:

```
1  #ident  "@(#)syslog.conf  1.4  96/10/11 SMI"  /* SunOS 5.0 */
2  #
3  # Copyright (c) 1991-1993, by Sun Microsystems, Inc.
4  #
5  # syslog configuration file.
6  #
7  # This file is processed by m4 so be careful to quote (``) names
8  # that match m4 reserved words.  Also, within ifdef's, arguments
9  # containing commas must be quoted.
10 #
11 *.err;kern.notice;auth.notice                    /dev/console
12 *.err;kern.debug;daemon.notice;mail.crit         /var/adm/messages
13
14 *.alert;kern.err;daemon.err                      operator
15 *.alert                                          root
16
17 *.emerg                                          *
18
19 # if a non-loghost machine chooses to have authentication messages
20 # sent to the loghost machine, un-comment out the following line:
21 #auth.notice                 ifdef(`LOGHOST', /var/log/authlog,
       @loghost)
22
23 mail.debug                   ifdef(`LOGHOST', /var/log/syslog,
       @loghost)
24
25 #
26 # non-loghost machines will use the following lines to cause "user"
27 # log messages to be logged locally.
28 #
29 ifdef(`LOGHOST', ,
30 user.err                                         /dev/console
31 user.err                                         /var/adm/messages
32 user.alert                                       `root, operator'
33 user.emerg                                       *
34 )
```

inetd - Meldungen

Der *Internet Services Daemon* (*inetd*) sendet normalerweise keine Meldungen an syslog. Um den Aufruf von Netzwerkfunktionen wie *telnet* oder *ftp* zu protokollieren, kann *inetd* mit der Option '*-t*' gestartet werden. Meldungen von *inetd* erfolgen nur mit den Leveln *daemon* und *notice*! Um den Internet Services Daemon immer mit der *Trace-Option* zu starten, muss der Aufruf von *inetd* in der Datei */etc/init.d/inetsvc* angepasst werden (/usr/sbin/inetd -s **-t**).

Beispiel eines Logeintrages von *inetd*:

```
May 23 15:01:45 sunlicht inetd[220]: 100083[865] from 239.92.239.60 44772
May 23 15:13:02 sunlicht inetd[220]: telnet[3494] from 193.16.214.129 1403
May 23 15:16:25 sunlicht inetd[220]: telnet[3640] from 193.16.214.129 1406
May 23 15:18:31 sunlicht inetd[220]: ftp[3706] from 193.16.214.129 1407
```

logger - Kommando

Mit dem *logger* Kommando können Meldungen an syslog generieren werden.

Syntax:

logger [-i] [-f *file*] [-p *priority*] [-t *tag*] [*message*]

-i Schreibe zusätzlich zur Meldung die Prozess-ID des logger-Kommandos

-f *file* verwende den Inhalt der Datei '*file*' als Meldung

-p *priority* gib der Meldung den Level '*priority*'

-t *tag* Markiere die Meldung mit dem Label '*tag*'

message Meldungstext

Beispiele:

```
# logger System reboot
# logger -p local0.notice -t HOSTIDM -f /var/tmp/mylog
```

Logdateien können laufend mitgelesen werden mit dem Kommando *tail* und der Option '*-f*'. Die Option '*-f*' bewirkt, dass die Datei geöffnet bleibt und jeder neuer Eintrag angezeigt wird.

Beispiel:

```
# tail -f /var/adm/messages
```

7 Benutzer Administration

Die Verwaltung von Benutzerkennungen (User) kann auf Kommandoebene oder über die grafische Oberfläche mittels *admintool* erfolgen.

7.1 Benutzerverwaltung mit admintool

Aufruf von *admintool* aus der Shell:

```
# admintool &
```

Die Bedienung ist selbsterklärend. User können neu angelegt, gelöscht und modifiziert werden.

Wird eine Benutzerkennung neu angelegt, so kopiert *admintool* automatisch die zur jeweiligen Shell gehörigen Initialisierungsdateien (.profile für Bourne- und Kornshell, .login und .cshrc für die C-Shell) aus dem Verzeichnis */etc/skel* ins Homeverzeichnis. Andere Vorlagen, welche sich auch im Verzeichnis */etc/skel* befinden, kopiert *admintool* nicht! Dies ist ein Unterschied zum Kommando *useradd*, welches sonstige Dateien auch kopiert.

7.2 Benutzerverwaltung mit Shell-Kommandos

Die Kommandos, welche nun besprochen werden, verändern die Systemdateien */etc/passwd* (siehe Seite 26), */etc/shadow* (siehe Seite 27) und */etc/group* (siehe Seite 28)

useradd - Erstellen von Benutzerkennungen

Syntax:

useradd [-c *Kommentar*] [-d *Homeverzeichnis*] [-e *Ablaufdatum*]
[-f *Inaktiv-Zeit*] [-g *Primary-Group*] [-G *Secondary-Group*[,...]]
[-m [-k *Vorlagenverzeichnis*]] [-p *Passwort*] [-s *Shell*] [-u UID] *Benutzerkennung*

-c Kommentar (5. Feld der Datei */etc/passwd*)

-d Homeverzeichnis der neuen Benutzerkennung (6. Feld der Datei */etc/passwd*)

-m Existiert das Homeverzeichnis noch nicht, so wird es erzeugt und die Dateien im Verzeichnis */etc/skel* (bzw. im Verzeichnis, welches mit der Option *-k* angegeben wird) werden ins Homeverzeichnis kopiert.

-k Falls anstatt des Standard-Vorlagenverzeichnisses */etc/skel* ein anderes Verzeichnis gewünscht ist, kann dies mit dieser Option angegeben werden.

-e Datum ab wann die Kennung gesperrt wird (8. Feld der Datei */etc/shadow*)

-f Wie lange darf User inaktiv bleiben bevor die Kennung gesperrt wird (7. Feld der Datei */etc/shadow*)

-u Angabe der UID (3. Feld der Datei */etc/passwd*)

-g Angabe der Gruppe deren Mitglied der User werden soll. Es ist sowohl der Name als auch die Gruppennummer (GID) erlaubt. (4. Feld der Datei */etc/passwd*)

-G Angabe der 'secondary group' in die der User zusätzlich zur 'primary group' aufgenommen werden soll. Bei der Angabe von mehreren Gruppen, müssen diese durch Kommas voneinander getrennt werden (ohne Leerzeichen!). (Datei */etc/group*)

-p Verschlüsseltes Passwort (2. Feld der Datei */etc/passwd*). Normalerweise wird das Passwort mit dem Kommando *passwd* vergeben nachdem die Kennung angelegt wurde. Allerdings kann die neue Kennung bis zur Vergabe des Passwortes eine Sicherheitslücke darstellen.

-s Name der Login-Shell (absoluter Pfadname). (7. Feld der Datei */etc/passwd*); kann mit *passwd -e* verändert werden;

userdel - Löschen von Benutzerkennungen

Syntax:

userdel [-r] *Benutzerkennung*

-r Es wird auch das Homeverzeichnis der Benutzerkennung gelöscht. Ohne Angabe dieser Option bleiben die Dateien des gelöschten Users erhalten

usermod - Ändern von Benutzerkennungen

Syntax:

usermod [-c *Kommentar*] [-d *Homeverzeichnis*[-m]]
[-e *Ablaufdatum*] [-f *Inaktive-Zeit*]
[-g *Primary-Group*] [-G *Secondary-Group*[,...]]
[-l *Loginname*] [-p *Passwort*]
[-s *Shell*] [-u UID] *Benutzerkennung*

-c Kommentar (5. Feld der Datei */etc/passwd*)

-d Homeverzeichnis der neuen Benutzerkennung (6. Feld der Datei */etc/passwd*)

-m Existiert das Homeverzeichnis noch nicht, so wird es erzeugt und die Dateien im Verzeichnis */etc/skel* (bzw. im Verzeichnis, welches mit der Option *-k* angegeben wird) werden ins Homeverzeichnis kopiert.

-e Datum ab wann die Kennung gesperrt wird (8. Feld der Datei */etc/shadow*)

-f Wie lange darf User inaktiv bleiben bevor die Kennung gesperrt wird (7. Feld der Datei */etc/shadow*)

-l Neuer Loginname anstatt des derzeitigen Namens der Benutzerkennung

-u Angabe der UID (3. Feld der Datei */etc/passwd*)

-g Angabe der Gruppe deren Mitglied der User werden soll. Es ist sowohl der Name als auch die Gruppennummer (GID) erlaubt. (4. Feld der Datei */etc/passwd*)

-G Angabe der 'secondary group' in die der User zusätzlich zur 'primary group' aufgenommen werden soll. Bei der Angabe von mehreren Gruppen, müssen diese durch Kommas voneinander getrennt werden (ohne Leerzeichen!). (Datei */etc/group*)

-p Verschlüsseltes Passwort (2. Feld der Datei */etc/passwd*). Normalerweise wird das Passwort mit dem Kommando *passwd* vergeben nachdem die Kennung angelegt wurde. Allerdings kann die neue Kennung bis zur Vergabe des Passwortes eine Sicherheitslücke darstellen.

-s Name der Login-Shell (absoluter Pfadname). (7. Feld der Datei */etc/passwd*)

passwd - Ändern von Passwörtern

Syntax:[1]

passwd [-e] [-h] [-f] [-l] [*user*]

-e Ändern der Shell, welche beim Login gestartet wird. Diese wird im Dialog abgefragt.

-h Ändern des Homverzeichnisses

-f Zwingt den User beim nächsten Login zur Passwortänderung

-l Sperren (lock) der Benutzerkennung

Beispiele:

```
pisch@sunlicht:/export/home/pie # > passwd -e user1
Old shell: /bin/sh
New shell: /bin/ksh
pisch@sunlicht:/export/home/pie # > passwd user1
New password:
Re-enter new password:
passwd (SYSTEM): passwd successfully changed for user1
pisch@sunlicht:/export/home/pie # >
```

7.3 Initialisierungsdateien von Benutzerkennungen

Unmittelbar nach der Anmeldung werden Initialisierungen durchgeführt. Abhängig von der für die jeweilige Benutzerkennung gewählten Shell wird dies durch unterschiedliche Skript-Dateien durchgeführt. Es gibt Dateien, welche systemweit für alle Benutzerkennungen gelten und individuelle Dateien, welche durch den Benutzer selbst verändert werden können (falls nicht vom Administrator bewusst verhindert).

Shell	System	User	Vorlage in /etc/skel
Bourne (sh)	/etc/profile	$HOME/.profile	local.profile
Korn (ksh)	/etc/profile	$HOME/.profile	local.profile
		$HOME/.kshrc (*)	
C (csh)	/etc/.login	$HOME/.cshrc	local.cshrc
		$HOME/.login	local.login

(*) Die Datei */etc/.kshrc* wird nur dann durchlaufen, wenn die Umgebungsvariable *ENV=$HOME-/.kshrc* gesetzt und exportiert wurde! Anstatt des Dateinamens *.kshrc* kann auch ein anderer Name angegeben werden, dies ist jedoch ein allgemein üblicher Name und sollte deshalb auch verwendet werden.

Werden Benutzer über das Kommando *useradd* angelegt, so werden auch andere Dateien aus dem Vorlagenverzeichnis */etc/skel* ins Homeverzeichnis übernommen. Bei Verwendung von *admintool* werden nur die in der letzten Spalte oben angeführten Dateien übernommen und umbenannt (das Prefix *local* wird gelöscht).

Für Benutzerkennungen, welche mit *CDE* (grafische Oberfläche) arbeiten, gibt es die Initialisierungsdatei *$HOME/.dtprofile*. Darin werden individuelle Einstellungen zu CDE gespeichert. Wird ein Terminalfenster geöffnet und Kornshell ist die User-Shell, so wird automatisch die Datei *.kshrc* durchlaufen. Beim Öffnen eines Konsolefensters wird zuerst *.profile* und dann *.kshrc* durchlaufen. Bei Solarisversionen < V2.6 musste in der Datei .dtprofile die Umgebungsvariable *DTSOURCEPROFILE=true* gesetzt werden um obigen Effekt zu erzielen.

Insbesondere die beiden Umgebungsvariablen *PATH* und *MANPATH* müssen gegebenenfalls angepasst werden. *PATH* enthält die Verzeichnisse, in denen ein Kommando, welches nicht mit absolutem Pfadnamen aufgerufen wurde, gesucht wird. *MANPATH* enthält die Verzeichnisse, in denen sich Manualseiten befinden. Beim Einsatz von Fremdsoftware werden Manualseiten z.Bsp. unter */usr/local/man* installiert. Damit diese Manualseiten mit dem Kommando *man* gefunden werden, muss *MANPATH* erweitert werden:

```
# MANPATH=$MANPATH:/usr/local/man; export MANPATH
```

7.4 Shell Features

Die drei am häufigsten (und praktisch auf allen UNIX-Derivaten verfügbaren) Shells haben unterschiedliche Eigenschaften. Die Bourne-Shell bietet am wenigsten Komfort, ist dafür aber klein und extrem stabil. Die C-Shell ähnelt der Programmiersprache C, unterstützt Funktionen wie Jobcontrol und History. Die C-Shell ist **die** Shell für die SAP/R3 Administration. Die Korn-Shell entstand später, ist zur Bourne-Shell kompatibel und besitzt weitgehend alle Vorteile der C-Shell auch. Folgende Tabelle listet einige Features der 3 Shells auf:

Feature	Bourne	C	Korn
Aliase	-	ja	ja
Kommando Editierfunktion	-	eingeschränkt	ja
History Liste	-	ja	ja
^D ignorieren (ignoreeof)	-	ja	ja
unabhängige Initialisierungsdatei (von .profile)	-	ja	ja
Job Control (Möglichkeit Prozesse in den Hintergrund bzw. wieder in den Vordergrund zu versetzen)	-	ja	ja
Logout Datei	-	ja	-
Dateien vor dem Überschreiben schützen (noclobber)	-	ja	ja

7.5 Administration und Konfiguration des CDE

7.5.1 Login Manager

Der *Login Manager* ist für die Anzeige des Anmeldebildschirmes, die Authentifizierung und das Starten einer Benutzer Session verantwortlich. Nach dem Abmelden des Benutzers wird wieder ein Anmeldefenster präsentiert.

Es können folgende Eigenschaften beeinflusst werden:

- Darstellungsform des Login-Fensters
- Änderung der Standardsprache
- Ausführung von Kommandos vor dem Start einer Benutzer Session

Alle diese Einstellungen können entweder für alle oder für einzelne 'Displays' vorgenommen werden.

Um Änderungen durchzuführen, muss zunächst die Datei */usr/dt/config/C/Xresources* nach */etc/dt-/config/C* kopiert werden. Die Änderungen werden beim nächsten Login bzw. nach Auswahl des Menüpunktes '*Options –> Reset Login Screen*' wirksam.

7.5.1.1 Änderung des Logo

Standardmäßig wird am Loginfenster das *SUN Microsystems* Logo angezeigt. Um das zu ändern, muss in der Datei *Xresources* der Eintrag *Dtlogin*logo*bitmapFile* angepasst werden.

1. `# cd /etc/dt/config/C`
   `# cp /usr/dt/config/C/Xresources .`
2. Editieren der Datei */etc/dt/config/C/Xresources*:
   `Dtlogin*logo*bitmapFile:/Pfad_der_gewünschten_Bitmap_Datei`
3. Speichern der Datei
4. Abmelden –> Das neue Logo erscheint am Bildschirm

7.5.1.2 Ändern der 'Welcome Message'

Ohne Änderung wird die Meldung „Welcome to *hostname*“ angezeigt. Eine Änderung wird durch Anpassung der Zeile
`Dtlogin*greeting.labelString: der gewünschte Text`
bewirkt. Im Text können noch folgende Schlüsselwörter verwendet werden:

%LocalHost% Wird durch den Hostnamen des Rechners ersetzt.

%DisplayName% wird durch den Namen des Xserver Display ersetzt.

Nach Eingabe des Benutzernamens erscheint die Meldung „Welcome *username*“. Das kann durch den Eintrag
`Dtlogin*greeting.persLabelString: Welcome %s`
erreicht. Anstatt '*%s*' wird der Benutzername eingefügt.

7.5.1.3 Umgebungs Variablen

Wenn sich ein Benutzer über CDE anmeldet, so werden die Dateien *.profile* und *.login* NICHT automatisch ausgeführt. Das ist, weil der Xserver läuft, bevor der User angemeldet ist. Damit diese Skripte durchlaufen werden, muss die Umgebungsvariable *DTSOURCEPROFILE* auf *true* gesetzt werden. Diese Variable befindet sich in der Datei *$HOME/.dtprofile*.

Umgebungsvariablen können an folgenden Stellen belegt werden:

- *$HOME/.dtprofile*
- Standardvariablen des Login Manager
- Systemweite Umgebungsvariablen in den Konfigurationsdateien des Login Manager
- Ressourcen für Zeitzone und Sprache
- benutzerspezifische Variablen, welche in den Initialisierungsdateien der Shells gesetzt werden

Umgebungsvariablen für CDE finden sich in mehreren Dateien. Wirksam ist jener Eintrag, welcher zuerst gefunden wird. Die Reihenfolge ist:

1. *$HOME/.dt/config* (existiert nicht automatisch)
2. */etc/dt/config* (existiert nicht automatisch)
3. */usr/dt/config* (existiert immer und enthält Standard)

7.5.1.4 Starten und Stoppen des Login Managers

Der Login Manager kann durch das Runscript */etc/rc2.d/S99dtlogin* gestartet (Option *start)* bzw. gestoppt (Option *stop)* werden.

Das automatische Starten des CDE kann beeinflusst werden:

/usr/dt/bin/dtconfig -d # Disable automatic startup of CDE

/usr/dt/bin/dtconfig -e # Enable automatic startup of CDE

Dabei wird das Runscript */etc/rc2.d/S99dtlogin* entweder entfernt bzw. hinzugefügt.

Fehlermeldungen des Login Managers Treten Fehler auf, so protokolliert der Login Manager diese in der Datei */var/dt/Xerrors*.

Damit Änderungen von Konfigurationsdateien des Login Managers wirksam werden, muss entweder ein Logoff durchgeführt werden, oder (nur unter der Kennung *root* möglich) folgende Kommandos ausgeführt werden:

- `# /usr/dt/bin/dtconfig -reset`

oder

- Feststellen der PID des *dtlogin* Dämons und senden des Signales *HUP* an diesen Prozess:
  `# cat /var/dt/Xpid`
  `# kill -HUP PID`

7.5.2 Der Session Manager

Eine *session* ist eine Zusammenstellung von Applikationen, Einstellungen und Ressourcen, welche dem Benutzer am Desktop zur Verfügung stehen.

Um Einstellungen speichern und wiederherstellen zu können, verwendet der Session Manager das *Inter-Client Communications Convention Manual (ICCCM) Session Management Protocol*. *ICCCM* regelt die Kommunikation zwischen den Clients (aus Sicht des XServers), wie z.Bsp. Bereichsauswahl, Window Management, Farbauswahl, ...

Der *Session Manager* speichert und stellt wieder her:

- Laufende Applikationen
- Einstellungen von Applikationsfenstern wie Farbe, Fonts, Fenstergröße
- Einstellungen des Xserver wie Verhalten der Maus, Audio Lautstärke, Tastatur Klick

Der *Session Manager* ermöglicht:

- Ablauf eines individuell Shellscripts nach dem Login
- Änderung des Window Managers
- Änderung der *fail-safe session*

7.5.2.1 Session Typen

Initial Session Wenn der Benutzer sich das erste Mal anmeldet, generiert der Session Manager einen Standard-Desktop.

Current Session Das ist jene Session, welche beim Logoff abgespeicher wird und nach erneutem Login wieder erscheint.

Home Session Das ist eine Session, welche der Benutzer abspeichern kann, um zu einem definierten, bekannten Desktop zu gelangen.

7.5.2.2 Die erste Session

Wenn sich ein Benutzer erstmalig anmeldet, so liest der Session Manager die Datei */usr/dt/config/C-/sys.resources*. Hier befinden sich die Standardeinstellungen.

Beim Abmelden wird die *Current Session* im Verzeichnis *$HOME/.dt/sessions/current* abgespeichert.

Erstellen einer 'ersten Session' Das 'customizen' einer Session kann beinhalten:

- Verteilung einer 'ersten Session' an andere Systeme
- Ablauf eines Skriptes
- Starten eines anderen Window Managers
- Erstellen von Display abhängigen Sessions

Folgende Schritte sind durchzuführen:

1. Login durchführen
2. gewünschte Applikationen starten und Einstellungen durchführen
3. Logout – dabei wird das Verzeichnis *$HOME/.dt/sessions/current* erstellt
4. Obiges Verzeichnis auf das System kopieren, wo es gewünscht wird.
5. Entfernen von Display spezifischen Session Verzeichnissen

7.5.2.3 Session Manager Konfigurationsdateien

Die Konfigurationsdateien eines Benutzers befinden sich in den Verzeichnissen *$HOME/.dt/sessions-/current* und *$HOME/.dt/sessions/home*. *current* enthält die aktuelle Einstellung, *home* die als *home-session* abgespeicherte Einstellung.

dt.session Die Datei *dt.session* enthält Namen des aktiven Fensters, Fenstergeometrie, Status (minimized, maximized,...) etc.

dt.resources *dt.resources* enthält Farben der Fenster, des Hintergrundes, der Sprache etc.

dt.settings *dt.settings* enthält Einstellungen des Session Managers für Dinge wie *shutdown state* und *shutdown mode*.

7.5.2.4 Start Up Applikationen

Um Programme gleich nach dem Login automatisch zu starten, können Kommandos in die Datei *sessionetc* gestellt werden.

Für eine systemweite Einstellung kann das Skript */usr/dt/config/sessionetc* nach */etc/dt/config/session-etc* kopiert und dort modifiziert werden.

Soll nur ein bestimmter User davon betroffen sein, dann ist diese Datei nach *$HOME/.dt/sessions/ses-sionetc* zu kopieren.

7.5.3 Workspace Manager

Der *Workspace Manager* ist der von *CDE* verwendete Window Manager. Wie auch andere Window Manager kontrolliert er:

- Die Darstellung von Komponenten von Fensterrahmen
- Das Verhalten von Fenstern (Fokus, Überlappung, ...)
- Tastenzuordnung
- Darstellung von minimierten Fenstern (Icons)
- Menüs von Desktop und Fenstern
- Anzahl von *Workspaces*[2]
- Hintergrund des *Workspace*. Der Anwender kann den Hintergrund mit Hilfe des *CDE Style Manager* verändert werden.
- *Front Panel* – Das Front Panel besitzt eigene Konfigurationsdateien, welche vom Workspace Manager verwaltet werden.

Konfigurationsdateien des Workspace Manager Der Workspace Manager sucht in den folgenden Dateien nach Konfigurationsdateien, wobei *$LANG* die Sprachumgebung des Anwenders ist:

- persönliche Konfigurationsdateien
  - *$HOME/.dt/$LANG/dtwmrc*
  - *$HOME/.dt/$LANG/wsmenu.dtwmrc*
- systemweite Konfigurationsdateien
  - */etc/dt/config/$LANG/sys.dtwmrc*
  - */etc/dt/config/$LANG/wsmenu*
- Workspace Manager interne Konfigurationsdateien (*Built-in*)
  - */usr/dt/config/$LANG/sys.dtwmrc*
  - */usr/dt/config/$LANG/wsmenu*

Die erste Datei, die gefunden wird, wird verwendet. Wird keine Datei gefunden, so werden die Standardwerte (*Built-in*) des Workspace Manager verwendet.

[2] *Workspace* ist der unter Windows bekannte *Desktop*. Die meisten Window Manager unterstützen mehrere Workspaces.

Workspace Manager Menüs Der Workspace Manager hat 3 Standardmenüs:

- *Workspace Menü* – auch *Root Menu* genannt. Es erscheint, wenn die rechte Maustaste (Taste 3) über dem Hintergrund gedrückt wird.
- *Window Menü* – Diese Menü erscheint, wenn in einem Fenster die linke oder rechte Maustaste (Tasten 1 oder 3) gedrückt wird.
- *Front Panel Menü* – Das Menü wird dargestellt, wenn die rechte Maustaste (Taste 3) über dem Front Panel Menü Knopf betätigt wird.

Erzeugen eines User Workspace Menüs Es soll das Workspace Menü, welches Anwender bei ihrer 'ersten Session' erhalten, erstellt werden. Bei der ersten Verwendung von CDE wird das Verzeichnis */usr/dt/config/C/wsmenu* nach *$HOME/.dt/wsmenu* kopiert.

Im folgenden Beispiel wird dem Tools Menü das Programm *Spell* hinzugefügt.

1. Login als *root*
2. `# cd /usr/dt/config/C/wsmenu/Tools`
3. `# ln -s /usr/dt/appconfig/appmanager/C/Desktop_Tools/Spell Spell`

Benutzer können Menüs hinzufügen, indem ihre Datei *$HOME/.dt/wsmenu* modifiziert wird und anschließend durch den Workspace Menüpunkt '*Windows –> Update Workspace Menu*' aktiviert wird.

7.5.4 Front Panel

Die Standardkonfiguration des Front Panel liegt in den Dateien */usr/dt/appconfig/types/C/*.fp*. Wie auch bei den anderen Konfigurationsdateien von CDE, gibt es zusätzlich noch benutzerspezifische und systemweit gültige Dateien. Die Reihenfolge in der gesucht wird ist folgende (die zuerst gefundenen Werte sind gültig):

1. *$HOME/.dt/types/fp_dynamic/*.fp*
2. */etc/dt/appconfig/types/C/*.fp*
3. */usr/dt/appconfig/types/C/*.fp*

Im Falle von Namenskonflikten in den Konfigurationsdateien gelten folgende Regeln:

- Haben Einträge denselben Namen, wird der erste gefundene Eintrag verwendet.
- Beziehen sich 2 Einträge auf dieselbe Position, werden sie in der Reihenfolge des Einlesens plaziert.

Um solche Konflikte zu verhindern, folgende Tipps:

1. Handelt es sich um ein *built-in control*, so kopiert man den entsprechenden Eintrag von der Datei */usr/dt/appconfig/types/C/dtwm.fp* nach */etc/dt/appconfig/types/C/load.fp*.

2. In der Datei */etc/dt/appconfig/types/C/load.fp* muss nun bei der Definition *PANEL FrontPanel:* die Zeile
   LOCKED True
   hinzugefügt werden.

8 Festplatten Verwaltung

8.1 Device Namen

Der *Physical Device Name* wird beim erstmaligen Systemstart bzw. bei einer Rekonfiguration (im OBP: ok **boot r**) generiert. Diese Namen können im OBP mittels Kommando *show-devs* angezeigt werden. Im laufenden System stehen die physikalischen Namen im Verzeichnis */devices*.

Custom Device Aliase sind vereinfachte physikalische Namen, welche im OBP mit dem Kommando *nvalias* erzeugt, mit *devalias* aufgelistet und mit *nvunalias* gelöscht werden können.

Logical Device Names sind die Namen der Gerätedevices unter dem Verzeichnis */dev*. Diese sind symbolische Links auf *physical names* unter dem Verzeichnis */devices*.

*Instance Name*s sind Kurznamen (’*abbreviations*’) für *physical names*.

Eine Rekonfiguration des Geräte im System kann auf zweierlei Wege veranlasst werden:

- im OBP durch Boot mit der Option ’r’
  ok **boot r**
- oder im laufenden System die Datei */reconfigure* erzeugen und einen Reboot durchführen.
  # touch /reconfigure
  # shutdown -i6 -g0 -y

dmesg Kommando

dmesg zeigt Diagnoseinformationen und listet *physical device names* und *instance names* auf. Da *dmesg* aus einem Puffer liest, der u.U. durch Meldungen überschrieben wird, erhält man eine ’saubere’ Ausgabe nur nach einem Reboot.

```
# dmesg
Apr 26 15:25
cpu0: SUNW,UltraSPARC-IIi (upaid 0 impl 0x12 ver 0x90 clock 360 MHz)
SunOS Release 5.6 Version Generic_105181-16 [UNIX(R) System V Release 4.0]
Copyright (c) 1983-1997, Sun Microsystems, Inc.
mem = 131072K (0x8000000)
avail mem = 125861888
Ethernet address = 8:0:20:b1:24:a7
root nexus = Sun Ultra 5/10 UPA/PCI (UltraSPARC-IIi 360MHz)
pci0 at root: UPA 0x1f 0x0
pci0 is /pci@1f,0
PCI-device: pci@1,1, simba #0
PCI-device: pci@1, simba #1
```

```
dad0 at pci1095,6460 target 0 lun 0
dad0 is /pci@1f,0/pci@1,1/ide@3/dad@0,0
<QUANTUM FIREBALL ST3.2A cyl 6254 alt 2 hd 16 sec 63>
root on /pci@1f,0/pci@1,1/ide@3/disk@0,0:a fstype ufs
glm0:   Rev. 5 Symbios 53c875 found.
glm0 is /pci@1f,0/pci@1/scsi@1
glm1:   Rev. 5 Symbios 53c875 found.
glm1 is /pci@1f,0/pci@1/scsi@1,1
glm2:   Rev. 5 Symbios 53c875 found.
glm2 is /pci@1f,0/pci@1/scsi@3
glm3:   Rev. 5 Symbios 53c875 found.
glm3 is /pci@1f,0/pci@1/scsi@3,1
su0 at ebus0: offset 14,3083f8
su0 is /pci@1f,0/pci@1,1/ebus@1/su@14,3083f8
su1 at ebus0: offset 14,3062f8
su1 is /pci@1f,0/pci@1,1/ebus@1/su@14,3062f8
keyboard is </pci@1f,0/pci@1,1/ebus@1/su@14,3083f8> major <37> minor <0>
mouse is </pci@1f,0/pci@1,1/ebus@1/su@14,3062f8> major <37> minor <1>
stdin is </pci@1f,0/pci@1,1/ebus@1/su@14,3083f8> major <37> minor <0>
SUNW,m64B0 is /pci@1f,0/pci@1,1/SUNW,m64B@2
m64#0: 1152x900, 2M mappable, rev 4750.7c
stdout is </pci@1f,0/pci@1,1/SUNW,m64B@2> major <35> minor <0>
se0 at ebus0: offset 14,400000
se0 is /pci@1f,0/pci@1,1/ebus@1/se@14,400000
SUNW,hme0: CheerIO 2.0 (Rev Id = c1) Found
SUNW,hme0 is /pci@1f,0/pci@1,1/network@1,1
SUNW,hme0: Using Internal Transceiver
SUNW,hme0: 10 Mbps half-duplex Link Up
dump on /dev/dsk/c0t0d0s1 size 131512K
```

prtconf Kommando

prtconf zeigt die Systemkonfiguration mit *instance names* an.

```
# prtconf | grep -v "not attached"
System Configuration:  Sun Microsystems  sun4u
Memory size: 128 Megabytes
System Peripherals (Software Nodes):

SUNW,Ultra-5_10
options, instance #0
pci, instance #0
pci, instance #0
ebus, instance #0
power, instance #0
se, instance #0
su, instance #0
su, instance #1
fdthree, instance #0
network, instance #0
SUNW,m64B, instance #0
ide, instance #0
dad, instance #0
atapicd, instance #2
pci, instance #1
scsi, instance #0
scsi, instance #1
```

```
scsi, instance #2
scsi, instance #3
st, instance #21
pseudo, instance #0
```

8.2 Partitionierung der Festplatte

Die Partitionierung der Platten wird während der Solaris-Installation durch *installtool* durchgeführt. Wird später eine Platte hinzugefügt, so muss diese partitioniert werden.

Zuvor kann es erforderlich sein, die benötigten Geräteknoten einzurichten. Das ist im laufenden System mit dem Kommando *disks* möglich.

disks # erzeugt Links für Platten in /dev/dsk/ auf Geräteknoten

format

format wird von der Shell aus aufgerufen.

```
# > format
Searching for disks...done

AVAILABLE DISK SELECTIONS:
0. c0t0d0 <QUANTUM FIREBALL ST3.2A cyl 6254 alt 2 hd 16 sec 63>  rootdisk
/pci@1f,0/pci@1,1/ide@3/dad@0,0
Specify disk (enter its number): 0
selecting c0t0d0: rootdisk
No defect list found
[disk formatted, no defect list found]
Warning: Current Disk has mounted partitions.

FORMAT MENU:
disk       - select a disk
type       - select (define) a disk type
partition  - select (define) a partition table
current    - describe the current disk
format     - format and analyze the disk
repair     - repair a defective sector
show       - translate a disk address
label      - write label to the disk
analyze    - surface analysis
defect     - defect list management
backup     - search for backup labels
verify     - read and display labels
save       - save new disk/partition definitions
volname    - set 8-character volume name
!<cmd>     - execute <cmd>, then return
quit
format>
```

Die Standardgrößen der Partitionen stehen in der Datei */etc/format.dat*.

Um eine Platte in Solaris ansprechen zu können, muss das *Disk Label* oder auch *VTOC (volume table of contents)* genannt, auf die Platte geschrieben werden (mit *format*). Die VTOC belegt den ersten Sektor der Platte[1]. Die VTOC enthält unter anderem folgende Informationen:

- Name der Platte (Volume Name) - optional
- Sektorgröße in Bytes
- Anzahl der Partitionen
- Partition Tabellen
- Partition-Flags, welche markieren, ob Parition beschreibbar bzw. mountbar ist

Bei der Einteilung der Partitionen mittels *format* ist darauf zu achten, dass keine Lücken (Platzvergeudung) bzw. Überlappungen (Datenverlust) zwischen den Partitionen entstehen. Ist die Partitionierung fertiggestellt und auf Platte geschrieben, kann diese auch als neue Standardpartitionierung in die Datei */etc/format.dat* geschrieben werden (Menüpunkt *save*).

ACHTUNG: Bei Solaris ist die gesamte Platte in Slice 2 (nicht wie unter Reliant Unix üblich, Slice 7) konfiguriert.

prtvtoc Kommando

Mit **prtvtoc** lassen sich die aktuellen Partitioninformationen einer Platte auslesen.

```
# prtvtoc /dev/rdsk/c0t0d0s2
* /dev/rdsk/c0t0d0s2 (volume "rootdisk") partition map
*
* Dimensions:
*     512 bytes/sector
*      63 sectors/track
*      16 tracks/cylinder
*    1008 sectors/cylinder
*    6256 cylinders
*    6254 accessible cylinders
*
* Flags:
*   1: unmountable
*  10: read-only
*
* Unallocated space:
*       First     Sector    Last
*       Sector     Count    Sector
*     6303024      1008   6304031
*
*                          First     Sector    Last
* Partition  Tag  Flags    Sector     Count    Sector  Mount Directory
0      2    00          0   6029856   6029855   /
1      3    01    6029856    263088   6292943
```

[1] Die VTOC sieht nicht bei allen UNIX-Sysstemen gleich aus. Es gibt VTOC's welche 8 Partitionen unterstützen, andere unterstützen 16 Partitionen (Solaris 2.7 / SPARC). Detaillierte Informationen dazu können den Include-Dateien im Verzeichnis */usr/include/sys/* entnommen werden.

```
2       5    00          0   6304032   6304031
6       0    00    6292944      5040   6297983
7       0    00    6297984      5040   6303023
```

Muss exakt dieselbe Partitionierung von einer Platte auf eine andere übernommen werden (z.Bsp. für eine Plattenspiegelung), so kann dies mit den folgenden Kommandos erfolgen. Es soll die Partitionierung der Platte /dev/dsk/c0t0d0s2 auf die Platte /dev/dsk/c0t8d0s2 kopiert werden:

```
# prtvtoc -s /dev/rdsk/c0t0d0s2 | fmthard -s - /dev/rdsk/c0t8d0s2
```

8.3 Solaris Platten-Dateisysteme

Für den Benutzer ist ein Dateisystem eine Ansammlung von Dateien und Verzeichnissen worin Informationen gespeichert ist.

Aus der Sicht des Betriebssystemes ist ein Dateisystem eine Zusammensetzung von Kontrollstrukturen und Datenblöcken.

Solaris unterstützt unterschiedliche Dateisysteme:

Platten-Dateisysteme:

ufs Unix File System – Standard Dateisystem von Solaris auf den Platten

hsfs High Sierra File System – CD-ROM Dateisystem, definiert in der Norm ISO9660

pcfs PC File System – ermöglicht das Lesen und Schreiben von DOS-formatierten Platten und Disketten

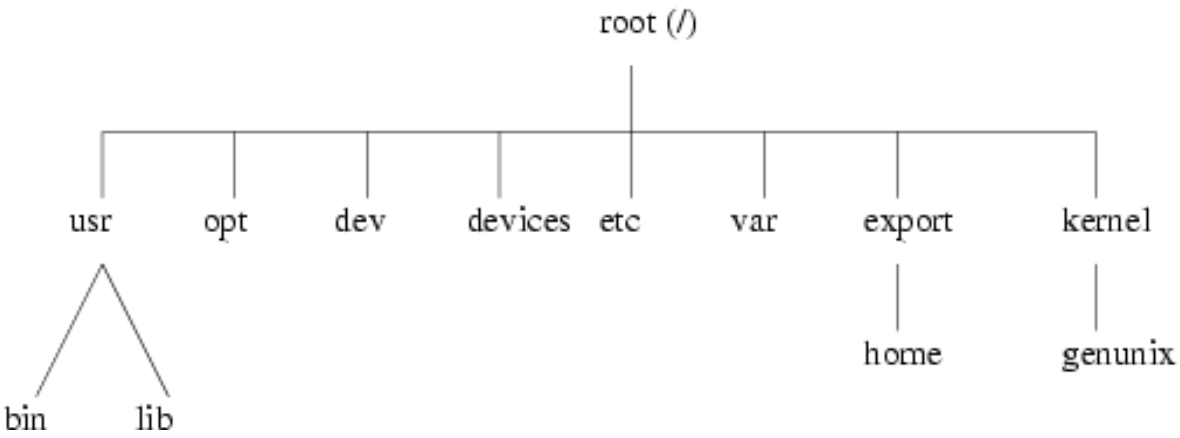
Netzwerk-Dateisysteme:

nfs Network File System – ermöglicht den Zugriff auf Dateisysteme über das Netzwerk (siehe Kapitel 12, Seite 123)

RAM-basierte Dateisysteme:

proc, fd das sind Pseudodateisystem wie z.Bsp. */proc* und */dev/fd*. Die Daten befinden sich nur im RAM, werden aber wie ein Dateisystem auf Platte dargestellt (Siehe Kapitel 14, Seite 136.).

8.3.1 Dateisystem-Struktur



usr unix system resources - Programme und Libraries

opt third-party software

dev links auf Gerätedevices in /devices

devices Deviceeinträge für Gerätetreiber

etc Administrations-Dateien

var variable Daten (Spool, Mail,...)

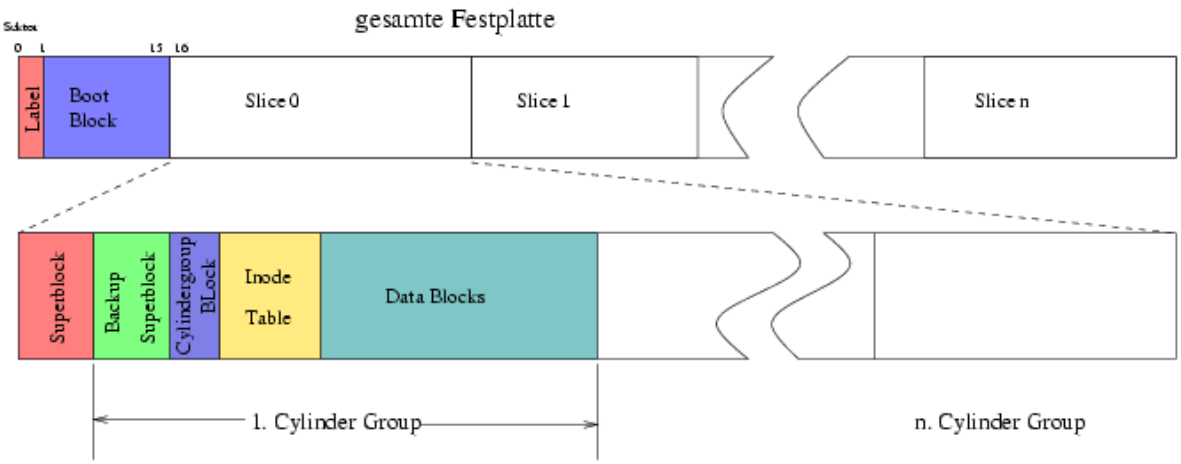
export Verzeichnisse welche per NFS freigegeben werden, sowie home-Verzeichnisse der User

kernel enhält Kernel und Kernelmodule

8.3.2 Das ufs - Dateisystem

Wenn eine Platte mit *format* partitioniert wurde, so wurden damit zwar logische Bereiche definiert, es befindet sich aber noch keine Struktur auf der Platte, welche Informationen darauf organisiert. Diese Partitionen werden *raw partitions* genannt. Solche *raw partitions* werden z.Bsp. von Datenbank-Serverprogrammen verwendet. Solche Programme verwalten diesen Plattenbereich selbst. Das Programm selbst ist dafür verantwortlich, die Daten zu organisieren und den Zugriff darauf zu kontrollieren. Das erhöht die Performance, da die Struktur exakt für diesen Anwendungszweck optimiert werden kann.

Um Dateien und Verzeichnisse auf einer Partition speichern zu können, wird eine Struktur – das Dateisystem – auf den Partitionbereich aufgebracht, die der Betriebssystemkernel verwaltet.



Jedes ufs-Dateisystem wird aus Performancegründen in mehrere *Zylindergruppen* unterteilt. Standardmäßig werden 16 Plattenzylinder zu einer Gruppe zusammengefasst. Dadurch wird erreicht, dass Dateien in einem schmalen Bereich auf der Platte abgelegt werden, um die nötigen Positionierbewegungen der Schreib-Leseköpfe zu minimieren.

Superblock

In jedem Dateisystem befindet sich ein *Superblock*, der aufgrund der Wichtigkeit in jeder Zylindergruppe als Kopie (*Backup Superblock)* existiert. Dieser beinhaltet folgende Informationen:

- Anzahl der Datenblöcke
- Anzahl der Zylindergruppen
- Größe der Datenblöcke und Fragmente
- Hardware-Daten (dienen der Optimierung der Zugriffe)
- Name des Mount-Punktes
- Dateisystem-Status (active, clean, stable)

Zylindergruppen

Das Betriebssystem ist ständig darum bemüht, Dateien innerhalb weniger Zylinder zusammenzufassen um die Performance zu erhöhen. Jede Zylindergruppe beinhaltet folgende Informationsblöcke:

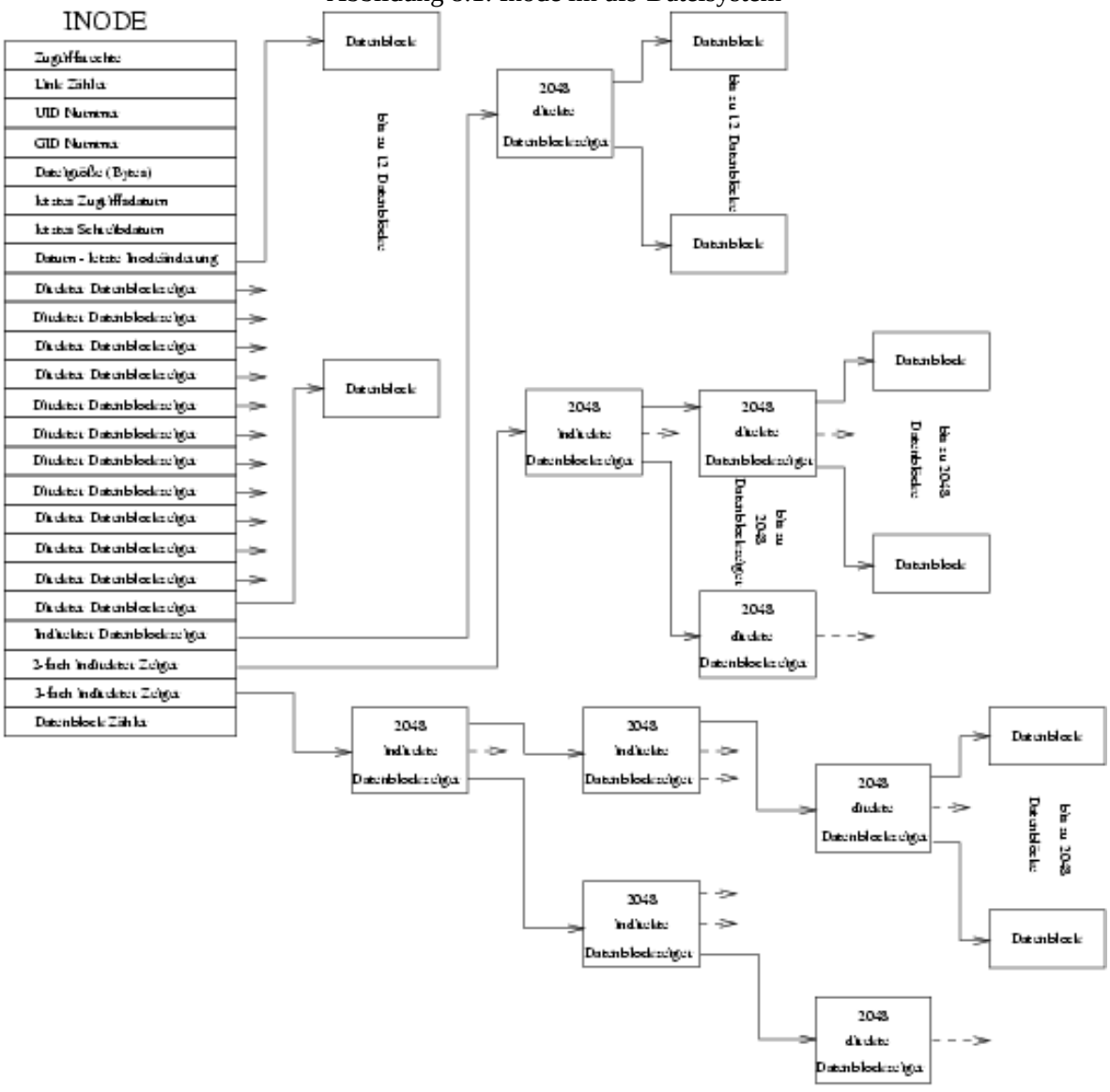
- Cylinder Group Block – darin stehen:
  - Anzahl der Inodes
  - Anzahl von Datenblöcken in dieser Zylindergruppe
  - Anzahl von Verzeichnissen
  - Freie Blöcke, freie Inodes und freie Fragmente
  - Freie Block-Map
  - Freie Inode-Map
- Inode Table – diese Tabelle enthält Informationen zu den Dateien und Verzeichnissen (Details weiter unten)
- Data Blocks – darin befinden sich die eigentlichen Daten; die Blockgröße beträgt standardmäßig 8192 Bytes. Diese können noch in *Fragmente* von 1024 Bytes unterteilt werden, um beim Speichern von kleinen Dateien nicht unnötig Festplattenplatz zu vergeuden.

Inodes

Was unter DOS in der FAT steht, ist unter Unix in den Inode-Tables zu finden. Hier sind Informationen zu Dateien, Verzeichnissen, Links, Geräteknoten etc. zu finden. Das sind (Details finden sich in den Include-Dateien unter dem Verzeichnis */usr/include/sys/fs*):

- Zugriffsrechte
- Dateityp (echte Datei, Verzeichnis, Hardlink, symbolischer Link, Blockdevice, Characterdevice, . . .)
- Anzahl der Hardlinks auf diese Inode
- UID (Eigentümer Identifier)

Abbildung 8.1: Inode im ufs-Dateisystem



- GID (Gruppen Identifier)
- Größe in Bytes
- Datum des letzten Zugriffes
- Datum der letzten Änderung
- Datum der letzten Inode-Änderung
- Zeiger auf Datenblöcke
- Datenblockzähler

Mit Hilfe der mehrfach indizierten Zeiger auf Datenblöcke wäre es möglich Dateien bis zu 70 Terabyte Größe zu erzeugen. Dies wird jedoch im ufs-Dateisystem auf 1 Terabyte begrenzt. Es ist auch nicht empfehlenswert, sehr große Dateisystem als ufs-Dateisystem anzulegen, da ein Filesystemcheck dann bis zu Stunden dauern kann. Dafür eignet sich besser das Veritas Dateisystem (dies ist bei Solaris im Gegensatz zu Reliant Unix optional).

Verzeichnis (Directory)

Es fällt auf, dass der Dateiname noch in keiner der bisher besprochenen Strukturen vorgekommen ist. Der Dateiname wird nicht in der Inode abgelegt! Wir erinnern uns: Um den Namen einer Datei ändern zu können, benötigen wir Schreibrecht auf jenes Verzeichnis in dem die Datei sich befindet. Der Name befindet sich also im Verzeichnis.

Ein Verzeichnis belegt, gleich wie Dateien auch, Datenblöcke. Sie unterscheiden sich zu den normalen Dateien nur durch den Dateityp in der Inodestruktur. Die Daten im Verzeichnis sind die Namen der Dateien und deren zugehörige Inodenummer.

Beim Löschen einer Datei, wird im Verzeichnis an der Stelle des Namens dieser Datei lediglich eine Markierung vorgenommen. Der Dateiname steht weiterhin im Verzeichnis, bis irgendwann dieser Platz für einen neuen Dateinamen benötigt wird. Das hat zur Folge, dass die Größe von Verzeichnissen niemals schrumpft. Hat man einmal mehrere hundert Dateien in einem Verzeichnis angelegt und diese später wieder gelöscht, so werden die Datenblöcke, die der Verzeichniseintrag belegt, nicht wieder freigegeben. Diese Datenblöcke werden erst beim Löschen des Verzeichnisses frei. Das ist vielleicht nicht sehr schön, wenn man aber bedenkt, dass ein Datenblock ohnehin 8 KBytes beinhaltet (also Platz für mehrere hundert Dateinamen von üblicher Länge bietet), so ist verständlich, dass die Platzverschwendung gerne zugunsten des Performancegewinnes in Kauf genommen wird.

Existiert ein *Hardlink* auf eine Datei, so gibt es also einfach mehr als nur einen Verzeichniseintrag, der auf dieselbe Inode verweist. Das erklärt auch, warum Hardlinks nur innerhalb eines Dateisystemes möglich sind: Die Inodenummern sind nur innerhalb eines Dateisystemes eindeutig. Bei jedem Erzeugen eines Verweises auf eine Inodenummer in einem Verzeichnis wird der Linkzähler (hardlink counter) in der Inodestruktur erhöht. Bei jedem Löschen eines Verweises (mit dem Kommando *rm*) wird dieser Zähler verringert. Erreicht der Zähler Null, so kann die Inode und alle Datenblöcke, auf die die Inode verweist, freigegeben werden.

Genaugenommen werden Inode und Datenblöcke erst dann freigegeben, wenn kein Prozess mehr die Datei geöffnet hält. Der Superuser kann eine Datei löschen, die gerade in Verwendung ist. Der Prozess, der die Datei verwendet, kann ungestört weiterarbeiten, obwohl die Datei bereits aus dem Verzeichnis gelöscht wurde. Inode und Datenblöcke bestehen ja noch solange, bis der letzte *close()* die Datei endgültig schließt. Allerdings kann kein *open()* mehr auf die Datei gemacht werden, da dafür ja der Verzeichniseintrag benötigt wird. Sieht man sich die Platzbelegung des Dateisystemes mit dem Kommando *df* an, so bleibt der Platz noch belegt bis zum Ende des Prozesses, der die Datei benötigt. –> Die Summe aller Blöcke, die das Kommando *ls -l* anzeigt, muss nicht identisch sein mit dem belegten Platz, den das Kommando *df* anzeigt!

Zu diesem Thema passt noch ein anderes Phänomen unter UNIX: Eine Datei kann weniger Datenblöcke belegen als sie rechnerisch belegen müsste, d.h. sie kann *Löcher* haben. Dies kommt insbesondere bei Dateien vor, bei denen nicht sequentiell geschrieben wir, sondern (wie z.Bsp. Hashdateien) vor dem Schreiben gezielt positioniert wird. Dadurch entstehen Leerstellen, welche auch keinen Plattenplatz benötigen. Diese Eigenschaft spart nicht nur Speicherplatz und steigert die Performance, sondern hat auch unangenehme Auswirkungen. Will man Dateien von einem Dateisystem auf ein anderes kopieren (die Belegung wurde zuvor mit *df* oder *du* kontrolliert), so kann es passieren, dass am Ziellaufwerk plötzlich zu wenig Platz mehr ist. Oder laut *df*-Kommando sollte das Band für die Datensicherung ausreichen, dennoch verlangt die Sicherung ein Folgeband. Woher kommt die wunderbare Datenvermehrung?: Die Lücken werden beim Kopieren bzw. Sichern mit *NIL* (hexadezimal 0x00) aufgefüllt.

Datenblöcke und Fragmente

Ein Datenblock hat im Normalfall eine Größe von 8 KB (8192 Bytes). Belegt eine Datei nur einen Bruchteil eines Datenblockes, so wird dieser Datenblock *fragmentiert*, d.h. in Teile von 1 KB unterteilt. Die nicht benützten *Fragmente* können für eine andere Datei benützt werden. Durch dieses *Fragmentieren* kann eine Menge Plattenplatz eingespart werden.

Wächst die Datei später, sodass ein kompletter Datenblock belegt werden könnte, so werden beim ufs-Dateisystem (im Gegensatz zu DOS-Dateisystemen mit FAT o.ä.) jene Fragmente, welche von einer anderen Datei belegt werden, umgelagert sodass der gesamte Datenblock dieser einen Datei zur Verfügung steht. Damit wird ein Fragmentieren des Dateisystemes, so wie es unter DOS oder Windows passiert, weitgehend verhindert.

Erzeugen eines ufs-Dateisystems

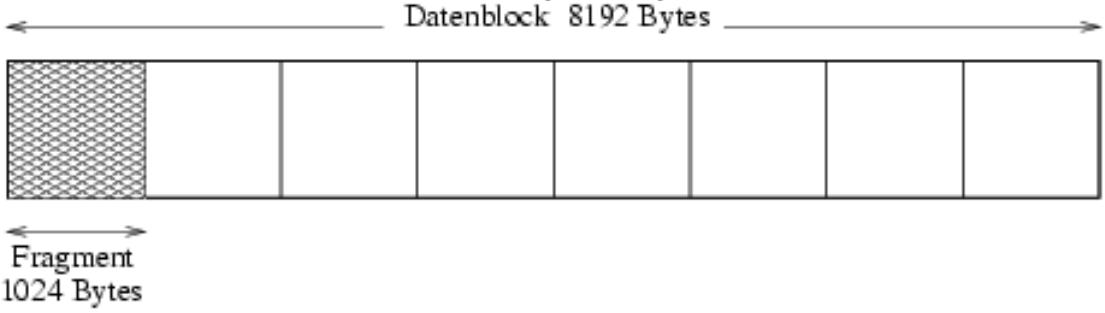
Ein ufs-Dateisystem wird mit dem Kommando **newfs** erzeugt.

Syntax:

newfs [-m %frei] *Rawpartition*

-m Platz der frei bleiben soll, damit das System Raum zum Optimieren des Dateisystems hat. (normalerweise 10%)

Abbildung 8.2: Fragmente



Ein Datenblock wird von 2 Dateien belegt.

Die erste Datei ist angewachsen. Die 2. Datei wird in einen anderen Datenblock kopiert, damit die 1. Datei möglichst in einem Stück abgespeichert werden kann.

Beispiel:

```
# newfs /dev/rdsk/c0t2d0s0
newfs: construct a new file system /dev/rdsk/c0t2d0s0: (y/n)? y
/dev/rdsk/c0t2d0s0: 41040 sectors in 57 cylinder of 9 tracks, 80 sectors
21.0MB in 4 cy groups (16 c/g, 5.90MB/g, 2688 i/g)
super-block backups (for fsck -F ufs -o b=#) at: 32, 11632, 23232, 34832,
#
```

newfs ist ein front end - Programm für **mkfs**, welches eigentlich das Dateisystem auf der Partition erzeugt. **mkfs** kennt viele Optionen, welche Parameter des Dateisystemes beeinflussen. Z.Bsp. kann die Anzahl der Inodes je Zylindergruppe, die Datenblockgröße etc. festgelegt werden. Das kann u.U. interessant sein, wenn man vorher schon weiß, dass sehr viele kleine Dateien auf dem Dateisystem abgelegt werden sollen oder nur wenige, extrem große Dateien.

Filesystemcheck fsck

Das Programm **fsck** dient dazu, Fehler und Inkonsistenzen in einem Dateisystem zu entdecken und nach Möglichkeit auch zu reparieren. Solche Fehler können entstehen, wenn das System nicht ordnungsgemäß beendet wurde (Stromausfall, Hardware-Fehler, ...) und die Daten im Speicher nicht mehr auf Platte geschrieben werden konnten.

Sobald ein Dateisystem eingehängt (gemountet) wird, wird dies mit einem Flag in den Dateisystemstrukturen markiert. Dieses Flag wird beim Aushängen (unmount) wieder gelöscht. Beim nächsten Versuch, dieses Dateisystem einzuhängen wird das Flag überprüft und es kann somit festgestellt werden, ob ein Filesystemcheck nötig ist.

Beim Systemstart wird **fsck** im *preen mode* (nicht interaktiver Modus) gestartet. Stößt *fsck* in diesem Modus auf ein Problem, das eine Eingabe erfordert, wird mit einer Fehlermeldung abgebrochen und das System bleibt im Singleuser Modus. Es muss nun ein manueller *fsck* aufgerufen werden damit die Dateisysteme repariert werden können.

Syntax:

fsck [-F ufs] [-m] [-n|-N] [-y|-Y] [-o *spezielle_Option*[,*spez.Option*]] [*Gerätedevice*]

-F ufs das zu überprüfende Dateisystem ist ein ufs-Dateisystem

-m nur überprüfen, nicht reparieren; es wird am Ende ein Ergebnis im Klartext ausgegeben

-n|-N alle Fragen, die eventuell auftreten werden automatisch mit NEIN beantwortet

-y|-Y alle anstehenden Fragen werden automatisch mit JA beantwortet

spezielle Optionen:

-o Diese Optionen werden mit **-o** eingeleitet. Es könne mehrere durch Komma getrennte spezielle Optionen angegeben werden.

b=*n* Angabe eines alternativen Superblocks. Das kann ein Dateisystem retten, falls nur der 1. Superblock defekt ist. Für *n* kann 32 angegeben werden. Das ist immer ein Ersatzsuperblock.[2]

[2] Eine Liste aller Superblockkopien kann mittels **newfs -Nv** erhalten werden. ACHTUNG! Beim Aufruf dieses Kommandos unbedingt die Option *-Nv* angeben, da sonst ein Dateisystem erzeugt wird und alle Daten verloren sind!!

f (force) Es wird ein fsck erzwungen, auch wenn das Dateisystem laut Flag *clean* ist.

p (preen mode) Das Kommando wird nicht interaktiv aufgerufen.

w Nur schreibbare Dateisysteme überprüfen

Wird kein Gerätedevice angegeben, so werden alle Dateisysteme überprüft, die in der Datei */etc/vfstab* im Feld *device to fsck* einen Eintrag haben und deren Feld *fsck pass* einen Wert ungleich 0 hat. Bei Angabe eines Gerätedevices sollte das Rawdevice angegeben werden, da die Überprüfung schneller läuft. Die Überprüfung sollte nur im ausgehängten (unmounted) Zustand oder (/ und /usr können nicht ausgehängt werden) im Singleuser-Modus erfolgen!

Der Filesystemcheck erfolgt in 5 Phasen. Dabei können u.a. folgende Inkonsistenzen festgestellt werden (unvollständige Liste):

- Anzahl der freien Datenblöcke im Superblock stimmt nicht überein mit Information im Cylinder group block
- Anzahl der freien Inodes im Superblock stimmt nicht mit Cylinder group block - Information überein
- Mehr als eine Inode verweist auf denselben Datenblock
- Wert des Linkcounters in Inode stimmt nicht mit der Anzahl der Verweise auf die Inode in den Verzeichnissen überein
- Inkonsistenz zwischen Datenblockzähler und Anzahl der Verweise auf Datenblöcke
- Auf angeblich benützte Inodes gibt es keine Verweise
- Im Verzeichnis befinden sich Verweise auf nicht existierende Inodes
- Verzeichnis der Größe Null (ungültig, da zumindest das Verzeichniss selbst und das darüberliegende Verzeichnis vorhanden sein muss!)

Beispiele:

```
# fsck                                        # überprüfen von Dateisystemen laut /etc/vfstab
# fsck /dev/rdsk/c0t0d0s7          # Dateisystem auf Slice 7 der Platte c0t0d0 wird überprüft
# fsck /opt   # Das Dateisystem dessen Mountverzeichnis /opt ist (laut Datei /etc/vfstab) wird
überprüft
# fsck -o f,p /dev/rdsk/c0t0d0s5     # fsck im nicht interaktiven Modus wird erzwungen, auch
wenn clean flag gesetzt ist
# fsck -o b=32 /dev/rdsk/c0t0d0s7 # fsck mit Hilfe des alternativen Superblocks Block-Nr. 32
```

UFS Logging

Solaris unterstützt das *UFS Logging*. Dabei werden - ähnlich wie Transaktionen bei Datenbanken - Änderungen protokolliert. Bei einem Absturz kann mit Hilfe dieser Logs eine raschere Reparatur des Dateisystems erfolgen.

Überwachung des Plattenplatzes mit df, du und quot

df Kommando

Mit **df** kann die Belegung von Dateisystemen festgestellt werden. Diese wird absolut in Blöcken bzw. Kbytes oder relativ in Prozent ausgegeben. Außerdem kann die Belegung der vorhandenen Inodes ermittelt werden.

df [-k] [-i] [*Verzeichnis*|*Gerätedevice*]

- **-k** Ausgabe der Werte in Kbytes
- **-i** Belegung der Inodes
- ***Verzeichnis*|*Gerätedevice*** Ausgabe der Daten jenes Dateisystemes, auf dem sich *Verzeichnis* bzw. *Datei* befindet

du Kommando

Mit **du** wird der benötigte Plattenplatz in Blöcken (512 Bytes) unterhalb eines Verzeichnisses ermittelt.

du [-a] [-s] [-k] [*Verzeichnis*]

- **-k** Ausgabe in KBytes anstatt in Blöcken
- **-s** Ausgabe einer Summe inkl. aller Unterverzeichnisse. Ohne dieser Option werden Informationen zu jedem Unterverzeichnis gesondert ausgegeben.
- **-a** Ausgabe nicht nur für Verzeichnisse, sondern auch für jede Datei
- ***Verzeichnis*** Verzeichnis, dessen Größe des Inhaltes ermittelt werden soll

quot Kommando

quot ermöglicht die Bestimmung des Platzverbrauches in Kbytes jedes einzelnen Benutzers. Das Kommando muss vom Superuser ausgeführt werden.

quot [-a] [-f] [*Dateisystem*]

- **-a** Liste von allen Dateisystemen
- **-f** Zusätzliche Ausgabe der Anzahl von Dateien
- ***Dateisystem*** Dateisystem, dessen Belegung überprüft werden soll

8.4 Das Mounten von Dateisystemen

Unter *mounten* versteht man das Einhängen von Dateisystemen (unabhängig von deren Struktur) in die Dateibaumstruktur. Anhand des Pfadnamens einer Datei lassen sich keine Dateisystemgrenzen erkennen, der ursprüngliche Dateibaum wurde lediglich um eine weitere Verästelung erweitert. In Windowssystemen ist dies nicht möglich. Ist ein Dateisystem zu voll um in einem Unterverzeichnis weitere Daten aufzunehmen, kann es unter Windows nicht durch einfaches Dazuhängen eines neuen Dateisystems vergrößert werden. Eine völlige Neuorganisation ist nötig.

Die Datei /etc/vfstab

In der Datei */etc/vfstab* werden Mountpoints und Gerätenamen der Dateisysteme verwaltet. Durch den Eintrag in diese Datei kann das automatische Mounten der Dateisysteme bei Systemstart realisiert werden.

Beispiel:

```
#device           device             mount           FS      fsck    mount   mount
#to mount         to fsck            point           type    pass    at boot options
#
#/dev/dsk/c1d0s2  /dev/rdsk/c1d0s2   /usr            ufs     1       yes     -
fd                -                  /dev/fd         fd      -       no      -
/proc             -                  /proc           proc    -       no      -
/dev/dsk/c0t0d0s1 -                  -               swap    -       no      -
/dev/dsk/c0t0d0s0 /dev/rdsk/c0t0d0s0 /               ufs     1       no      -
swap              -                 /tmp             tmpfs   -       yes     -
```

Die Datei enthält folgende Spalten:

device to mount Gerätedevice, welches gemountet werden soll.

device to fsck Gerätedevice, welches für den Filesystemcheck verwendet wird. Normalerweise ist dies das Character-Device (Rawdevice). Bei den Dateisystemtypen *proc*, *fd*, *tmpfs* und *swap* wird kein Filesystemcheck durchgeführt, sodass an dieser Stelle anstatt eines Gerätedevices ein '-' steht.

mount point Verzeichnisname, in den das Dateisystem eingehängt werden soll. *Swap* ist eigentlich kein Dateisystem, wird aber dennoch in diese Datei eingetragen, um beim Systemstart automatisch eingebunden zu werden. *Swap* erhält keinen Mountpoint.

FS typ Dateisystemtyp (ufs, nfs, hsfs, pcfs, swap, tmpfs, proc, fd . . .)

fsck pass Diese Zahl bestimmt die Reihenfolge, in der die Dateisysteme gecheckt werden sollen. root sollte den Wert '1' erhalten. fsck überprüft Dateisysteme, welche sich auf derselben Platte befinden sequentiell, Dateisysteme auf unterschiedlichen Platten parallel.

mount at boot *yes* – das Dateisystem wird automatisch beim Systemstart eingehängt; *no* – das Dateisystem wird nicht automatisch eingehängt.

mount options Es können mehrere durch Komma voneinander getrennte Optionen angegeben werden. Wird keine Option angegeben, so ist stattdessen ein '-' zu setzen. Die Optionen sind abhängig vom Dateisystemtyp.

mount Kommandos

Das Kommando **mount** dient zum Einhängen von Dateisystemen in einen Mountpoint bzw. zur Anzeige aller eingehängten Dateisysteme. Beim Einhängen wird ein Eintrag in die Datei */etc/mnttab* gemacht.

Syntax:

mount Ohne Angabe einer Option werden die eingehängten Dateisysteme aufgelistet. Genaugenommen wird der Inhalt der Datei */etc/mnttab* ausgewertet. Das bedeutet, dass die Ausgabe von *mount* nicht unbedingt der Tatsache entspricht!

mount -a Es werden alle jene Dateisysteme gemountet, welche in der Datei */etc/vfstab* im Feld *mount at boot* das Wort *yes* stehen haben. Die Verwendung von *mount* mit der Option *-a* entspricht dem Kommando *mountall*

mount [-o *Optionen*] *Gerätedevice* | *Mountpoint*
Ist ein Gerätedevice in der Datei /etc/vfstab eingetragen, so genügt die Angabe des Gerätedevices oder des Mountpoints.

mount [-F *FS-Typ*] [-o *Optionen*] *Gerätedevice Mountpoint*
Enthält /etc/vfstab den Eintrag des Gerätedevices nicht, so muss der Mountpoint angegeben werden. Handelt es sich nicht um ein ufs-Dateisystem, so muss auch der Dateisystemtyp angegeben werden.[3] Stimmt der angegebene Dateisystemtyp nicht mit dem tatsächlichen Dateisystem überein, so erfolgt eine Fehlermeldung.

Optionen: (unvollständig – siehe auch Seite 128)

logging (ufs-FS) UFS-Logging wird aktiviert (siehe Seite 81).

nolargefiles (ufs-FS) Das erzeugen von *large files* (Dateien > 2 GBytes) wird verhindert. Wurde schon einmal auf diesem Dateisystem ein *large file* erzeugt, so kann das Dateisystem nicht mehr mit der Option *nolargefiles* gemountet werden.

ro (Read Only) Auf das Dateisystem kann nur lesend zugegriffen werden. Beim Typ *hsfs* (CDROM) muss diese Option angegeben werden, da sonst eine Fehlermeldung erfolgt.

noatime Die access-time (siehe Kapitel 8.3.2, Seite 75) wird nicht bei jedem Zugriff aktualisiert, was die Performance verbessern kann.

Beispiele:

```
# mount /dev/dsk/c0t3d0s7 /export/home
# mount -o logging /dev/dsk/c0t3d0s6 /usr
# mount -o nolargefiles /dev/dsk/c0t3d0s7 /export/home
# mount /export/home                    # es muss ein Eintrag in /etc/vfstab vorhanden sein
# mount -F hsfs -o ro /dev/dsk/c0t6d0s0 /cdrom                    # einhängen der CD-ROM
# mount -F pcfs /dev/diskette /mnt             # einhängen einer DOS-formatierten Diskette
# mount -F nfs hostxy:/usr/share/man /usr/share/man         # einhängen eines NFS-Shares
```

[3] Der Dateisystemtyp wird nach folgender Regel behandelt:

1. falls angegeben, wird die Option -F *FS-Typ* ausgewertet
2. der FS-Typ wird der Datei */etc/vfstab* entnommen, sofern dieses Gerätedevice eingetragen ist
3. der Standard-FS-Typ wird der Datei */etc/default/fs* entnommen (normalerweise ufs)

umount Kommando

umount *mountpoint* hängt ein Dateisystem aus und entfernt den Eintrag in der Datei */etc/mnttab*.

mountall, umountall

mountall und **umountall** kennen jeweils nur die Option *-l*. Dies bezieht sich auf lokale Dateisysteme. **mountall** hängt alle Dateisysteme ein, die in der Datei */etc/vfstab* eingetragen sind und *mount at boot* auf *yes* gesetzt haben.**umountall** hängt alle Dateisysteme aus. Dateisysteme, die noch benützt werden (Prozesse haben Ressourcen in diesem Dateisystem geöffnet) können nicht ausgehängt werden. Es erfolgt eine Fehlermeldung, dass das Dateisystem *in use* bzw. *busy* ist. Um den Prozess ausfindig zu machen, kann das Kommando *fuser*[4] verwendet werden (siehe Seite 46 und im Online Manual).

[4] **fuser** ist dafür geeignet, alle Prozesse, die eine bestimmte Datei (auch Gerätedatei) geöffnet haben zu finden bzw. mit der Option *-k* zu beenden.

8.5 Volume Management

Seit Solaris 7 gibt es den *volume management daemon* ***vold***. Dieser Dämon-Prozess überwacht CD-ROM und Diskettenlaufwerk und mountet automatisch Datenträger, sobald diese eingelegt werden.[5]

Versucht man diese Datenträger manuell ein- bzw. auszuhängen, so kann dies zu unvorhergesehenen Problemen führen. Es kann passieren, dass kein Zugriff auf CD-ROM bzw. Diskette mehr möglich ist! Wird manuelles mounten bevorzugt, so muss zuvor der vold-Dämon beendet werden (über das rc-Runskript */etc/init.d/volmgt* {*start*|*stop*}).

Befindet sich ein Dateisystem auf dem Datenträger, so wird dieser unter folgendem Pfad gemountet:

Medium	Pfadname
CD-ROM	/cdrom/*cdrom-name*
Diskette	/floppy/*floppy-name*

Ist kein Dateisystem auf dem Datenträger, kann dieser nicht gemountet werden. Zugreifen kann man dann direkt über den Gerätenamen:

Medium	Pfadname
CD-ROM	/vol/dev/aliases/cdrom0
Diskette	/vol/dev/aliases/floppy0

[5] Mit dem Kommando *volcheck* kann das Überprüfen, ob ein Datenträger eingelegt wurde manuell ausgeführt werden.

8.6 Virtual Volume Management

Um die Einschränkung von nur einer physikalischen Platte je Dateisystem aufheben zu können wurde die *Virtual Volume* Struktur geschaffen. Damit ist es möglich, Dateisysteme über mehrere Festplatten und Festplattenslices zu legen. Aus der Sicht des Betriebssystemes sind Zugriffe auf ein Virtual Volume gleich wie auf 'herkömmliche' Slices. Es gibt auch hier Character- und Block-Devices. Lediglich die Namen der Gerätedevices lauten anders. Programme wie *newfs, fsck* oder *mount* können bei Gerätedevices von Virtual Volumes vollkommen gleich angewandt werden wie bei Gerätedevices von herkömmlichen Slices.

Unter Solaris stehen 2 unterschiedliche *Virtual Volume Manager* zur Verfügung:

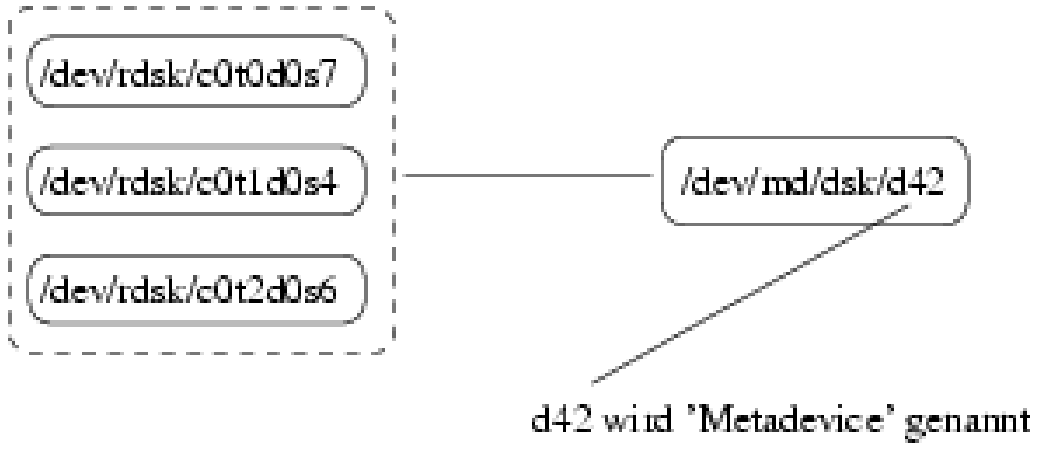
- Solstice DiskSuite
- Sun StorEdge Volume Manager

Solstice DiskSuite

Typischer Gerätename von Virtual Volumes, welche mit Solstice DiskSuite erstellt wurden sind z.Bsp.:

/dev/**md**/**r**dsk/d42	Character Device
/dev/**md**/dsk/d42	Block Device

Solstice verwendet Slices, welche mit dem Standardprogramm *format* partitioniert wurden. Solstice fasst mehrere (theoretisch auch nur 1 Stück möglich) solcher Slices zu einem *Virtual Volume* zusammen.



Das Produkt *Solstice DiskSuite* befindet sich auf der *Solaris Easy Access Server 2.0* CDROM. Die Installationsschritte sind:

```
# cd /cdrom/cdrom0                    # Ins richtige Verzeichnis auf der CD wechseln
# ./installer &                       # Das Installationsprogramm starten
```

Bei der anschließenden Auswahl der Software *Solstice AdminSuite* und *DiskSuite* auswählen. Nach der Installation muss noch der Patch *104468-11* installiert werde, der sich auf derselben CD befindet:

```
# cd /cdrom/cdrom0/products/AdminSuite_2.3+AutoClient_2.1/Patches/sparc # Wechsel ins Verzeichnis
# patchadd 104468-11                                                   # Installation des Patch
```

Aufgerufen wird *DiskSuite* durch das Kommando:

```
# solstice &
```

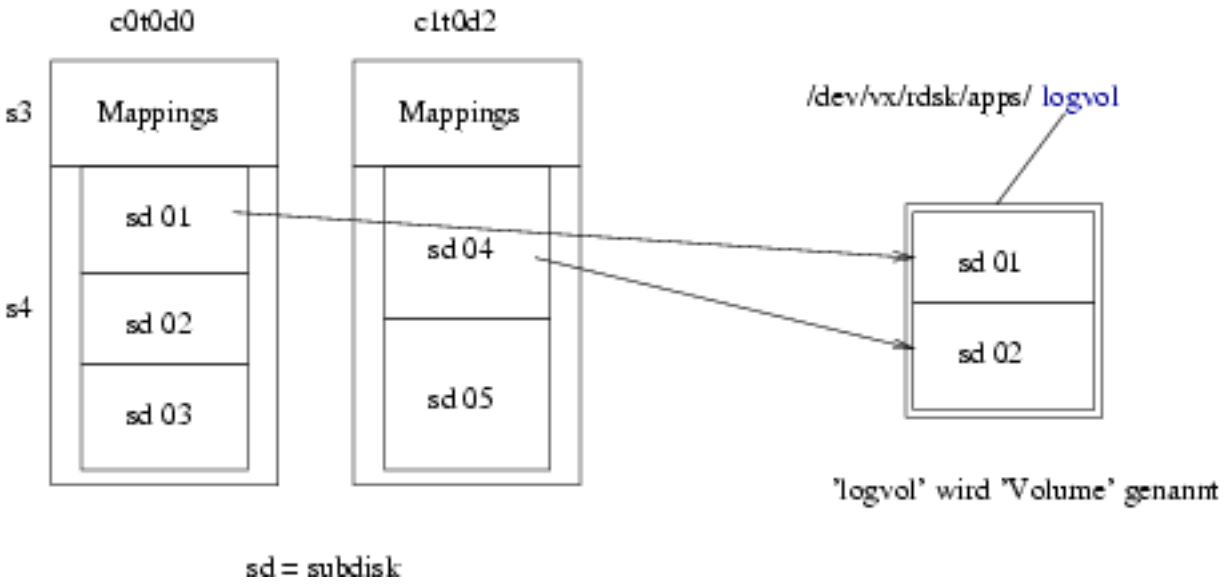
Sun StorEdge Volume Manager

Beispiel für Gerätenamen:

/dev/**vx**/**r**dsk/apps/logvol	Character Device
/dev/**vx**/dsk/apps/logvol	Block Device

Sun StoreEdge Volume Manager teilt Festplatten in nur 2 Slices (Slice 3 und Slice 4) auf. Slice 3 wird *privat area* genannt und enthält Verwaltungsinformation. Der Platz von Slice 4 wird für die Bereitstellung von *virtual devices* verwendet. Aneinandergrenzende Gruppen von Sektoren können zu *subdisks* zusammengefasst werden.

Ein Vorteil dieses Konzeptes ist, dass die Anzahl von Subdisks je Festplatte unbeschränkt ist. Hingegen ist mittels herkömmlicher Partitionierung mit *format* eine maximale Anzahl von 8 Slices je Platte möglich.



Virtual Volume Typen

Häufige Anwendung finden die beiden Virtual Volume Typen:

- Concatenated Volumes
- Striped Volumes

Abbildung 8.3: Concatenated Volume

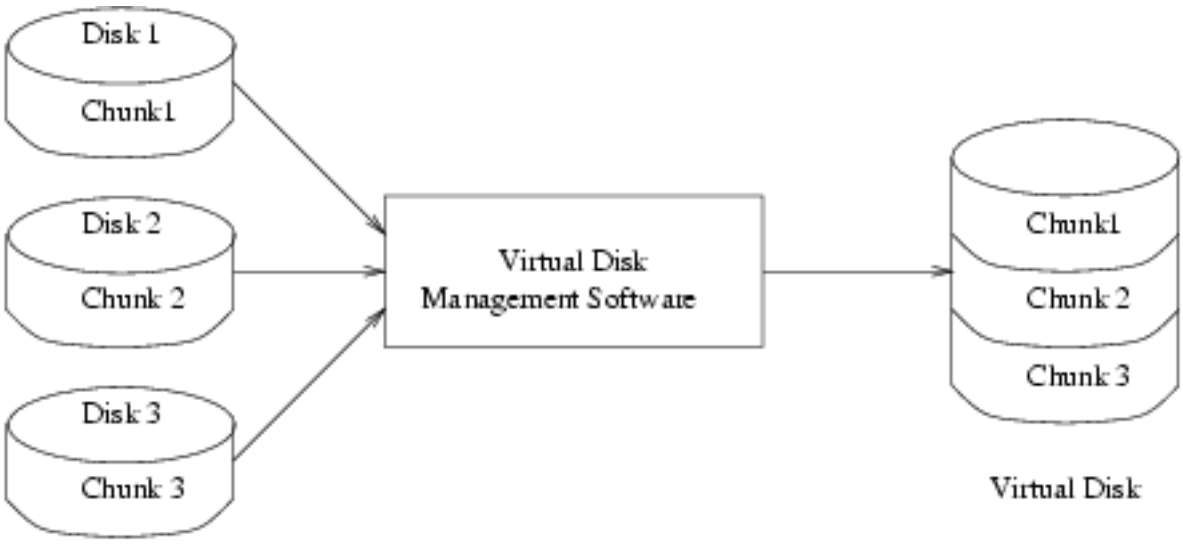
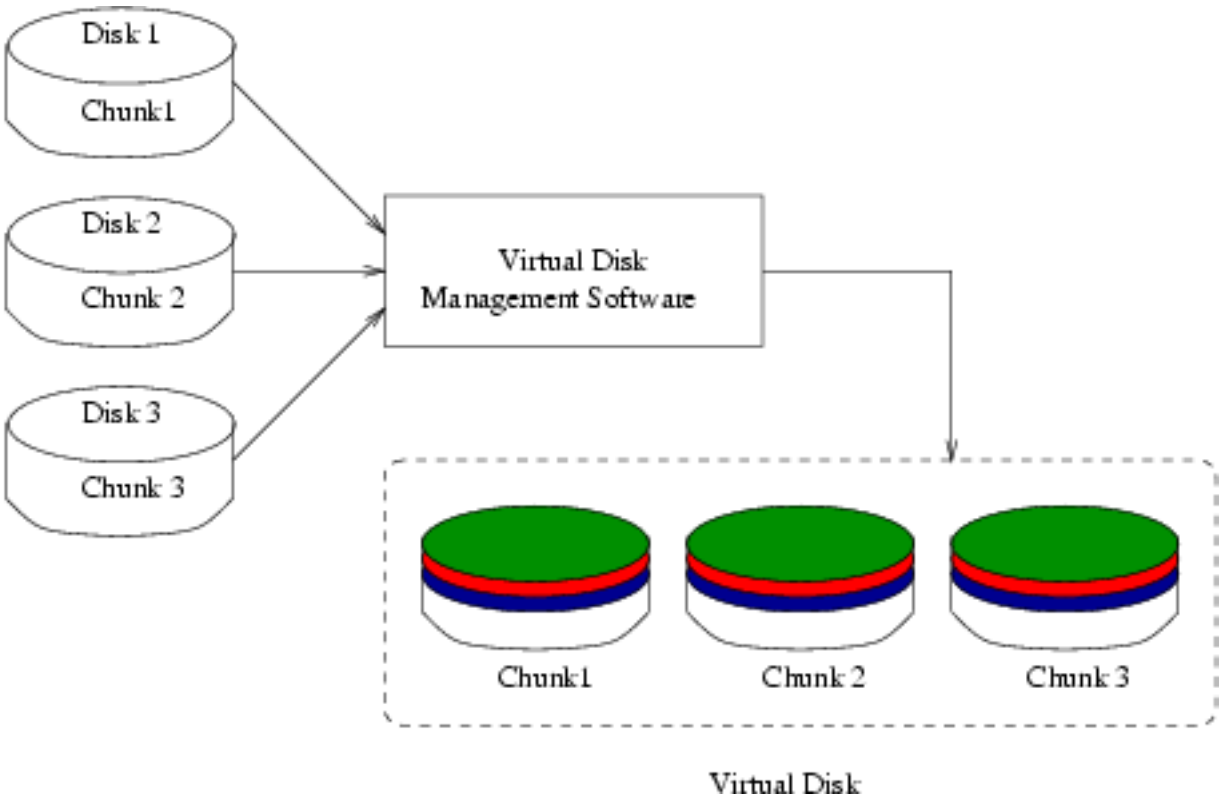


Abbildung 8.4: Striped Volume



Concatenated Volumes *Concatenated Volumes* sind mehrere zu einem logischen Laufwerk hintereinandergereihte Platten. Beim Beschreiben dieses logischen Laufwerkes wird zunächst die erste Platte gefüllt, anschließend jeweils eine Weitere.

Besondere Eigenschaften dieses Typs:

- Man kann ein logischen Laufwerk erzeugen, das größer ist als eine physikalische Platte
- Das logische Laufwerk kann im laufenden Betrieb vergrößert werden durch Anhängen einer weiteren Platte.

Striped Volumes Bei einem *Striped Volume* werden auch mehrere einzelne Platten zu einer logischen Einheit zusammengefasst. Aber zum Unterschied von Concatenated Volumes werden die Daten auf alle Einzelstücke verteilt. Dadurch wird die Performance beträchtlich verbessert. Beim Schreiben auf die logische Platte wird eine gewisse Anzahl von Bytes zunächst auf die erste Platte, der nächste Teil auf die nächste usw. geschrieben. Die Größe dieser Teilstücke nennt sich *stripe size* und kann angepasst werden, um die Performance zu optimieren. Typische Werte für die stripe size sind 64 KBytes.

Besondere Eigenschaften:

- Es macht Sinn, die physikalischen Platten auf unterschiedliche Kontroller zu hängen.
- Daten können parallel geschrieben werden.
- Striped Volumes bringen eine große Performancesteigerung
- Die Segmentgröße (stripe size) kann zur Optimierung der Performance eingestellt werden.
- Eine typische Segmentgröße ist 64 KBytes.

9 Datensicherung

Datenverluste können entstehen durch Hardwarefehler, Softwarefehler und bewusste oder unbewusste menschliche Eingriffe (z.Bsp. versehentliches Löschen von vermeintlich alten Dateien). Für diese Fälle, aber auch um Daten zu archivieren ist es nötig, Dateien oder ganze Dateisysteme auf andere Datenträger auszulagern.

Man unterscheidet bei Sicherungen zwischen Komplettsicherungen (full backup) und inkrementellen Teilsicherungen (incremental backup):

full backup Der ausgewählte Bereich wird vollständig gesichert.

incremental backup Im ausgewählten Bereich wird nur das gesichert, was sich seit der letzten Vollsicherung oder der letzten Teilsicherung verändert hat.

9.1 Sicherungslaufwerke

Nach dem Einbau eines Bandlaufwerkes können durch Eingabe des Kommandos *tapes* die erforderlichen Geräteknoten erzeugt werden.

```
# tapes                    erzeugt Links für Tapes in /dev/rmt/ auf Geräteknoten
```

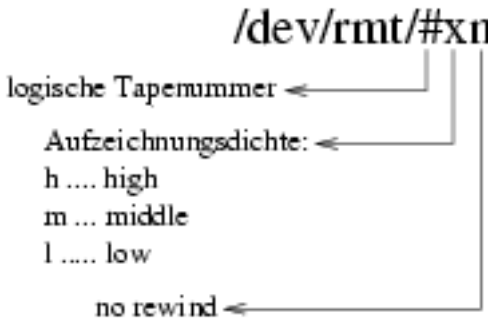
Zur Datensicherung sind derzeit (Stand: Ende 1998) folgende Datenträger üblich:

Medium	Kapazität
½“ Band	120 MB
QIC ¼“ Cartridge	150 MB bis 8 GB
Exabyte 8mm Cartridge	bis 40 GB
4mm DAT Cartridge	bis 24 GB
Digital linear tape (DLT)	bis 70 GB

ACHTUNG! Die meisten Laufwerke unterstützen eine Hardware-Komprimierung, um die Kapazität zu erhöhen. Diese Art der Kompression ist wesentlich schneller, als die Komprimierung per Software am System, erreicht dafür aber bei weitem nicht dessen Komprimierungsraten. Werden vorwiegend Dateien gesichert, die bereits komprimiert vorliegen, so sollte die Hardware-Kompression ausgeschaltet werden, da diese Dateien sonst am Band größer werden!

Namen von Bandlaufwerken

Unter Solaris haben die Namen aller Bandlaufwerke, egal welchen Herstellers und welchen Typs, folgenden Aufbau:



Die *logische Tapenummer* ist eine Ziffer, die bei ’0’ beginnt. Das erste Bandlaufwerk im System erhält die Zahl ’0’.

Die *Aufzeichnungsdichte* kann folgende Werte tragen:

h high density

m medium density

l low density

c compressed

u ultra

Nicht jeder Laufwerkstyp unterstützt alle Aufzeichnungsdichten. Wird kein Wert angegeben, so wird standardmäßig *high* und *uncompressed* verwendet.

No rewind wird verwendet, um ein Zurückspulen am Ende der Sicherung zu verhindern. Das ist dann notwendig, wenn mehrere Sicherungsarchive hintereinander auf dasselbe Band gespielt werden sollen. Z.Bsp. ist es sinnvoll, alle Partitionen der Systemplatte als getrennte Archive auf ein Band zu sichern. Man kann dann bei Bedarf auf das gewünschte Archiv positionieren (mittels Kommando *mt*) und gezielt auf ein bestimmtes Archiv zugreifen. Im untenstehenden Beispiel werden alle verfügbaren Namen des ersten Laufwerkes im System aufgelistet. Wie man sieht, sind diese Namen lediglich symbolische Links auf den eigentlichen Geräteknoten im Verzeichnis */devices*.

```
pisch@sunlicht:/dev/rmt # > ls -l
Gesamt 48
lrwxrwxrwx  1 root  root  43 Dez 17 12:20 0 -> ../../devices/pci@1f,0/pci@1/scsi@1/st@5,0:
lrwxrwxrwx  1 root  root  44 Dez 17 12:20 0b -> ../../devices/pci@1f,0/pci@1/scsi@1/st@5,0:b
lrwxrwxrwx  1 root  root  45 Dez 17 12:20 0bn -> ../../devices/pci@1f,0/pci@1/scsi@1/st@5,0:bn
lrwxrwxrwx  1 root  root  44 Dez 17 12:20 0c -> ../../devices/pci@1f,0/pci@1/scsi@1/st@5,0:c
lrwxrwxrwx  1 root  root  45 Dez 17 12:20 0cb -> ../../devices/pci@1f,0/pci@1/scsi@1/st@5,0:cb
lrwxrwxrwx  1 root  root  46 Dez 17 12:20 0cbn -> ../../devices/pci@1f,0/pci@1/scsi@1/st@5,0:cbn
lrwxrwxrwx  1 root  root  45 Dez 17 12:20 0cn -> ../../devices/pci@1f,0/pci@1/scsi@1/st@5,0:cn
lrwxrwxrwx  1 root  root  44 Dez 17 12:20 0h -> ../../devices/pci@1f,0/pci@1/scsi@1/st@5,0:h
```

```
lrwxrwxrwx  1 root  root  45 Dez 17 12:20 0hb -> ../../devices/pci@1f,0/pci@1/scsi@1/st@5,0:hb
lrwxrwxrwx  1 root  root  46 Dez 17 12:20 0hbn -> ../../devices/pci@1f,0/pci@1/scsi@1/st@5,0:hbn
lrwxrwxrwx  1 root  root  45 Dez 17 12:20 0hn -> ../../devices/pci@1f,0/pci@1/scsi@1/st@5,0:hn
lrwxrwxrwx  1 root  root  44 Dez 17 12:20 0l -> ../../devices/pci@1f,0/pci@1/scsi@1/st@5,0:l
lrwxrwxrwx  1 root  root  45 Dez 17 12:20 0lb -> ../../devices/pci@1f,0/pci@1/scsi@1/st@5,0:lb
lrwxrwxrwx  1 root  root  46 Dez 17 12:20 0lbn -> ../../devices/pci@1f,0/pci@1/scsi@1/st@5,0:lbn
lrwxrwxrwx  1 root  root  45 Dez 17 12:20 0ln -> ../../devices/pci@1f,0/pci@1/scsi@1/st@5,0:ln
lrwxrwxrwx  1 root  root  44 Dez 17 12:20 0m -> ../../devices/pci@1f,0/pci@1/scsi@1/st@5,0:m
lrwxrwxrwx  1 root  root  45 Dez 17 12:20 0mb -> ../../devices/pci@1f,0/pci@1/scsi@1/st@5,0:mb
lrwxrwxrwx  1 root  root  46 Dez 17 12:20 0mbn -> ../../devices/pci@1f,0/pci@1/scsi@1/st@5,0:mbn
lrwxrwxrwx  1 root  root  45 Dez 17 12:20 0mn -> ../../devices/pci@1f,0/pci@1/scsi@1/st@5,0:mn
lrwxrwxrwx  1 root  root  44 Dez 17 12:20 0n -> ../../devices/pci@1f,0/pci@1/scsi@1/st@5,0:n
lrwxrwxrwx  1 root  root  44 Dez 17 12:20 0u -> ../../devices/pci@1f,0/pci@1/scsi@1/st@5,0:u
lrwxrwxrwx  1 root  root  45 Dez 17 12:20 0ub -> ../../devices/pci@1f,0/pci@1/scsi@1/st@5,0:ub
lrwxrwxrwx  1 root  root  46 Dez 17 12:20 0ubn -> ../../devices/pci@1f,0/pci@1/scsi@1/st@5,0:ubn
lrwxrwxrwx  1 root  root  45 Dez 17 12:20 0un -> ../../devices/pci@1f,0/pci@1/scsi@1/st@5,0:un
pisch@sunlicht:/dev/rmt # >
```

9.2 Sicherungskommandos

Im Solaris-System inkludiert sind mehrere geeignete Programme. Für die Sicherung von kompletten Dateisystemen eignet sich vor allem **ufsdump** und das Gegenstück **ufsrestore**.

ufsdump Kommando

ufsdump wird verwendet, um komplette Dateisysteme zu sichern. Es ist aber auch möglich einzelne Verzeichnisse oder Dateien damit zu sichern. Es unterstützt Komplettsicherungen (full backup) sowie inkrementelle Teilsicherung (incremental backup).

Syntax:

ufsdump *Optionen* [*Argumente*] *Sicherungsdatei(en)*

- **0-9** Setzt den *Dump-Level*. Level 0 als niedrigster Level führt eine Vollsicherung durch. Die Levels 1-9 werden für Differenzsicherungen (inkrementelle Sicherung) verwendet
- **u** *Update dump record* – Es wird ein Vermerk in die Datei */etc/dumpdates* geschrieben. Dieser enthält das gesicherte Laufwerk, das Datum und den Level der Sicherung. Diese Daten werden bei Differenzsicherungen benötigt.
- **b *Blockfaktor*** Gibt die Größe der Datenblöcke auf Band in Vielfachen von 512 Bytes an. Der Standardwert wird von ufsdump selbstständig anhand des Laufwerkstyps gewählt (laut Online-Manual).
- **f *Device*** Gibt den Gerätenamen des Bandlaufwerkes an
- **v** Nach erfolgter Sicherung wird der Inhalt der Sicherung mit den Daten am Dateisystem verglichen.
- ***Sicherungsdatei*** Das ist üblicherweise das Gerätedevice (raw oder block) des Dateisystemes. Dieses Dateisystem sollte möglichst nicht gemountet sein (ufsdump kann mit der ufs-Dateisystemstruktur umgehen).

Weitere Optionen –> siehe Online Manual

Differenzsicherung:

Mit *ufsdump* werden Differenzsicherungen durch Verwendung der Levels 0-9 durchgeführt. Ein bestimmter Level sichert nur jene Daten, die sich seit der letzten Sicherung mit niedrigerem Level verändert haben. Z. Bsp. könnten folgende Sicherungspläne sinnvoll sein:

Wochentag	Level	gesicherte Daten
Freitag	0	Vollsicherung
Montag	1	Differenz seit Freitag
Dienstag	1	Differenz seit Freitag
Mittwoch	1	Differenz seit Freitag
Donnerstag	1	Differenz seit Freitag

Wird im obigen Beispiel am Mittwoch eine Platte kaputt, so muss nach deren Tausch zunächst die Vollsicherung vom letzten Freitag und anschließend die jüngste Sicherung (Dienstag oder Mittwoch - je nachdem, ob der Schaden vor oder nach der Mittwoch - Sicherung entstand) eingespielt werden. Dies ist eine sehr günstige Sicherungsmethode, da nur einmal pro Woche eine Vollsicherung nötig ist, aber dennoch relativ wenig Restaurierungsvorgänge zur Wiederherstellung der Daten benötigt werden.

Wochentag	Level	gesicherte Daten
Sonntag	0	Vollsicherung
Montag	3	Differenz seit gestern
Dienstag	4	Differenz seit gestern
Mittwoch	2	Differenz seit Sonntag
Donnerstag	5	Differenz seit gestern
Freitag	6	Differenz seit gestern
Samstag	1	Differenz seit Sonntag

Stirbt die Platte in unserem Beispiel am Samstag vor der Sicherung, so müssen nacheinander folgende Sicherungen eingelesen werden: Vollsicherung vom letzten Sonntag - Differenzsicherung vom Mittwoch, Donnerstag und Freitag. Der Aufwand ist erheblich größer, dafür ist die tägliche Sicherungsmenge kleiner. Hat man Zeitprobleme die täglichen Sicherungen unterzubringen, könnte dies eine Lösung sein.

Beispiel einer Sicherung mit *ufsdump*:[1]

```
pisch@sunlicht:/ # > ufsdump 0uf /dev/rmt/0 /export/home/pie
DUMP: Writing 32 Kilobyte records
DUMP: Date of this level 0 dump: Mit 03 Mai 2000 14:35:59 MET DST
DUMP: Date of last level 0 dump: the epoch
DUMP: Dumping /dev/rdsk/c0t0d0s0 (sunlicht:/) to /dev/rmt/0.
DUMP: Mapping (Pass I) [regular files]
DUMP: Mapping (Pass II) [directories]
DUMP: Estimated 3620 blocks (1,77MB).
DUMP: Dumping (Pass III) [directories]
DUMP: Dumping (Pass IV) [regular files]
DUMP: 3582 blocks (1,75MB) on 1 volume at 681 KB/sec
DUMP: DUMP IS DONE
pisch@sunlicht:/ # >
```

ufsrestore Kommando

Mit **ufsrestore** können Sicherungen zurückgespielt werden, die mit *ufsdump* erzeugt wurden. Beim Zurückspielen der Daten mit *ufsrestore* wird eine Datei mit dem Namen *restoresymtable* erzeugt. Diese Datei wird während des Einspielens von Differenzsicherungen benötigt, sollte aber anschließend gelöscht werden.

Syntax:

[1] In diesem Fall wurde nicht ein gesamtes Dateisystem, sondern nur ein Unterverzeichnis gesichert.

ufsrestore *Optionen* [*Argumente*] [*Dateinamen, . . .*]

Optionen:

- **i** Interaktiver Restore – Die Datenrestaurierung erfolgt im Dialog. Es können einzelne Dateien oder Verzeichnisse markiert werden, welche dann zurückgespielt werden.
- **r** Die komplette Sicherung wird retourgespielt
- **t** Listet den Inhalt der Sicherung auf.
- **x** Nur jene Dateien werden zurückgespielt, die in der Kommandozeile angegeben werden (*Dateinamen, . . .)*
- **f *Device*** Angabe des Bandlaufwerkes (bzw. der Datei) das die Sicherung enthält
- **v** verbose – Die Dateien werden während des Einspielens aufgelistet

Um eine Datensicherung retourzuspielen, muss das entsprechende Dateisystem gemountet sein (war die Platte zuvor kaputt, so muss zuerst ein Dateisystem eingerichtet und dieses gemountet werden). Man muss sich genau im Wurzelverzeichnis der gesicherten Dateien befinden, da *ufsdump* die Dateien mit relativem Pfadnamen sichert.

```
pisch@sunlicht:/ # > cd /export/home/pie                    # Wechsel ins richtige Verzeichnis
pisch@sunlicht:/export/home/pie/ # > ufsrestore ivf /dev/rmt/0       # Aufruf von ufsrestore
Verify volume and initialize maps
Media block size is 126
Dump   date: Wed May 03 14:35:59 2000
Dumped from: the epoch
Level 0 dump of a partial file system on sunlicht:/export/home/pie
Label: none
Extract directories from tape
Initialize symbol table.
ufsrestore > pwd
/
ufsrestore > ls                                              # Anzeige des Sicherungsinhaltes
.:
2 *./           2 *../      186074  export/
ufsrestore > cd export/home/pie/kurs      # Verzeichniswechsel innerhalb der Sicherungsdaten
ufsrestore > ls
./export/home/pie/kurs:
361772  ./        441110  devices/  420183  man/
138375  ../       414600  files/    457197  platten/
ufsrestore > add man                                    # Auswahl des gewünschten Verzeichnisses
Make node ./export
Make node ./export/home
Make node ./export/home/pie
Make node ./export/home/pie/kurs
Make node ./export/home/pie/kurs/man
ufsrestore > ls                                  # Kontrolle, ob Verzeichnis nun markiert ist
./export/home/pie/kurs:
361772 *./        441110  devices/  420183 *man/
138375 *../       414600  files/    457197  platten/
ufsrestore > extract                                                  # Start des Retoursicherns
Extract requested files
You have not read any volumes yet.
Unless you know which volume your file(s) are on you should start
with the last volume and work towards the first.
```

```
Specify next volume #: 1
extract file ./export/home/pie/kurs/man/kill.1
extract file ./export/home/pie/kurs/man/ps.1
extract file ./export/home/pie/kurs/man/format.1m
extract file ./export/home/pie/kurs/man/fsck.1m
extract file ./export/home/pie/kurs/man/fsck_ufs.1m
extract file ./export/home/pie/kurs/man/newfs.1m
extract file ./export/home/pie/kurs/man/mkfs.1m
extract file ./export/home/pie/kurs/man/mkfs_ufs.1m
extract file ./export/home/pie/kurs/man/mount.1m
extract file ./export/home/pie/kurs/man/mount_nfs.1m
extract file ./export/home/pie/kurs/man/mount_ufs.1m
extract file ./export/home/pie/kurs/man/mount_hsfs.1m
extract file ./export/home/pie/kurs/man/eeprom.1m
extract file ./export/home/pie/kurs/man/ufsdump.1m
extract file ./export/home/pie/kurs/man/ufsrestore.1m
extract file ./export/home/pie/kurs/man/shutdown.1m
Add links
Set directory mode, owner, and times.
set owner/mode for '.'? [yn] y
ufsrestore >
ufsrestore > quit
pisch@sunlicht:/export/home/pie/mist # >
```

tar Kommando

tar ist ein weit verbreitetes Kommando, das zur Sicherung von Verzeichnissen in ein Archiv (ob auf Band, Diskette oder in eine Datei auf Platte) verwendet wird. Zum Beispiel werden auch Patches für Solaris als Tar-Archiv ausgeliefert.

Syntax:

tar {c|t|x}[v][p][f *Archivdateidatei*] *Dateiname* [*Dateiname*, . . .]

- **c** Sicherungsarchiv erzeugen
- **t** Inhalt des Sicherungsarchives anzeigen
- **x** Dateien aus Sicherungsarchiv einspielen
- **f *Archivdatei*** Als Archivdatei wird normalerweise das Bandgerät oder Diskettenlaufwerk angegeben. Es kann sich aber auch um eine Datei handeln. Wird anstatt eines Dateinamens das Zeichen '-' angegeben, so wird von *stdin* gelesen (mit der Option *t* oder *x*) bzw. auf *stdout* geschrieben (mit der Option *c*).
- **v** Während des lesens/schreibens werden die Dateien aufgelistet
- **p** (*preserve*) Es werden die Rechte übernommen (umask - Einstellung bleibt unberücksichtigt). Als Superuser werden auch setuid- oder sgid-Bit übernommen
- ***Dateiname*** Angabe der Datei bzw. des Verzeichnisses, welches gesichert werden soll. Bei der Angabe eines Verzeichnisses, wird auch alles darunter gesichert.

tar unterstützt noch eine Vielzahl an zusätzlichen Optionen, welche aber seltener verwendet werden -> siehe Online Manual.

Beispiel:

```
pisch@sunlicht:/export/home/pie # > tar cvf archiv.tar bin
a bin/ 0K
a bin/tree 1K
a bin/locate 1K
pisch@sunlicht:/export/home/pie # >
```

```
pisch@sunlicht:/export/home/pie # > tar xvf archiv.tar
tar: Blockgröße = 7
x bin, 0 bytes, 0 tape blocks
x bin/tree, 404 bytes, 1 tape blocks
x bin/locate, 35 bytes, 1 tape blocks
pisch@sunlicht:/export/home/pie # >
```

mt Kommando

mt ermöglicht es, Kommandos an das Bandgerät zu schicken.

Syntax:

mt [-f *Gerätedevice*] *Kommando* [*Zähler*]

-f *Gerätedevice* Name des Bandlaufwerkes. Es ist zu beachten, dass für die meisten Kommandos das *norewind*-Device angegeben werden muss (z.Bsp. /dev/rmt/0**n**).

Kommandos:

status gibt den Status des Gerätes aus

rewind spult Band zurück

retension spult Band bei Befarf an den Beginn, dann ans Ende und wieder bis zum Beginn

erase löscht das gesamte Band

fsf # positioniert um Anzahl # Archive vorwärts

bsf # positioniert um Anzahl # Archive rückwärts

eom Positioniert zur Endemarke des Bandes

10 Drucken mit Solaris

10.1 Print Service Architektur

Das Drucksystem von Solaris hat Client-Server Architektur. Der *print server* nimmt Druckaufträge entgegen, stellt diese in eine Warteschlange (Spool) und schickt die Daten an den Drucker, der entweder lokal an einer Schnittstelle oder im Netzwerk hängt. Der Druckserver ist auch dafür verantwortlich, dass keine Druckdaten bei Fehlersituationen verloren werden.

Der *print client* bildet die Schnittstelle zum Anwender bzw. zum Programm. Er schickt Aufträge zum Druckserver.

Seit der Solaris Version 2.6 hat das Drucksystem eine neue Architektur. Der *Print Protocol Adapter* (*in.lpd*)ersetzt den früher verwendeten *listen* und *lpNet*. Der Spool wurde dadurch modularer, es können mehrere Spoolsysteme am selben System koexistieren und bietet die Möglichkeit der Erweiterung durch Drittsoftware (z.Bsp. Unterstützung von Apple- oder Novell-Druckprotokoll). Das Drucksystem unterstützt das *BSD print protocol*, welches bei Solaris um einige Funktionen erweitert wurde.

Folgende Software-Pakete sind für das Drucksystem relevant:

Paket Name	Beschreibung	Basisverzeichnis
SUNWpcr	SunSoft Print - Client	root (/)
SUNWpcu	SunSoft Print - Client	/usr
SUNWpsr	SunSoft Print - LP Server	root (/)
SUNWpsu	SunSoft Print - LP Server	/usr
SUNWpsf	Postscript Filter	/usr
SUNWscplp	SunSoft Print - Source Compatibility	/usr

Es ist möglich nur den Print-Client zu installieren, wenn der Server nicht benötigt wird. Serverseitig muss allerdings auch der Client installiert werden.

Abbildung 10.1: Druck auf lokalen Drucker

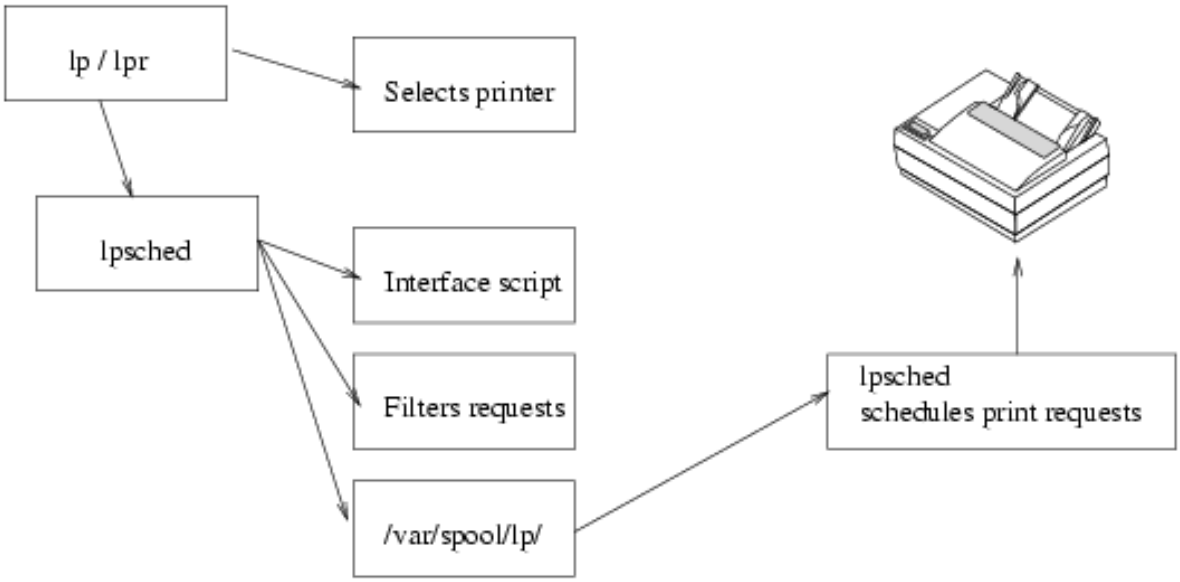
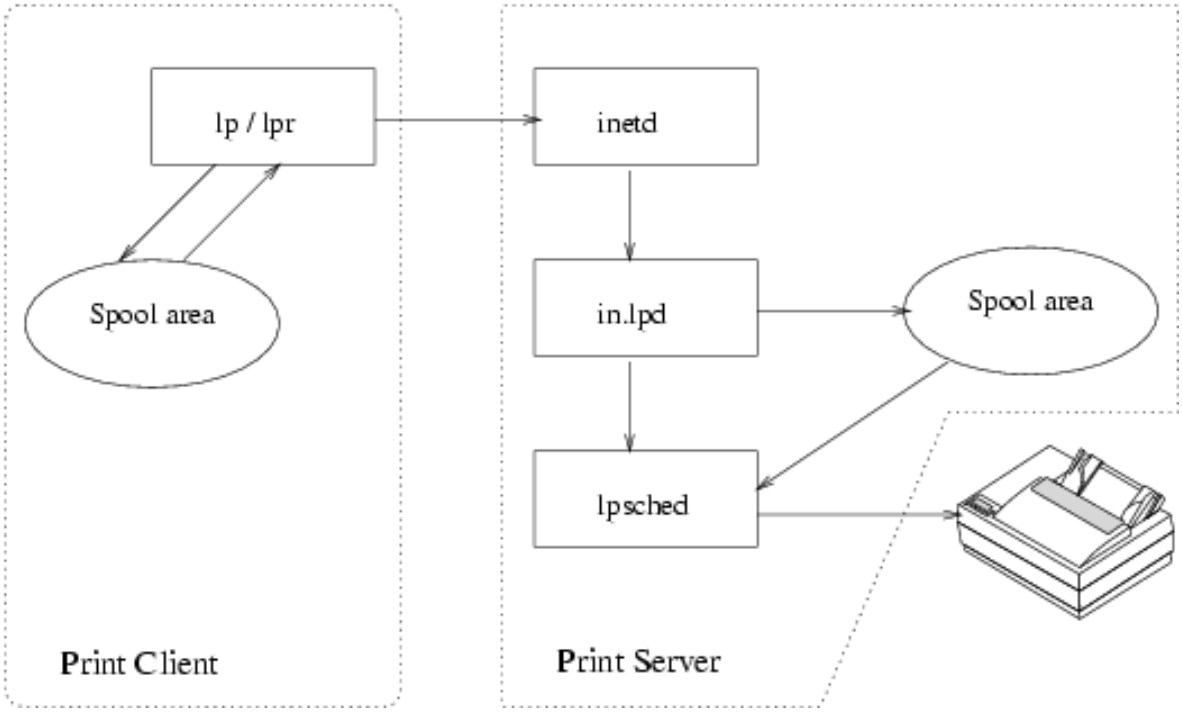


Abbildung 10.2: Druck auf Remote-Drucker



10.1.1 Verzeichnisse des Drucksystems

Verzeichnis	Inhalt
/usr/bin	User Kommandos
/etc/lp	Druckserver Konfigurationsdateien
/usr/share/lib	terminfo Datenbank
/usr/sbin	Administrator Kommandos
/usr/lib/lp	lp Dämonen, Verzeichnisse für Binärdateien und Postscript Filter
/var/lp/logs	lp Dämon Logs
/var/spool/lp	Spoolverzeichnis für Druckaufträge

10.1.2 Druckfunktionen

Queuing Wenn Druckaufträge erstellt werden, gelangen diese zunächst in eine Warteschlange. Dieser Vorgang wird *queuing* genannt. Man kann durch die Kommandos *accecpt* bzw. *reject* die Aufnahme in die Warteschlange erlauben oder sperren.

Tracking Der Druckdienst überprüft zyklisch jeden Druckauftrag. Der User kann seine Druckaufträge löschen und der Administrator kann den Status der Aufträge ändern. Derselbe Mechanismus regelt auch das Wiederaufsetzten von Drucken nach Systemabsturz oder Reboot.

Fault Notification Das Drucksystem liefert bei Problemen Fehlermeldungen an die Konsole oder per Mail. Es kann eine individuelle Fehlermeldung einstellbar.

Initialization Bevor ein Auftrag an den Drucker geschickt wird, muss der Drucker initialisiert werden.

Filtering Druckaufträge benötigen teils - je nach Dateiformat - eine Aufbereitung der Daten. Beispielsweise müssen Grafikdateien anders behandelt werden als normale Textdateien. Diese Datenkonviertierung wird durch *filter* durchgeführt.

10.1.3 Content Types und Filter

Jeder Druckauftrag enthält zumindest eine Datei. Der Inhalt der Datei kann in unterschiedlichen Formaten vorliegen. Diese Formate werden *content types* genannt. Ein Beispiel für einen content type ist *PostScript*. Abhängig vom Content Type müssen unterschiedliche Filterprogramme (*print filters*) aufgerufen werden, um die Datei druckerspezifisch aufzubereiten.

Bei der Druckerkonfiguration muss angegeben werden, welche *content types* der Drucker akzeptiert. Nicht für jeden Druckertyp können alle Dateiformate aufbereitet werden.

Standard-PostScriptfilter befinden sich unter dem Verzeichnis */usr/lib/lp/postscript*. Filterbeschreibungsdateien sind in */etc/lp/fd*. Die sogenannte *filter lookup tabel* ist in der Datei */etc/lp/filter.table* enthalten.

10.1.4 Printer Types

Der Druckertyp (*printer type*) legt fest, welche *terminfo*-Beschreibungsdatei verwendet wird. Diese Dateien liegen im Binärformat im Verzeichnis */usr/lib/terminfo*. Die Dateien enthalten die Steuersequenzen des entsprechenden Druckertyps. Der Inhalt der Binärdatei kann mit Hilfe des Kommandos *infocmp* im Klartext ausgegeben werden.

10.1.5 Interface Programs

Interface Programme sind im Allgemeinen Shell-Skripte, die dafür sorgen, die Schnittstelle zum Drucker zu initialisieren und die Daten zum Drucker zu senden. Außerdem können dem Interfaceprogramm über das Druckprogramm (*lp* oder *lpr*) Optionen (wie z.Bsp. *nobanner* etc.) übergeben werden. Die Interfaceprogramme werden bei der Konfiguration der Drucker in das Verzeichnis */etc/lp/interfaces* mit dem Druckernamen kopiert.

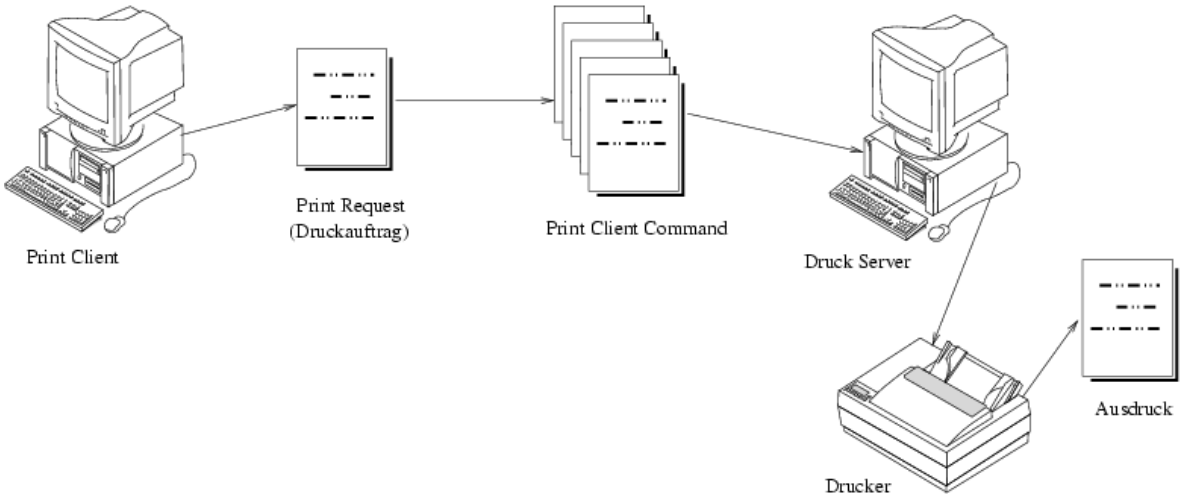
10.2 Druckerkonfiguration

Für die Konfiguration eines Druckers sind folgende Schritte nötig:

- Installation der Drucker-Hardware
- Print-Server einrichten
  Der Printserver kann mittels *admintool* oder per Kommando mit *lpadmin* und *lpfilter* konfiguriert werden.
- Print-Client einrichten
  Der Client kann ebenso mit *admintool* oder *lpadmin* konfiguriert werden.

10.3 Der Druckvorgang

Beim Absetzen eins Druckauftrages passieren folgende Abläufe:



1. Ein User setzt mittels Druckkommando einen Druckauftrag ab. Dieser wird in die Druckerwarteschlange eingereiht.
2. Das Drucker-Clientprogramm wählt einen passenden Drucker aus um den Auftrag zu senden.
3. Der Drucker-Client schickt den Druckauftrag mittels BSD Druckprotokoll an den Druck-Server.
4. Der Druck-Server bearbeitet den Auftrag und sendet ihn an den Drucker.
5. Der Drucker bringt den Auftrag auf Papier

10.4 Druckerauswahl

Der Printer Clientprozess muss entscheiden an welchen Drucker der Druckauftrag geschickt werden muss. Folgende Kriterien werden der Reihe nach ausgewertet:

1. Der gewünschte Drucker wurde bereits beim Druckkommando angegeben:
   `# lp -d` *`printername datei`*
   oder
   `# lpr -P` *`printername datei`*
2. Umgebungsvariable *PRINTER* (wird bei Verwendung des Kommandos *lpr* zuerst ausgewertet) oder *LPDEST* (wird bei Verwendung von *lp* zuerst ausgewertet) ist gesetzt (Name des Standarddruckers). Wird kein Drucker angegeben und eine der beiden Variablen ist gesetzt, so wird dieser Drucker verwendet.
3. *$HOME/.printers* enthält einen Eintrag „*_default:bsdaddr=server,printer:*“. Existiert diese Datei und sie enthält eine Zeile mit dem Schlüsselwort „*_default*“, wird dieser Drucker verwendet.
4. Die Datei */etc/printers.conf* enthält einen Eintrag „*_default*“?
5. Existiert ein *_default*-Eintrag im Nameservice *NIS* oder *NIS+*?

10.5 Druckkommandos

Kommando	Funktion
lp	sendet Daten zum Druckserver
lpstat	Zeigt Status des Print Service
cancel	löscht Druckaufträge
lpadmin	für diverse Administrationsaufgaben
accept	erlaubt das Einspoolen von Aufträgen
reject	verhindert das Einspoolen von Aufträgen
lpmove	lenkt Druckauftrag auf anderen Drucker um
enable	erlaubt das Ausspoolen (Ausgabe) zum Drucker
disable	verhindert das Ausspoolen (Ausgabe) zum Drucker

10.5.1 Starten und Stoppen des Drucksystems:

/etc/init.d/lp stop Stoppen des Drucksystemes

/etc/init.d/lp start Starten des Drucksystemes

10.5.2 lp Kommando

lp [-d *Druckername*] [-n *Kopien*] [-o *spezielle Optionen*] *Dateiname*

-d *Druckername* Schickt Druckauftrag zu bestimmten Drucker. Wird diese Option nicht verwendet, so wird der Standarddrucker verwendet. Nach welchen Kriterien der Standarddrucker ausgewählt wird siehe Kapitel „Druckerauswahl“, Seite 105.

-n *Kopienanzahl* Veranlasst den Druckjob mehrmals zu drucken.

-q # Priorität des Druckjobs festlegen. Priorität ’0’ ist die höchste mögliche Priorität.

-o *Option* Erlaubt die Angabe von speziellen Optionen. Bei der Angabe von mehreren Optionen müssen diese zwischen Hochkommas und durch Leerzeichen voneinander getrennt eingegeben werden (siehe Beispiel).
Unter anderen sind folgende Optionen möglich:

nobanner Gibt keine Banner-Seite aus

nofilebreak Gibt mehrere Dateien ohne Seitenumbruch dazwischen aus

length=#[i|c] Legt Seitenlänge in Zeilen (ohne Buchstabe am Ende), Inch oder Zentimetern fest

width=#[i|c] Legt Seitenbreite fest (Einheiten wie oben)

lpi=# Legt Zeilenabstand in Zeilen pro Zoll fest

cpi=#|pica|elite|compressed Legt Zeichenabstand fest

stty='*stty-Option*' Erlaubt spezielle Schnittstelleneinstellungen (siehe Online Manual zum Kommando *stty*)

Beispiel:

```
# lp meineDatei
request id is sunlicht-17 (1 file)
# lp -n 2 meineDatei
request id is sunlicht-18 (1 file)
# lp -d HP_1OGA -o "nobanner nofilebreak" *.txt
request id is sunlicht-19 (4 files)
```

10.5.3 lpstat Kommando

lpstat zeigt den Status der Druckwarteschlange an

lpstat [-d] [-R] [-s] [-t] [-a [*list*]] [-o [*list*]] [-p [*list*]][1]

-d Zeigt Namen des Standarddruckers an

-R Zeigt Position der Druckaufträge in der Warteschlange an

-s Zeigt Sammelinformation der für dieses System verfügbaren Drucker an

-t Zeigt dasselbe wie Option *-s* und zusätzlich den Status der Drucker

-a [*Druckernamen*] Zeigt an, ob der/(die) angegebene(n) Drucker Aufträge annimmt oder nicht

-o [*Druckernamen*] Zeigt Status der Druckauftäge für den/die Drucker

-p [*Druckernamen*] Zeigt Status und Verfügbarkeit der Drucker an

Beispiel:

```
pisch@sunlicht:/etc/lp # > lpstat -t
scheduler is running
system default destination: lp
device for HP_1901C: /dev/null
character set c
HP_1901C accepting requests since Don 02 Mär 2000 18:33:55 MET
printer HP_1901C is idle. enabled since Don 02 Mär 2000 18:33:55 MET. available.
pisch@sunlicht:/etc/lp # >
```

10.5.4 cancel Kommando

Mit **cancel** werden Druckaufträge aus der Warteschlange entfernt.

cancel [*Job-ID*] [*Druckername*]

cancel -u *Username* [*Druckername*]

[1] Die Liste ist unvollständig. Siehe Online Manual.

Job-ID Jeder Druckauftrag erhält eine Job-ID, welche mit dem Kommando *lpstat* ermittelt werden kann. Es kann damit ein bestimmter Druckauftrag entfernt werden.

Druckername Wird nur der Druckername angegeben, werden alle diesem Drucker zugeordneten Aufträge gelöscht

-u *Username* Löschen von Druckaufträgen eines bestimmten Benutzers. Wird nur der User angegeben, werden alle Aufträge dieses Benutzers gelöscht

Beispiele:

Umlenken von Jobs auf einen anderen Drucker:

```
# reject -r "Drucker HPLJ3 ist defekt" HPLJ3  # weitere Druckaufträge an diesen Drucker abweisen
# lpstat -o HPLJ3                              # anstehende Druckaufträge auflisten
# lpstat -a HPLJ4M                             Kontrolle, ob HPLJ4M Aufträge annimmt
# lpmove HPLJ3 HPLJ4M                          # Jobs von HPLJ3 auf HPLJ4M umlenken
# accept HPLJ3                                 # HPLJ3 wieder aktivieren nach Reparatur
```

11 Geräte Administration

11.1 Serielle Geräte

Ein serielles Gerät (Drucker, Terminal, Modem, ...) ist ein Gerät, das Daten Bit für Bit nacheinander überträgt. Ein paralleles Gerät schickt die Daten im Gegensatz dazu byteweise (ein oder sogar mehrere Bytes zugleich). Ein defacto Standard für parallele Schnittstellen ist das *Centronics^{TM} Parallel Printer Interface*.
Telefonleitungen verwenden nur 2 Drähte, sodass nur eine serielle Übertragung möglich ist.

Ein *Interface Port* ist eine Verbindung, die Datenübertragung zwischen Geräten erlaubt. Sie besteht aus einem Hardwareteil und einem Softwareteil (Gerätetreiber). Beispiele dafür sind: RS232, Centronics Interface, RJ-45 UTP, SCSI, ...

Die Seriellen Ports bei Sun verwenden die Standards *RS-423* (25-poliger RS-232 Stecker) und *RS-232*. Sun-Workstations werden im Allgemeinen mit 2 seriellen Schnittstellen ausgeliefert. Diese werden *ttyA* und *ttyB* genannt (*teletypewriter*).

Alphanumerische Terminals Gerät mit serieller Ein-/Ausgabeschnittstelle, Tastatur und Bildschirm. Es können nur alphanumerische Zeichen dargestellt werden.

Modems Der/das Modem ist ein DCE (Data Communications Equipment), das Informationen in Signale wandelt (moduliert), damit diese über Telefonleitungen übertragen werden können. An der Gegenstelle steht wiederum ein Modem, das die Signale empfängt, demoduliert und über die Schnittstelle an den Computer weiterreicht. Modem ist ein Kurzwort für *modulator-demodulator*

Modemtypen:

Auto-dial-only Ein Modem das mittels Puls- oder Tonwahl (je nachdem, was für den Verbindungsaufbau erforderlich ist) eine Verbindung zur Gegenstelle aufbauen kann. Die Entgegennahme von Anrufen ist nicht möglich.

Auto-answer-only Ein Modem, das die beiden Signale RI (Ring Indicator) und DCD (Data Carrier Detected) erkennt und bei ankommendem Ruf Daten mit dem angeschlossenen Computer austauschen kann. Ein aktiver Verbindungsaufbau mittels Wahl ist nicht möglich.

Bidirectional Ein Modem, das beide Funktionen (dial-in und dial-out) unterstützt.

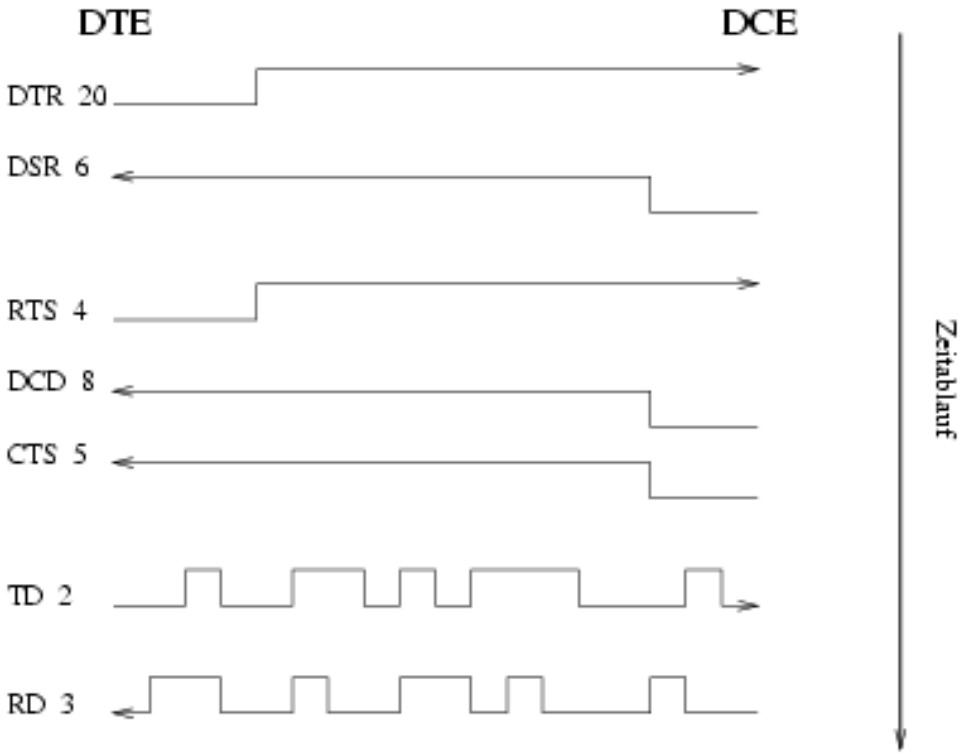
Serielle Drucker Serielle Drucker werden gleich wie Terminals und Modems angeschlossen.

Alle Gerätetypen benötigen ein *service* (ein Programm), das die Datenübertragung zwischen den Geräten durchführt. Dies ist üblicherweise ein *Portmonitor*, der ständig auf anstehende Daten wartet, um sie zu übertragen.

Data Terminal Equipment (DTE) Terminals, Drucker oder Workstations; Diese Geräte senden Daten bei einer 15-poligen RS232-Schnittstelle am Pin 2 und empfangen Daten am Pin 3.

Data Communications Equipment (DCE) Modems, Multiplexer, Data Switches; senden auf Pin 3, empfangen auf Pin 2;

Verbindungsaufbau zwischen DTE (Workstation) und DCE (Modem)



1. Wenn die Workstation (DTE) bereit ist zu senden (beim open() der Geräteschnittstelle), wird das Signal DTR (Data Terminal Ready) am Pin 20 der RS232 (25-polig) gesetzt.[1]
2. Die DTE überprüft nun, ob das Modem (DCE) bereit ist anhand des Signales DSR (Data Set Ready) am Pin 6.
3. Nun setzt die DTE das Signal RTS (Request To Send) am Pin 4 der Schnittstelle um anzuzeigen, dass sie Daten senden möchte.

[1] Wenn das Signal DTR fehlt, so kann man davon ausgehen, dass entweder die Hardware defekt ist, oder das Programm (Portmonitor) nicht korrekt läuft.

4. Das Modem überprüft, ob ein Trägersignal vorhanden ist und wenn ja, wird das Signal DCD (Data Carrier Detected) am Pin 8 gesetzt.

5. Anschließend setzt das Modem noch das Signal CTS (Clear To Send) am Pin 5, was der Workstation signalisiert, dass das Modem Daten an die Gegenstelle übertragen kann.

6. Nun beginnt der Datenaustausch über die beiden Leitungen TD (Transmit Data) und RD (Receive Data) auf Pin 2 bzw. Pin3.

11.2 Service Access Facility

Die *Service Access Facility* (*SAF*) ermöglicht den Zugriff auf serielle Geräte. Sie verwendet *STREAMS*-basierte Character Ein-/Ausgabe. Das *Streams-Modul 'ldterm'* wird verwendet für die Terminal Ein-/Ausgabe. Dieses Modul wird auch *line discipline module* genannt.

SAF Startprozess

1. *init* liest während des Systemstarts die Datei */etc/inittab*. Darin befindet sich ein Eintrag, dass der Prozess *sac* beim Wechsel in den Runlevel 2 zu starten ist.
2. *sac* liest die Datei */etc/saf/_sysconfig*. Diese Datei ist im Normalfall leer, kann aber Umgebungsvariablen enthalten. Beispielsweise kann darin die Zeitzone für Terminals in unterschiedlichen Regionen gesetzt werden.
3. Anschließend liest *sac* seine Administrationsdatei */etc/saf/_sactab*. Darin stehen Einträge, welche Portmonitore zu starten sind.
4. Jeder Portmonitor liest seine eigene Konfigurationsdatei
5. Jeder *Service Tag* - Eintrag informiert den Portmonitor, welche Services für jeden Port zu starten sind.

/etc/saf/_sactab

```
# VERSION=1
zsmon:ttymon::0:/usr/lib/saf/ttymon #
```

 # VERSION= gibt die Version der Service Access Facility an

zsmon Name des Portmonitors ('*portmonitor tag*' bzw. pmtag); Das ist der Name des Unterverzeichnisses von */etc/saf* in dem die Konfigurationsdaten des Portmonitors zu finden sind.

ttymon Typ des Portmonitors

:: an dieser Stelle können folgende 2 Kennzeichner stehen:

- **d** disable Portmonitor
- **x** starte Portmonitor nicht

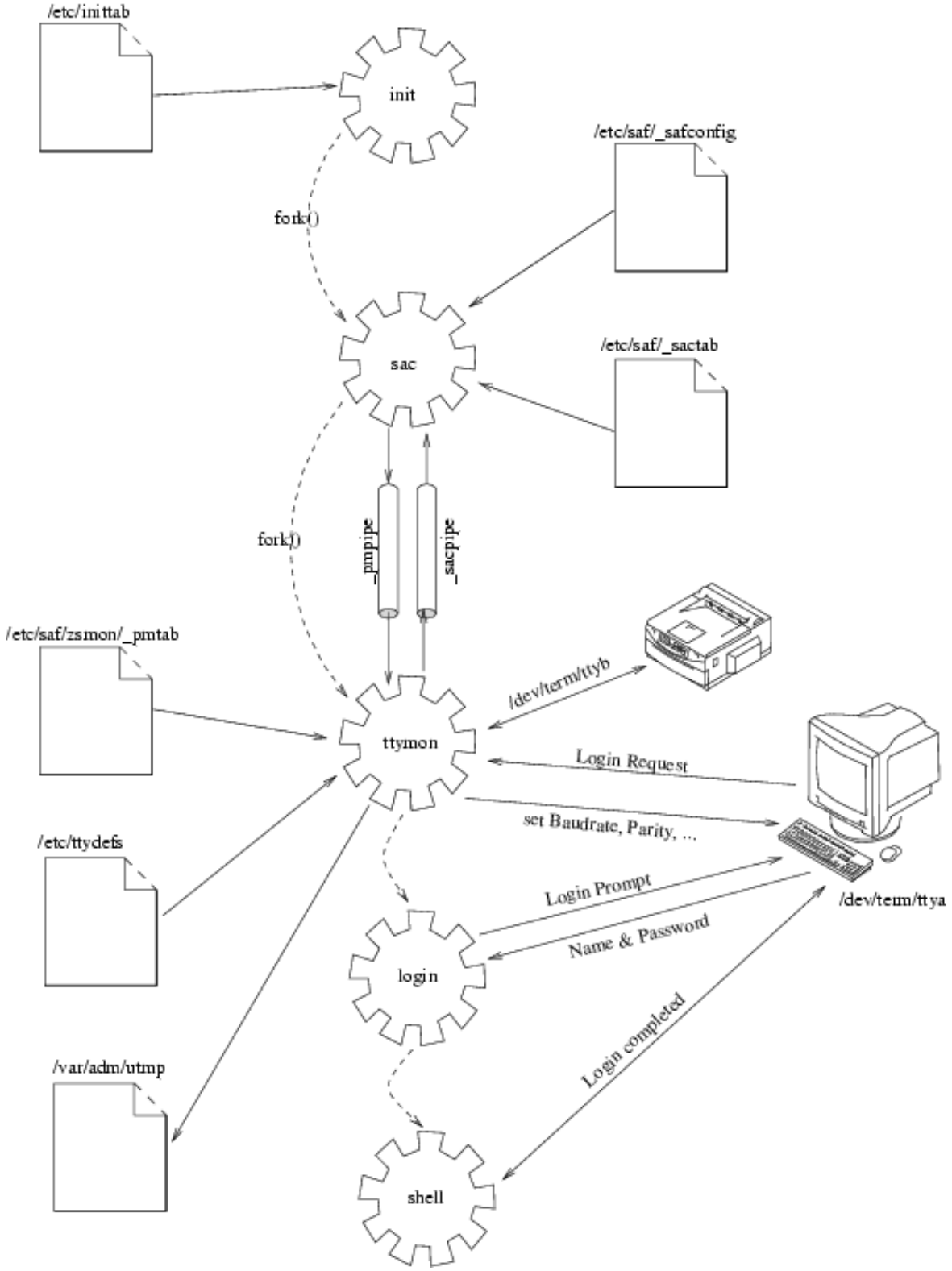
0 Returnwert für den Portmonitor. Der Wert '*0*' bedeutet, dass der Portmonitor nicht erneut gestartet wird, wenn er durch Fehler abbricht.

/usr/lib/saf/ttymon Gibt den Pfadnamen des Portmonitors an

/etc/saf/zsmon/_pmtab

```
# VERSION=1
ttya:u:root:reserved:reserved:reserved:/dev/term/a:I::/usr/bin/login::9600:ldterm,ttcompat:ttya
login\: ::tvi925:y:#
```

Abbildung 11.1: Service Access Facility



#VERSION= Version des Portmonitors

ttya Name des Service (’*service tag*’)

u an dieser Stelle sind folgende Kennzeichner möglich:

- **u** Erzeuge einen Eintrag in der Datei */var/adm/utmp* für diesen Service
- **x** disable Service

root legt Identität des Service fest

reserved reserviertes Feld

reserved reserviertes Feld

reserved reserviertes Feld

/dev/term/a Pfadname des TTY-Ports

l an dieser Stelle sind folgende Kennzeichner möglich:

- **c** connect on carrier
- **b** bidirectional
- **h** unterdrückt Hangup nach incoming call
- **l** Initialisiert Port
- **r** zwingt ttymon auf die Ausgabe der Loginaufforderung so lange zu warten, bis ein Zeichen am Port ankommt

/usr/bin/login Pfadname des Services, der bei Verbindungsaufbau gestartet wird

9600 Gibt Label in der Datei */etc/ttydefs* an

ldterm,ttcompat Streams-Module, welche verwendet werden sollen

ttya login\: Loginaufforderung, welche angezeigt werden soll

tty925 Terminaltyp

y Zeigt an, ob Software-Carrier gesetzt wird oder nicht (’*n*’)

Kommandoübersicht

sacadm -a | -r | -e | -d | -k | -s | -l [-p *pmtag*] -t *type* -c *command* -v *version*

-a *add* – Portmonitor hinzufügen

-r *remove* – Portmonitor löschen

-e *enable* – aktivieren eines Portmonitors

-d *disable* – deaktivieren eines Portmonitors

-l *list* – auflisten von Portmonitoren

-k *kill* – Portmonitor stoppen

-s *start* – Portmonitor starten

-p *pmtag* Kennzeichner (*tag*) des Portmonitors

-t *type* Portmonitor-Type (z.Bsp. ttymon)

-c *command* Kommando, welches den Portmonitor startet

-v *n* Versionsnummer des Portmonitors

Beispiel:

```
# sacadm -a -p newpmtag -t ttymon -c /usr/lib/saf/ttymon -v  `ttyadm -V `
```

pmadm -a | -r | -e | -d | -l [-p *pmtag* | -t *type*] -s *servicetag* -i *identify* -f *flag* -v *version* -m

-a *add* – Service für einen Portmonitor hinzufügen

-r *remove* – Service entfernen

-e *enable* – Service aktivieren

-d *disable* – Service deaktivieren

-l *list* – Services auflisten

-p *pmtag* Kennzeichner (*tag*) des Portmonitors

-s *servicetag* Kennzeichner des Service

-i *identify* ID, welche dem Service zugeordnet wird (meist '*root*'). Die ID muss in der Datei */etc/passwd* eingetragen sein.

-f flag Folgende Flags sind möglich:

u Der Service macht einen Eintrag in die Datei *utmp*

x Der Service wird gestartet, aber nicht *enabled*

-v *n* Versionsnummer des Portmonitors

-m *command* Portmonitor-spezifisches Konfigurationsparameter

-r remove – entfernen des Service

Beispiel:

```
# pmadm -a -p newpmtag -s ttya -i root -fu -v 1 \
-m " `ttyadm -l 9600 -d /dev/term/a -T tvi925 -i 'terminal disabled' \
-s /usr/bin/login -S y `"
```

ttyadm -l *ttylabel* -d *device* [-T *terminal_type*] [-i *message*] [-p *prompt*] -s *service* [-S y|n] [-V]

-l *ttylabel* Angabe eines Labels der Datei */etc/ttydefs*

-d *device* Voller Pfadname des Gerätedevices

-T *terminal_type* Terminaltype laut Eintrag im Verzeichnis */usr/share/lib/terminfo*

-i *message* Meldung, welche ausgegeben wird, wenn der Service deaktiviert (*disabled*) ist.

-p *prompt* Prompt bei aktiviertem Service (z.Bsp. **login:**)

-s *service* Voller Pfadname des Service Programmes

-S y|n Software carrier detect ist ein- (*y*) bzw. ausgeschaltet (*n*)

ACHTUNG!

Die Konfiguration der Konsole an der seriellen Schnittstelle muss in der Datei */etc/inittab* erfolgen:

```
co:234:respawn:/usr/lib/saf/ttymon -g -h -p " uname -n console login: " -T terminal_type -d /dev/console
-l console -m ldterm,ttcompat
```

Beispiele für die Verwendung von SAF-Kommandos:

```
# sacadm -k -p zsmon                                    # Stoppen des Portmonitors
# sacadm -s -p zsmon                                    # Starten des Portmonitors
# sacadm -d -p zsmon                                 # Deaktivieren des Portmonitors
# sacadm -e -p zsmon                                   # Aktivieren des Portmonitors
# sacadm -r -p zsmon                               # Löschen eines Portmonitor-Eintrages
# pmadm -d -p zsmon -s ttyb                          # Deaktivieren eines Services
# pmadm -e -p zsmon -s ttyb                            # Aktivieren eines Services
```

Konfiguration eines bidirektionalen Modemanschlusses:

```
# sacadm -r -p zsmon                                # Entfernen des aktuellen Portmonitors
# ttyadm -V                                              # Feststellen der Versionsnummer
1
# sacadm -a -p zsmon -t ttymon -c /usr/lib/saf/ttymon -v 1     # Portmonitor hinzufügen
# pmadm -a -p zsmon -s b -i root -fu -v 1 -m " `ttyadm -b -d /dev/term/b \
-l contty3H -m ldterm,ttcompat -s /usr/bin/login -S n `"            # Service für die
Schnittstelle /dev/term/b hinzufügen
```

Nun muss noch ein Eintrag in die Datei */etc/remote* hinzugefügt werden:

```
cuab:dv=/dev/cua/b:br#2400
```

Die Datei /etc/ttydefs

Die Datei */etc/ttydefs* enthält Einstellungsparameter für TTY-Ports. Diese Werte werden von *ttymon* verwendet, um die Schnittstelle zu initialisieren. Jeder Eintrag (jeweils eine Zeile) besteht aus 5 Feldern, welche durch Doppelpunkt ’*:*’ voneinander getrennt sind:

ttylabel Das erste Feld enthält den Kennzeichner, das sogenannte *ttylabel*, welches beim Kommando *ttyadm* mit der Option *-l* angegeben wird.

initial-flags Mit diesen Parametern wird die Schnittstelle initialisiert. Die Syntax entspricht der, wie sie mit dem Kommando *stty* verwendet wird.

final-flags Diese Einstellungen führt *ttymon* durch, direkt nachdem eine Verbindungsaufforderung erfolgte.

autobaud Steht das Zeichen ’*A*’ in diesem Feld, so wird *autobaud* aktiviert.

nextlabel Dieses Feld enthält das *ttylabel* jenes Eintrages, der beim Empfang eines BREAK-Signals als nächstes aktiviert werden soll. Damit können nacheinander unterschiedliche Baudraten ’ausprobiert’ werden, bis die Einstellung passt.

Auszug aus der Datei */etc/ttydefs*:

```
# VERSION=1
460800:460800 hupcl:460800 hupcl::307200
307200:307200 hupcl:307200 hupcl::230400
230400:230400 hupcl:230400 hupcl::153600
153600:153600 hupcl:153600 hupcl::115200
115200:115200 hupcl:115200 hupcl::76800
76800:76800 hupcl:76800 hupcl::57600
57600:57600 hupcl:57600 hupcl::38400
38400:38400 hupcl:38400 hupcl::19200
19200:19200 hupcl:19200 hupcl::9600
9600:9600 hupcl:9600 hupcl::4800
4800:4800 hupcl:4800 hupcl::2400
2400:2400 hupcl:2400 hupcl::1200
1200:1200 hupcl:1200 hupcl::300
300:300 hupcl:300 hupcl::460800

460800E:460800 hupcl evenp:460800 evenp::307200
307200E:307200 hupcl evenp:307200 evenp::230400
230400E:230400 hupcl evenp:230400 evenp::153600
153600E:153600 hupcl evenp:153600 evenp::115200
115200E:115200 hupcl evenp:115200 evenp::76800
76800E:76800 hupcl evenp:76800 evenp::57600
57600E:57600 hupcl evenp:57600 evenp::38400
38400E:38400 hupcl evenp:38400 evenp::19200
19200E:19200 hupcl evenp:19200 evenp::9600
9600E:9600 hupcl evenp:9600 evenp::4800
4800E:4800 hupcl evenp:4800 evenp::2400
2400E:2400 hupcl evenp:2400 evenp::1200
1200E:1200 hupcl evenp:1200 evenp::300
300E:300 hupcl evenp:300 evenp::19200

auto:hupcl:sane hupcl:A:9600

console:9600 hupcl opost onlcr:9600::console
console1:1200 hupcl opost onlcr:1200::console2
console2:300 hupcl opost onlcr:300::console3
console3:2400 hupcl opost onlcr:2400::console4
console4:4800 hupcl opost onlcr:4800::console5
console5:19200 hupcl opost onlcr:19200::console

contty:9600 hupcl opost onlcr:9600 sane::contty1
contty1:1200 hupcl opost onlcr:1200 sane::contty2
contty2:300 hupcl opost onlcr:300 sane::contty3
contty3:2400 hupcl opost onlcr:2400 sane::contty4
contty4:4800 hupcl opost onlcr:4800 sane::contty5
contty5:19200 hupcl opost onlcr:19200 sane::contty


4800H:4800:4800 sane hupcl::9600H
9600H:9600:9600 sane hupcl::19200H
19200H:19200:19200 sane hupcl::38400H
38400H:38400:38400 sane hupcl::2400H
```

```
53 2400H:2400:2400 sane hupcl::1200H
54 1200H:1200:1200 sane hupcl::300H
55 300H:300:300 sane hupcl::4800H
56
57 conttyH:9600 opost onlcr:9600 hupcl sane::contty1H
58 contty1H:1200 opost onlcr:1200 hupcl sane::contty2H
59 contty2H:300 opost onlcr:300 hupcl sane::contty3H
60 contty3H:2400 opost onlcr:2400 hupcl sane::contty4H
61 contty4H:4800 opost onlcr:4800 hupcl sane::contty5H
62 contty5H:19200 opost onlcr:19200 hupcl sane::conttyH
63
64 ppp115200:115200 cs8 -parenb hupcl:115200 cs8 -parenb hupcl::uu115200
65 ppp57600:57600 cs8 -parenb hupcl:57600 cs8 -parenb hupcl::uu57600
66 ppp38400:38400 cs8 -parenb hupcl:38400 cs8 -parenb hupcl::uu38400
67 ppp19200:19200 cs8 -parenb hupcl:19200 cs8 -parenb hupcl::uu19200
68 ppp9600:9600 cs8 -parenb hupcl:9600 cs8 -parenb hupcl::uu9600
69 ppp4800:4800 cs8 -parenb hupcl:4800 cs8 -parenb hupcl::uu4800
70 uu115200:115200 cs7 parenb parodd hupcl:115200 cs7 parenb parodd hupcl::
       ppp57600
71 uu57600:57600 cs7 parenb parodd hupcl:57600 cs7 parenb parodd hupcl::
       ppp38400
72 uu38400:38400 cs7 parenb parodd hupcl:38400 cs7 parenb parodd hupcl::
       ppp19200
73 uu19200:19200 cs7 parenb parodd hupcl:19200 cs7 parenb parodd hupcl::
       ppp9600
74 uu9600:9600 cs7 parenb parodd hupcl:9600 cs7 parenb parodd hupcl::ppp4800
75 uu4800:4800 cs7 parenb parodd hupcl:4800 cs7 parenb parodd hupcl::
       ppp115200
```

terminfo Verzeichnis und termcap Datenbasis

Im Verzeichnis */usr/share/lib/terminfo* befinden sich Gerätebeschreibungen für Terminals und Drucker. Die Option '*-T*' des Kommandos *ttyadm* bezieht sich auf diese Daten. Für jeden Gerätetyp existiert eine Datei in einem Unterverzeichnis, welches als Namen den ersten Buchstaben der Datei trägt.[2]

Diese Dateien enthalten Binärdaten, welche nicht mit einem Texteditor bearbeitet werden können. Falls der Inhalt dieser Dateien angezeigt werden soll, geschieht dies mit dem Kommando *infocmp*. Die Ausgabe von *infocmp* kann mit einem Editor modifiziert werden und anschließend mit *tic*, dem *terminfo compiler* in eine Binärdatei zurückverwandelt werden.

Beispiel einer terminfo-Gerätebeschreibung:

```
pische@pisch:~ > infocmp sun
# Reconstructed via infocmp from file: /usr/share/terminfo/s/sun sun|sun1|sun2|Sun Microsystems Inc.
workstation console,
am, km, msgr,
cols#80, lines#34,
bel=^G, clear=^L, cr=^M, cub1=^H, cud1=^J, cuf1=\E[C,
cup=\E[%i%p1%d;%p2%dH, cuu1=\E[A, dch=\E[%p1%dP,
dch1=\E[P, dl=\E[%p1%dM, dl1=\E[M, ed=\E[J, el=\E[K, ht=^I,
ich=\E[%p1%d@, ich1=\E[@, il=\E[%p1%dL, il1=\E[L, ind=^J,
```

[2] Zum Beispiel befindet sich die Gerätebeschreibung für ein VT100 Terminal in der Datei */usr/share/lib/terminfo/v/vt100*.

```
kb2=\E[218z, kbs=^H, kcub1=\E[D, kcud1=\E[B, kcuf1=\E[C,
kcuu1=\E[A, kdch1=\177, kend=\E[220z, kf1=\E[224z,
kf10=\E[233z, kf11=\E[234z, kf12=\E[235z, kf2=\E[225z,
kf3=\E[226z, kf4=\E[227z, kf5=\E[228z, kf6=\E[229z,
kf7=\E[230z, kf8=\E[231z, kf9=\E[232z, khome=\E[214z,
knp=\E[222z, kopt=\E[194z, kpp=\E[216z, kres=\E[193z,
kund=\E[195z, rev=\E[7m, rmso=\E[m, rs2=\E[s,
sgr=\E[0%?%p6%t;1%;%?%p2%t;4%;%?%p1%p3%|%t;7%;m,
sgr0=\E[m, smso=\E[7m, u8=\E[1t, u9=\E[11t,
pische@pisch:~ >
```

Die *termcap*-Datenbasis existiert aus Kompatibilitätsgründen, damit Applikationen, welche noch damit arbeiten, ablauffähig sind. Die Geräteinformationen stehen alle gesammelt in einer Datei mit dem Namen */etc/termcap*.

11.3 Terminals und Modems

Die Konfiguration von Terminals und Modems kann 'manuell' mit den SAF-Kommandos *sacadm*, *pmadm* und *ttyadm* erfolgen, oder bequemer mit Hilfe von *admintool*. *admintool* bietet eine grafische Oberfläche.

Das Kommando tip

/usr/bin/tip wird verwendet, um eine Verbindung über eine serielle Schnittstelle zu einem fernen System herzustellen. *tip* benötigt Konfigurationdaten in der Datei */etc/remote*. Möchte man einen *headless server* (ein Server ohne Konsole) installieren, so kann mit Hilfe von *tip* und eines Nullmodem-Kabels von einem anderen Rechner aus gearbeitet werden.

tip [-*baudrate*] *hostname* | *phone-number*

hostname Der *hostname* bezieht sich auf einen Eintrag, der in der Datei */etc/remote* vorgenommern werden muss.

Beispiele:

```
# tip mercury                          # Verbindung zum Rechner mercury herstellen
# tip -9600 mercury                              # Verbindungsaufbau mit 9600 Baud
# tip 12344321              # Verbindungsaufbau zur Gegenstelle mit der Telefonnummer xxx
```

Die Datei /etc/remote

Die Datei */etc/remote* enthält Informationen für das Kommando *tip*, um Verbindungen über eine serielle Schnittstelle zu einem Remotesystem aufzubauen. Jeder Eintrag besteht aus mehreren Feldern, welche durch Doppelpunkt *':'* voneinander getrennt sind.

Das erste Feld enthält einen Rechnernamen und optional einen durch das Zeichen '|' getrennten Aliasnamen. Anschließend folgen die sogenannten *device functions*. Diese beschreiben die Verbindung:

Device Functions:

at Autocall Unit Type

br Baudrate

du kennzeichnet eine *dial up line*

dv Gerätedevice der seriellen Schnittstelle

el Zeichen, welche ein *end-of-line* markieren

ie Zeichenkette die bei der Eingabe *end-of-file* markieren

oe Zeichenkette die bei der Ausgabe *end-of-file* markieren

pn phone-number – die Telefonnummer der gewünschten Gegenstelle; Bei der Angabe von mehreren alternativen Nummern, müssen diese durch das Zeichen '|' getrennt werden.

tc Zeiger auf die Fortsetzung eines Eintrages. Damit kann Schreibarbeit eingespart werden und die Konfigurationsdatei kleiner gehalten werden. Für mehrere Verbindungen gültige Werte können einmal angegeben werden und von beliebigen Einträgen 'angesprungen' werden.

Beispiel der Datei /etc/remote:

```
# The next 17 lines are for the PCMCIA serial/modem cards.
#
pc0:\
        :dv=/dev/cua/pc0:br#38400:el=^C^S^Q^U^D:ie=%$:oe=^D:nt:hf:
pc1:\
        :dv=/dev/cua/pc1:br#38400:el=^C^S^Q^U^D:ie=%$:oe=^D:nt:hf:
pc2:\
        :dv=/dev/cua/pc2:br#38400:el=^C^S^Q^U^D:ie=%$:oe=^D:nt:hf:
pc3:\
        :dv=/dev/cua/pc3:br#38400:el=^C^S^Q^U^D:ie=%$:oe=^D:nt:hf:
pc4:\
        :dv=/dev/cua/pc4:br#38400:el=^C^S^Q^U^D:ie=%$:oe=^D:nt:hf:
pc5:\
        :dv=/dev/cua/pc5:br#38400:el=^C^S^Q^U^D:ie=%$:oe=^D:nt:hf:
pc6:\
        :dv=/dev/cua/pc6:br#38400:el=^C^S^Q^U^D:ie=%$:oe=^D:nt:hf:
pc7:\
        :dv=/dev/cua/pc7:br#38400:el=^C^S^Q^U^D:ie=%$:oe=^D:nt:hf:
cuab:dv=/dev/cua/b:br#2400
dialup1|Dial-up system:\
        :pn=2015551212:tc=UNIX-2400:
sera:\
        :dv=/dev/term/a:br#9600:el=^C^S^Q^U^D:ie=%$:oe=^D:
sera19:\
        :dv=/dev/term/a:br#19200:el=^C^S^Q^U^D:ie=%$:oe=^D:
serb:\
        :dv=/dev/term/b:br#9600:el=^C^S^Q^U^D:ie=%$:oe=^D:
hardwire:\
        :dv=/dev/term/b:br#9600:el=^C^S^Q^U^D:ie=%$:oe=^D:
tip300:tc=UNIX-300:
tip1200:tc=UNIX-1200:
tip0|tip2400:tc=UNIX-2400:
tip9600:tc=UNIX-9600:
tip19200:tc=UNIX-19200:
UNIX-300:\
        :el=^D^U^C^S^Q^O@:du:at=hayes:ie=#$%:oe=^D:br#300:tc=dialers:
UNIX-1200:\
        :el=^D^U^C^S^Q^O@:du:at=hayes:ie=#$%:oe=^D:br#1200:tc=dialers:
UNIX-2400:\
        :el=^D^U^C^S^Q^O@:du:at=hayes:ie=#$%:oe=^D:br#2400:tc=dialers:
UNIX-9600:\
        :el=^D^U^C^S^Q^O@:du:at=hayes:ie=#$%:oe=^D:br#9600:tc=dialers:
UNIX-19200:\
        :el=^D^U^C^S^Q^O@:du:at=hayes:ie=#$%:oe=^D:br#19200:tc=dialers:
VMS-300|TOPS20-300:\
        :el=^Z^U^C^S^Q^O:du:at=hayes:ie=$@:oe=^Z:br#300:tc=dialers:
VMS-1200|TOPS20-1200:\
```

```
48          :el=^Z^U^C^S^Q^O:du:at=hayes:ie=$@:oe=^Z:br#1200:tc=dialers:
49  dialers:\
50          :dv=/dev/cua/b:
51  ──────────────────────────────────────────────────────────────────────
52  The attributes are:
53
54  dv      device to use for the tty
55  el      EOL marks (default is NULL)
56  du      make a call flag (dial up)
57  pn      phone numbers (@=>'s search phones file; possibly taken from
58                                 PHONES environment variable)
59  at      ACU type
60  ie      input EOF marks (default is NULL)
61  oe      output EOF string (default is NULL)
62  cu      call unit (default is dv)
63  br      baud rate (defaults to 300)
64  fs      frame size (default is BUFSIZ) -- used in buffering writes
65            on receive operations
66  tc      to continue a capability
```

12 Das NFS Dateisystem

Eigenschaften und Funktionen von NFS

- Dateien können zentral verwaltet werden und sind für alle sofort aktuell.
- Zentral liegende Software ist für alle zugänglich. Updates sind nur einmal durchzuführen.
- Daten scheinen sich am lokalen System zu befinden und können gleich behandelt werden.
- Einfaches Handling
- NFS ist ein Standard
- Gemeinsame Verzeichnisse mit Nur-Lese-Zugriff können von vielen Stationen verwendet werden und belegen nur einmal Platz.
- Home-Directories können zentral liegen und sind somit von verschiedenen Stationen aus erreichbar.
- Beschreibbare Verzeichnisse können freigegeben und mit bestimmten Zugriffsrechten versehen werden.
- Workstations ohne lokaler Platte können die benötigten Dateisysteme sowie Swap übers Netz von einem Remotesystem beziehen.
- Workstations können von einen Remotesystem mit unterschiedlicher Version bzw. Kernelarchitektur gebootet werden.

12.1 Ressourcen

NFS Server	NFS Client
Prozesse	
mountd	
nfsd	
statd	statd
lockd	lockd
Dateien	
/etc/dfs/dfstab	/etc/vfstab
/etc/dfs/fstypes	/etc/mnttab
/etc/dfs/sharetab	
/etc/rmtab	
Kommandos	
share	mount
unshare	umount
shareall	mountall
unshareall	umountall
dfshares	dfshares
dfmounts	dfmounts

12.1.1 NFS Dämonen

Für die Funktion von NFS sind mehrere Dämon-Prozesse nötig.

12.1.1.1 mountd – Mound Dämon

Der NFS-Client, der mittels Kommando *mount* eine Mount-Anforderung stellt, erhält von *mountd (/usr/lib/nfs/mountd)* ein *File Handle*[1] zurück. Das File Handle wird zusammen mit anderen Informationen von *mount* in die Datei */etc/mnttab* geschrieben. Serverseitig erfolgt ein Eintrag in die Datei */etc/rmtab* durch den Prozess *mountd*. Der Inhalt der Datei */etc/rmtab* wird beim Start von *mountd* gelöscht.

12.1.1.2 nfsd – NFS Server Dämon

Wenn ein Prozess des NFS-Client auf eine Datei am NFS-Server zugreifen will, so gelangt diese Anforderung zum NFS Server Dämon */usr/lib/nfs/nfsd*. Dieser Prozess führt dann den gewünschten Dateizugriff durch und liefert die Daten an den Client zurück. Die NFS Server Dämonen werden beim Systemstart im Skrip */etc/init.d/nfs.server* gestartet. Hier wird auch die Anzahl an nfsd-Threads festgelegt. Diese Anzahl bestimmt, wieviele Anforderungen gleichzeitig abgearbeitet werden können.

Ob die NFS-Dämonen gestartet werden oder nicht, hängt davon ab, ob in der Datei */etc/dfs/dfstab* Freigaben von Verzeichnissen eingetragen sind oder nicht.

[1] Ein File Handle ist ein eindeutiger Bezeichner für eine Datei oder ein Verzeichnis. Es beinhaltet Inodenummer und Gerätedevicenummer.

12.1.1.3 statd und lockd Dämon

Die beiden Dämonprozesse */usr/lib/nfs/statd* und */usr/lib/nfs/lockd* laufen sowohl am Server als auch am NFS-Client. Sie werden beim Hochfahren des Systems durch das Skript */etc/init.d/nfs.client* gestartet, wenn der Runlevel 2 erreicht wird. Die beiden Prozesse arbeiten zusammen und bearbeiten Sperrmechanismen (locking services) sowie Situationen wie *crash* und *recovery*.

12.1.2 Datei /etc/dfs/dfstab

Diese Datei enthält *share* Kommandos, welche Ressourcen per NFS freigeben. Diese Kommandos werden in den folgenden Fällen aufgerufen:

- Verwendung des Kommandos *shareall* (Voraussetzung: root startet das Kommando; NFS-Dämon läuft bereits)
- Beim Erreichen des Run-Level 3 wird das Skript */etc/init.d/nfs.server* aufgerufen (dieses enthält auch das Kommando *shareall*). Dieses Skript startet auch die benötigten Dämonen, falls Einträge in */etc/dfs/dfstab* vorgefunden werden.

Beispiel der Datei */etc/dfs/dfstab*:

```
1
2  #       Place share(1M) commands here for automatic execution
3  #       on entering init state 3.
4  #
5  #       Issue the command '/etc/init.d/nfs.server start' to run the NFS
6  #       daemon processes and the share commands, after adding the very
7  #       first entry to this file.
8  #
9  #       share [-F fstype] [ -o options] [-d "<text>"] <pathname> [
       resource]
10 #       .e.g,
11 #       share  -F nfs  -o rw=engineering  -d "home dirs"  /export/home2
12 share  -F nfs  -o rw=sunpc:root -d "Software"  /soft
```

12.1.3 Kommandos:

12.1.3.1 share Kommando

share stellt Dateiressourcen zur Verfügung, sodass diese von einem fernen Rechner gemountet werden können.

Syntax:

share

Ohne Optionen werden die aktuellen Freigaben aufgelistet.

share [-F FSType] [-o options] [-d description] Pfadname

-F FSType Legt den Filesystem-Typ fest. Diese Option ist nicht zwingend nötig, da *nfs* der Standardtyp ist.

-o options Optionen definieren die Zugriffsrechte der Clients auf die freigegebene Ressource. Optionen werden durch Kommas voneinander getrennt.

ro[=*access-list*] In der access-list angegebene Clients haben Nur-Leserecht. Ohne Angabe einer access-list (ein oder mehrere durch Doppelpunkt voneinander getrennte Clients) gilt Nur-Leserecht für alle.

rw[=*access-list*] Der Client darf lesen und schreiben. Allerdings gelten die üblichen Zugriffsrechte der Dateien weiterhin.

root=*client* Superuser des Rechners *client* hat Superuserrechte für die freigegebene Ressource

anon=*n* Ein unbekannter User bekommt die effektive UID = *n*. Standardmäßig erhalten unbekannte User die effektive UID von *nobody*. Bei Angabe des Wertes '-1' wird der Zugriff verweigert.

public Die Ressource wird per WebNFS freigegeben (siehe Seite 130).

index=*index.html* Im Zusammenhang mit der Option *public* kann eine Einstiegsseite (das muss eine Datei im HTML-Format sein) angegeben werden. Ohne Angabe diese Datei wird eine Dateibaumansicht generiert.

-d description Beliebiger Text, welcher die Freigabe beschreibt. Dieser Text wird bei Verwendung des Kommandos *share* ohne Optionen oder *dfshares* angezeigt.

Pfadname Angabe der Ressource, welche freigegeben werden soll.

Access-Listen können folgende Formate haben:

client

***client*:*client*:...** Angabe von Hostnamen bzw. IP-Adressen

@network Angabe von Netzwerken (z.Bsp.: @192.168.100 oder @mynet.com[2]).

.domain Beginnt der Name mit einem Punkt, so wird der Name als DNS-Domäne interpretiert.

netgroup_name Erlaubt den Zugriff von einer konfigurierten NIS- bzw. NIS+-Netgroup

Obige Formate können beliebig kombiniert werden - sie müssen jeweils durch Komma voneinander getrennt werden.

Freigaben, welche ohne dezidiertem Zugriffsrecht angegeben werden, ermöglichen standardmäßig Lese- und Schreibrecht.

[2] Bei der Angabe von Netzwerknamen, muss dieser in der Datei */etc/networks* aufgelistet sein.

12.1.3.2 unshare, unshareall und shareall

Syntax:

unshare [-F FSType] *Pfadname*

unshareall [-F FSType]

shareall [-F FSType]

unshare hebt die Freigabe einer ausgewählten Ressource auf. Hingegen betrifft *unshareall* alle freigegebenen Ressourcen.

shareall gibt alle in der Datei */etc/dfs/dfstab* eingetragenen Ressourcen frei.

12.1.3.3 dfshares Kommando

dfshares zeigt die freigegebenen Ressourcen in einem Netz bzw. eines Servers an.

Syntax:

dfshares [-F *FSType*] [*servername*]

servername Name des Servers, dessen Shares aufgelistet werden sollen. Wird kein Rechnernamen angegeben, so werden alle Hosts im Netz nach NFS-Shares durchsucht

Beispiel:

```
root@sunlicht:/etc/dfs > dfshares
RESOURCE       SERVER   ACCESS TRANSPORT
sunlicht:/soft sunlicht   -       -
```

12.1.3.4 dfmounts Kommando

dfmounts zeigt alle Ressourcen an, welche von einem Client gemountet wurden.

Syntax:

dfmounts [-F FSType] [*Clientname*]

Clientname Nur Ressource, welche vom Rechner '*Clientname*' gemountet wurden, werden angezeigt. Ohne Angabe eines Rechnernamens, werden alle gemounteten Ressourcen angezeigt.

Beispiel:

```
root@sunlicht:/etc/dfs > dfmounts
RESOURCE SERVER   PATHNAME CLIENTS
-      sunlicht /soft    pisch
root@sunlicht:/etc/dfs >
```

12.1.3.5 mount Kommando

/sbin/mount hängt ein Dateisystem oder eine NFS-Ressource in den lokalen Dateisystembaum.[3]

Syntax:

mount [-F nfs] [-o *options*] *server*:*pathname mount-point*

-F nfs Der Dateisystemtyp muss normalerweise nicht angegeben werden, da '*nfs*' ohnehin der Standardtyp für Netzwerkfilesysteme ist.

-o options Werden mehrere Optionen angegeben, müssen diese durch Komman voneinander getrennt werden (ohne Leerzeichen!).

rw|ro Zugriff read/write bzw. read only

bg|fg Schlägt der erste Mount-Versuch fehl, so werden folgende Versuche im Hintergrund '*bg*'(background) bzw. im Vordergrund '*fg*'(foreground) durchgeführt. Standard ist '*fg*'.

soft|hard Mit der Option '*soft*' erfolgt bei einem fehlgeschlagenen Mount-Versuch eine Fehlermeldung und abgebrochen. Die Option '*hard*' (Standard) bewirkt, dass so lange versucht wird, die Ressource zu mounten, bis dies gelingt.[4]

intr|nointr *intr* ermöglicht den Abbruch eines Mountversuches mit der Tastatur (Standard), falls die Option *hard* aktiv ist.

suid|nosuid Ermöglicht / verhindert die *setuid*-Ausführung. Standard ist setuid-Ausführung. (siehe Seite 39)

timeo=*n* Timeout in $\frac{1}{10}$Sekunden. Der Standard-Timeout beträgt 1,1 Sekunden.

retry=*n* Anzahl der Wiederholversuchen bei NFS-Mount (Standard =10 000)

retrans=n Anzahl der Sendewiederholungen, falls Verbindung per UDP (Standard = 5). Für TCP-Verbindungen ist der Wert belanglos.

server*:*pathname Name oder IP-Adresse des NFS-Servers und anschließend, durch Doppelpunkt getrennt, der Pfadname der Freigabe.

mount-point Verzeichnis, in das die NFS-Ressource eingehängt werden soll.

Empfohlene Optionen für unterschiedliche Verwendungen von NFS-Shares:

NFS Filesystem	Read-write / Read-only	System Startup	Server Crash	Interrupt	Security
/usr	ro	fg	hard	nointr	nosuid
/export/home/*xy*	rw	fg	hard	intr	suid
/opt/*appl*	ro	bg	soft	-	nosuid

[3] Hier wird nur die Syntax für das Einhängen einer NFS-Ressource behandelt. Details siehe Online-Manual und Seite 84.

[4] Um zu verhindern, dass ein System hängt, falls NFS-Ressourcen nicht gemountet werden können, empfiehlt es sich die Optionen *bg* und *soft* zu verwenden.

12.1.3.6 umount, mountall und umountall

Syntax:

umount [-F FSType] *server*:*pathname* | *mountpoint*

mountall [-r] [-F FSType]

-r Mountet alle NFS-Ressourcen, die in der Datei */etc/vfstab* eingetragen sind.

umountall [-r] [-F FSType]

-r Hängt alle gemounteten NFS-Dateisysteme ab

12.1.4 Kochrezept: Freigabe einer NFS-Ressource

NFS-Server ('serverhost')

1. Editieren der Datei */etc/dfs/dfstab*.
   Bsp.: share -F nfs -o ro=clienthost /usr/share/man
2. Falls dies die erste Freigabe in der Datei /etc/dfs/dfstab ist, so müssen noch die nötigen Dämonen gestartet werden. Das geschieht durch folgenden Aufruf:
   ```
   # /etc/init.d/nfs.server start
   ```
   Waren die Dämonen schon gestartet, so kann die zusätzliche Freigabe einfach durch den Aufruf von **shareall** freigegeben werden.
3. Kontrolle, ob die Freigabe tatsächlich erfolgte:
   ```
   # dfshares
   ```

NFS-Client ('clienthost')

1. Eintrag in die Datei */etc/vfstab* vornehmen, um ein automatisches Mounten beim Systemstart zu bewirken:
   ```
   #device         device        mount           FS      fsck    mount   mount
   #to mount       to fsck       point           type    pass    at boot options
   #
   #/dev/dsk/c1d0s2 /dev/rdsk/c1d0s2 /usr          ufs     1       yes     -
   serverhost:/usr/share/man -   /usr/share/man  nfs     -       yes     soft,bg
   ```
2. Restart des Systems oder besser **mountall -r** aufrufen, um alle Remote-Dateisysteme zu mounten.

Um gleichartige Ressourcen, welche von verschiedenen Servern freigegeben werden automatisch wahlweise mounten zu können, muss der Eintrag in der Datei */etc/vfstab* wie folgt aussehen. Im Beispiel haben 3 Server mit den Hostnamen *A, B* und *C* alle das Verzeichnis */usr/share/man* freigegeben. Der Client möchte die Ressource vom erstbesten Server mounten ohne sich Gedanken zu machen, ob ein Server hochgefahren ist, oder nicht:

```
A(0),B(2),C(1),/usr/share/man   -   /usr/share/man   nfs   -   yes   -
```

Die Ziffern in Klammer hinter dem Servernamen geben die Priorität an, welche entscheidet welcher Server bevorzugt wird.

12.2 WebNFS

WebNFS ist kein eigenes Protokoll, sondern lediglich ein Dienst, der den Zugriff auf eine NFS-Ressource per Web-Browser ermöglicht.

Der Zugriff erfolgt mittels Web-Browser wie z.Bsp. Netscape, Internet Explorer etc. durch die Eingabe der URL folgenden Formates:

nfs://*serverhost*/ # Zugriff auf *public file handle* des Servers *serverhost*

nfs://*serverhost*/*dir_name* # Zugriff relativ zu *public file handle*

nfs://*serverhost*//*dir_name* # Zugriff mit absolutem Pfadnamen

EINSCHRÄNKUNG: Es kann auf einem System nur eine Freigabe per WebNFS gemacht werden!

Die Freigabe per WebNFS erfolgt durch die spezielle Option *public* des Kommandos *share*.

Beispiel:

```
# share -F nfs -o public,index=index.html /export/directory
```

Im Beispiel wird die HTML-Datei *index.html* als Einstiegsseite verwendet.

ACHTUNG: WebNFS verwendet das TCP-Port 2049! Dies ist von Bedeutung, wenn eine Firewall auf der Verbindungsstrecke existiert.

13 Automount

Eigenschaften von Automount:

- Dateisysteme werden bei Bedarf gemountet –> Ressourcen werden effizienter genützt
- Dateisysteme werden automatisch abgehängt, wenn längere Zeit kein Zugriff darauf erfolgt (Standard: 10 Minuten).
- Automount belastet die Serverseite kaum zusätzlich. Die Administration von NFS-Mounts kann mit Hilfe von Name Service zentralisiert werden.
- Server Redundanz ermöglicht den Zugriff auf dasselbe Dateisystem über mehr als einen Rechner. Mittels automount kann der Zugriff über den am wenigsten ausgelasteten Server erfolgen.

13.1 Arbeitsweise von automount

autofs Dateisysteme werden in den sogenannten *automount maps* im Verzeichnis */etc* definiert. *autofs* ist ein Bestandteil des Solaris-Kernel und überwacht Zugriffe auf Verzeichnisse, welche als Mountpoints definiert wurden. Sobald der Kernel eine Anforderung für eine *autof*s-Ressource erkennt, wird das durch *autofs* abgefangen und der Dämon *automountd* durch eine Nachricht veranlasst, die erforderliche Ressource zu mounten.

ACHTUNG! Dateisysteme, welche durch *autofs* automatisch gemountet werden, dürfen nicht manuell 'mounted' oder 'unmounted' werden. Es kann zu Inkonsistenzen kommen, welche u.U. nur durch Reboot beseitigt werden können.

13.1.1 Das automount Kommando

/usr/sbin/automount wird während des Systemstarts durch das Skript */etc/init.d/autofs* aufgerufen. *automount* liest Einträge aus der *Master Mapfile /etc/auto_master* um anschließend die sogenannten *autofs mounts* zu initialisieren. Diese sind spezielle *mount points* unter denen später bei Anforderung Dateisysteme gemountet werden. Sie werden auch *trigger nodes* genannt.

Nachdem *automount* beim Systemstart einmals aufgerufen wird, um *autofs* zu initialisieren, wird das Kommando verwendet, wenn Änderungen an den Konfigurationsdateien vorgenommen werden. Ob *automount* nach einer Änderung der *Automount Maps* erforderlich ist oder nicht, hängt von der Art der Änderung (hinzufügen, löschen oder ändern) und vom Typ des betroffenen Map-Eintrages (auto_master, direct map, indirect map) ab. Da es aber keine negativen Auswirkungen hat, *automount* aufzurufen, ohne dass dies erforderlich wäre, empfiehlt es sich dies nach jeder Änderung zu tun.

automount [-t duration] [-v]

-t duration Zeitdauer, wie lange eine Ressource gemountet bleiben soll, ohne dass ein Zugriff erfolgt.

-v Ausgabe von Information

13.1.2 Der automountd Dämon

Der Dämonprozess *automountd* ist ein RPC Server, das – durch das *autofs* Kernelfilesystem gesteuert – das Mounten und Unmounten von Dateisystemen durchführt. Dieser Prozess wird während des Systemstarts durch das Skript */etc/init.d/autofs* gestartet.

automountd ist völlig unabhängig vom *automount* Kommando. Dadurch ist es möglich, Änderungen in den für *autofs* relevanten Konfigurationsdateien durchzuführen, ohne anschließend *automountd* erneut starten zu müssen.

Zusammenfassung der *autofs* Funktionen:

1. *autofs* fängt Zugriffe auf ein Dateisystem, welches automatisch gemountet werden muss ab.
2. *autofs* sendet ein Nachricht an den *automountd* Dämon.
3. *automountd* ermittelt anhand der *automount maps* das erforderliche Dateisystem und mountet es.
4. *autofs* veranlasst die Fortsetzung des abgefangenen Zugriffes.
5. *automountd* führt nach einer gewissen Zeit der Inaktivität einen ’unmount’ durch.

13.2 Typen von autofs maps

Automount maps sind ASCII, NIS oder NIS+ Dateien. Es gibt 3 verschiedene Arten von *autofs maps*.

13.2.1 Master Map

Die Datei */etc/auto_master* listet alle *mount points*, welche *autofs* überwachen muss.

Standardeinträge der Datei */etc/auto_master*:

```
1 # Master map for automounter
2 #
3 +auto_master
4 /net            -hosts          -nosuid,nobrowse
5 /home           auto_home       -nobrowse
6 /xfn            -xfn
```

Der Eintrag *+auto_master* ist ein Verweis auf eine externe NIS oder NIS+ Master Map. Falls eine solche Map existiert, wird diese anstelle dieser Zeile eingefügt.

Map-Einträge bestehen aus bis zu 3 durch Leerzeichen voneinander getrennte Felder:

MOUNT-POINT MAP-NAME [MOUNT-OPTIONS]

mount-point Der absolute Pfadname eines Verzeichnisses. Existiert das Verzeichnis noch nicht, so wird es automatisch erzeugt. Steht an dieser Stelle '/-', so weist dies auf einen *Direct Map* Eintrag hin. Die nötigen Informationen zum Mounten stehen in der zugehörigen Mapdatei (2. Feld).

map-name Der Name einer direkten oder indirekten Map. Diese Maps sind Verweise auf Mounting-Information.
Der Eintrag '*-hosts*' bedeutet, dass alle freigegebenen Ressourcen von Rechnern, die in */etc-/inet/hosts* bzw. NIS oder NIS+ eingetragen sind, im Verzeichnis */net* als Unterverzeichnis mit dem Rechnernamen des NFS-Servers zu finden sind.
Beispiel: Die freigegebenen Verzeichnisse eines NFS-Server, der in der Datei */etc/inet/hosts* unter dem Namen *nfshost* eingetragen ist, werden von *autofs* im Verzeichnis mit dem Namen */etc/nfshost* gemountet.
'*-xfn*' bezieht sich auf den Federated Name Service (FNS).

mount-options Dieses Feld ist optional. Hier können die vom Kommando *mount* bekannten Optionen angegeben werden.

13.2.2 Direct Map

Ein Eintrag in der Datei */etc/auto_master*, dessen erstes Feld '/-' lautet, weist auf eine *Direct Map* hin. Die vollständigen Pfadnamen stehen in einer Datei, deren Name im 2. Feld zu entnehmen ist.

Beispiel eines *Direct Map* Eintrages in die Datei */etc/auto_master*:

```
# Master map for automounter
#
```

```
+auto_master
/net            -hosts          -nosuid,nobrowse
/home           auto_home       -nobrowse
/xfn            -xfn
/-              auto_direct
```

Die zugehörige Datei, welche die *Direct Map* beschreibt, hat den Namen */etc/auto_direct* (siehe 2. Feld). Diese Datei muss manuell erzeugt werden.

Beispiel:

```
# Direct Map Datei 'auto_direct'
#
/usr/share/man -ro,soft sun,moon,mars:/usr/share/man
```

Im Beispiel wird bei Anforderung (Verzeichniswechsel mit *cd* nach */usr/share/man*) von einem der Rechner *sun*, *moon* oder *mars* das freigegebene Verzeichnis mit dem Namen */usr/share/man* automatisch gemountet.[1]

Die Datei besteht aus 3 Feldern:

mount_point Voller Pfadname des Mountverzeichnisses

options Optionen für das Kommando *mount*.

location Share-Name in dem Format: *servername*:*pathname*.

13.2.2.1 Erzeugen einer Direct Map:

1. Eintrag in die Datei */etc/auto_master:*
   /- auto_direct
2. Erzeugen der Datei */etc/auto_direct* mit den gewünschten Einträgen:
   /usr/share/man -ro *serverhost*:/usr/share/man
3. Änderungen aktivieren:
   `# automount -v`

13.2.3 Indirect Map

In den Standardeinträgen der Datei */etc/auto_master* befindet sich bereits ein Verweis auf eine Indirect Map:

```
# Master map for automounter
#
+auto_master
/net            -hosts          -nosuid,nobrowse
/home           auto_home       -nobrowse
/xfn            -xfn
/-              auto_direct
```

Die dazugehörige Datei */etc/auto_home* enthält standardmäßig jedoch nur folgenden Eintrag:

[1] Die Angabe von mehreren Servern, von welchen wahlweise gemountet wird, funktioniert nur, wenn die Freigaben an den Servern '*read only*' erfolgen.

```
# Home directory map for automounter
#
+auto_home
```

Dieser Standardeintrag würde eine NIS oder NIS+ Konfigurationsdatei verwenden.

Beispiele für mögliche Einträge in die Indirect Map Datei */etc/auto_home*:

```
# Home directory map for automounter
#
+auto_home
testuser     sun:/export/home/testuser
*            mars:/export/home/&
```

Der erste zusätzliche Eintrag hat folgende Bedeutung: Wechselt man in das Verzeichnis */home/testuser*, so wird automatisch der NFS-Share */export/home/testuser* des Servers mit dem Rechnernamen *sun* gemountet. Das Mountverzeichnis */home* steht im 1. Feld der Master Map Datei */etc/auto_master*.

Im zweiten zusätzlichen Eintrag, werden bei Anforderung jeweils die Homeverzeichnisse der User des Rechners *mars* nach */home* gemountet. Die beiden Zeichen '*' und '&' werden ersetzt durch den Namen der Homeverzeichnisse.

13.2.3.1 Erzeugen einer Indirect Map:

1. Editieren der Master Map Datei */etc/auto_master*:
   /services auto_patch
2. Erzeugen und editieren der Indirect Map Datei */etc/auto_patch*:
   patch mars:/export/patch
3. Änderungen aktivieren:
   # **automount -v**

14 RAM-basierte File Systeme

RAM-based File Systems sind Dateisysteme, welche die Daten im Hauptspeicher halten, aber im Dateibaum sichtbar sind, so als ob sie sich auf Platte befinden würden. Sie werden auch *Pseudo File Systems* genannt. Diese Dateisysteme werden aus Performancegründen verwendet. Sie gestatten den Zugriff mit denselben Systemaufrufen, wie sie für sonstige Dateien üblich sind.

14.1 /proc File System

Das */proc* Dateisystem ist ein Pseudodateisystem. Es wird im Verzeichnis /proc eingehängt und enthält darunter für jeden Prozess im System ein Unterverzeichnis dessen Name der PID des Prozesses entspricht. Innerhalb dieses Verzeichnisses befinden sich Dateien, welche den Zustand des Prozesses beschreiben. Programme können mit den Standard-Systemaufrufen (open(), close(), read(), write()) darauf zugreifen.

Das Dateisystem /proc wird während des Systemstarts aufgrund eines Eintrages in der Datei */etc/vfstab* gemountet.

Auszug aus der Datei */etc/vfstab*:

```
#device         device          mount    FS    fsck  mount   mount
#to mount       to fsck         point    type  pass  at boot options
#
/proc           -               /proc    proc  -     no      -
```

14.2 swapfs File System

Das Modell des *virtuellen Speichers* ermöglicht es, Programmen einen größeren Speicher vorzugaukeln als tatsächlich physikalisch vorhanden ist. Jeder einzelne Prozess hat theoretisch die Möglichkeit von diesem 'großen' Speicher Gebrauch zu machen. Reicht der physikalische Speicher nicht aus, so werden Daten auf die Platte ausgelagert. Dieser Plattenbereich wird *Swap* genannt.

Die minimale Größe wird durch 2 Parameter bestimmt:

- Um eventuelle Dumps bei einem Systemabsturz für eine Diagnose zu erhalten, muss der Swapbereich wenigsten gleich groß wie die Hauptspeichergröße sein. Reicht die Größe des Swap nicht aus, um diese Daten vorübergehend aufzunehmen, so steht nach einem Absturz kein Dump zur Verfügung.
- Die Größe von RAM und Swap zusammen muss wenigstens ausreichen, um System- und Anwenderprozessen Platz zu bieten. Je weniger Daten aufgrund von Hauptspeichermangels auf den Swap-Bereich ausgelagert werden müssen, desto performanter wird das System.

Üblicherweise werden für den Swapbereich ganze Slices reserviert. Durch den Eintrag in die Datei */etc/vfstab* wird der Swap beim Startup des Systems automatisch eingebunden.

Auszug aus der Datei */etc/vfstab*:

```
#device         device         mount    FS     fsck  mount   mount
#to mount       to fsck        point    type   pass  at boot options
#
/dev/md/dsk/d3  -              -        swap   -     no      -
```

Es ist aber auch möglich, eine Datei in einem Dateisystem als Swap zu verwenden (wenn auch nicht sehr performant und deshalb nur zu Testzwecken oder vorübergehend zu empfehlen). Das Hinzufügen eines *Swap Files* erfolgt auf folgende Weise:

```
# swap -s                                          # Übersicht über Swapbereich ausgeben
# swap -l                                             # Details zum Swapbereich ausgeben
# df -k                                             # Feststellen, wo genügend Platz ist
# mkfile 20m /export/swap          # Eine 20 MByte große Datei namens /export/swap erzeugen
# swap -a /export/swap                                # Datei zum Swapbereich hinzufügen
# swap -l                                  # Kontrolle, ob nun ausreichen Swapspace vorhanden
```

swap Kommando:

swap [-l|-s] [-a|-d *swapfile*]

-l Listet Swapspace detailliert auf

-s Zusammenfassung des vorhandenen Swapspace

-a *swapfile* Hinzufügen von Swapspace

-d *swapfile* Entfernen von Swapspace; Swapspace kann während des Betriebes entfernt werden.

14.3 tmpfs File System

tmpfs ist ein Dateisystem, welches das *VM Subsystem* (*virtual memory subsystem*) verwendet. Temporärdateien werden dadurch im schnellen Hauptspeicher abgelegt, solange Platz ist. Andernfalls werden sie in den Swapspace geschrieben. Um das zu realisieren, enthält die Datei */etc/vfstab* einen speziellen Eintrag.

Auszug aus der Datei */etc/vfstab*:

```
#device         device         mount    FS    fsck  mount   mount
#to mount       to fsck        point    type  pass  at boot options
#
swap            -             /tmp      tmpfs -     yes     -
```

Da die Temporärdateien im Virtual Memory (/tmp) abgelegt werden, sind diese nach einem Reboot nicht mehr vorhanden.

14.4 fdfs File System

fdfs (File Descriptor File System) ist ein Pseudodateisystem, welches eine Dateisystem-Schnittstelle zu Filedeskriptoren zur Verfügung stellt. Achtung!: Fehlt der Eintrag für *fdfs* in der Datei */etc/vfstab*, so ist kein Systemstart mehr möglich!

Auszug aus der Datei */etc/vfstab*:

```
#device         device         mount    FS    fsck  mount   mount
#to mount       to fsck        point    type  pass  at boot options
#
fd              -              /dev/fd  fd    -     no      -
```

14.5 CacheFS File System

Das *Cache File System* kann verwendet werden, um die Performance auf langsame Dateisysteme zu *cachen* und somit zu beschleunigen. Die Verwendung ist vor allem sinnvoll für Zugriffe auf CD-ROM oder Remote Filesysteme wie NFS.

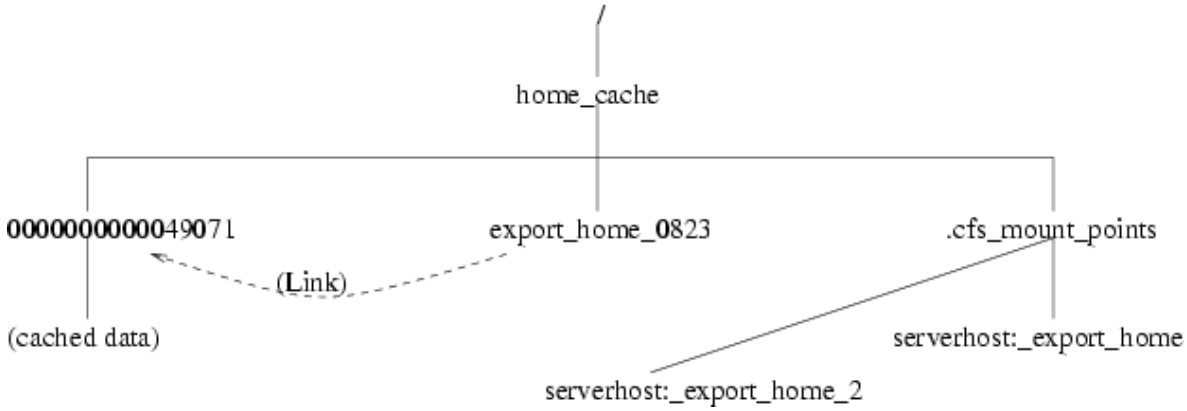
Begriffserklärung:

Back Filesystem Originaldateisystem wie z.Bsp. CD-ROM, NFS. Das ist jenes Dateisystem, welches rel. langsam ist und deshalb durch 'caching' beschleunigt werden soll.

Front Filesystem Das ist das gemountete Dateisystem. Dies ist das durch Zwischenspeicherung (caching) beschleunigte Dateisystem.

Consistency Der Begriff Konsistenz bezieht sich in diesem Zusammenhang auf die Synchronisation zw. dem Back- und dem Front-Filesystem.

Abbildung 14.1: Struktur des Cache-Filesystems



14.5.1 Einrichtung eines Cache-Filesystems

1. Erzeugen eines Cache-Bereiches
   # **cfsadmin -c /export/home_cache**
2. Erzeugen eines Verzeichnisses, worin das Cache-Dateisystem eingehängt wird
   # **mkdir /export/home**
3. Mounten des langsamen Dateisystems (hier NFS)[1]
   # **mount -F chachefs -o backfstype=nfs,**
   **cachedir=/export/home_cache,cacheid=export_home_0823 **
   serverhost:/export/home /export/home
4. Verwenden des eingehängten Verzeichnisses

Ein permanenter Eintrag in der Datei */etc/vfstab* hätte folgendes Aussehen:

```
#device         device      mount    FS    fsck  mount   mount
#to mount       to fsck     point    type  pass  at boot options
#
#serverhost:/export/home -  /export/home cachefs - yes backfstype=nfs,cachedir=/export/home_cache,cacheid=export_
```

14.5.2 Kommandos

14.5.2.1 cachefsstat

chachefsstat zeigt Statistikdaten zu einem Cache-Filesystem.

[1] Das *mount*-Kommando für *cachefs* unterstützt die Option *demandconst*. Damit kann die automatische Konsistenzüberprüfung deaktiviert werden. Das ist nur für CD-ROM's oder sonstige Dateisysteme auf welche nur lesend zugegriffen wird zu empfehlen.

cachefsstat [-z] [mount-path]

-z setzt Statistikdaten zurück

mount-path Pfad des Mountverzeichnisses

14.5.2.2 cfsadmin

Administrative Tätigkeiten werden mit dem Kommando *cfsadmin* durchgeführt.

cfsadmin -c [-o cacheFS-parameters] cache_directory

cfsadmin -d [cache_ID | all] cache_directory

cfsadmin -l cache_directory

cfsadmin -s [mntpt1...| all]

cfsadmin -u [-o cacheFS-parameters] cache_directory

-c Erzeugen eines Cache-Verzeichnisses

-d Löschen eines Cache-Verzeichnisses

-l Ausgabe der 'cached filesystems' sowie Statistikdaten

-s Durchführung einer Konsistenzprüfung

-u Cache-Parameter ändern (*update*). Parameter können nur vergrößert werden. Soll ein Parameter verkleinert werden, so muss das Cache-Filesystem neu eingerichtet werden.

14.5.2.3 cachefslog

Mit *cachefslog* kann die Ausgabe von Logging-Daten gesteuert werden.

cachefslog [-f *logfile* | -h] *cachefs_mount_point*

Ohne Option wird angezeigt, ob Logging aktiv ist oder nicht

-f *logfile* Logginginformation wird in die Datei *logfile* geschrieben. Die Datei darf noch nicht existieren, sie wird automatisch erzeugt.

-h *halt* – Stoppt logging

cachefs_mount_point Mountverzeichnis

Wenn Logging aktiviert ist, kann mit dem Kommando *cachfswssize* die Größe des Cache angezeigt werden.

Syntax:

cachefswssize *logfile*

Ein Cache-Filesystem kann, wie Plattendateisysteme auch, mit dem Kommando *fsck* auf Konsistenz überprüft werden. Als Filesystemtyp ist *cachefs* anzugeben.

Beispiel:

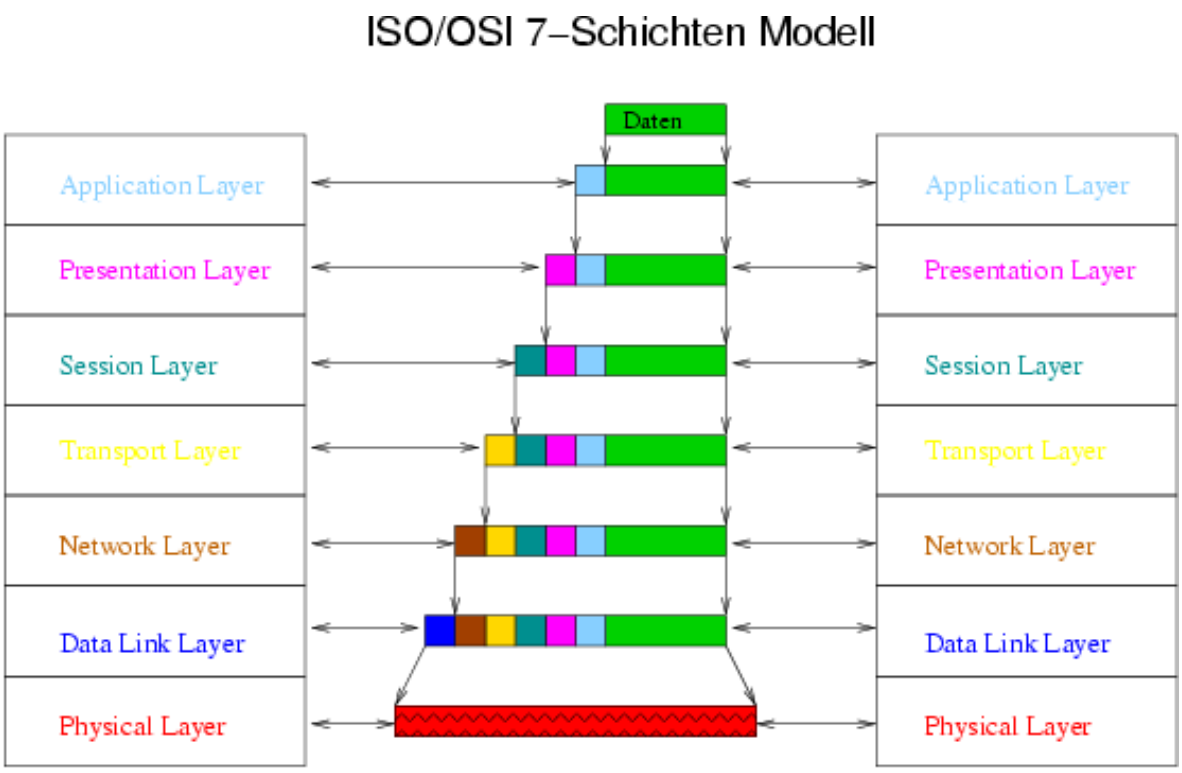
umount /export/home
fsck -F cachefs /export/home_cache

15 Netzwerk Grundlagen

15.1 Netzwerk Modelle

15.1.1 ISO / OSI 7 Schichten Modell

Zu Beginn der 80-er Jahre entwickelte die *International Organization for Standardization (ISO)* das *Open Systems Interconnection (OSI)* Referenzmodell. Es beschreibt die Struktur und Funktionsweise von Datenkommunikation. Dazu wurden - beginnend bei der Applikation bis zur Hardware - 7 sogenannte *Layer* (Schichten) definiert. Jede Schicht stellt der übergeordneten Schicht über eine klar definierte Schnittstelle Dienste zur Verfügung und nimmt ihrerseits Dienste der untergeordneten Schicht in Anspruch.



Die Aufgaben der einzelnen Schichten werden im Folgenden beschrieben:[1]

[1] Die Beschreibung der Schichten sind der Dokumentation des Programmes 'knetdump' (Linux) entnommen.

15.1.1.1 1.Schicht = Bitübertragungsschicht (Physical Layer)

Die unterste Schicht hat die Aufgabe, physikalische Signale in den verschiedenen Kommunikationskanälen zu koordinieren. Die elektrischen, funktionalen und prozeduralen Parameter zur Steuerung des physikalischen Übertragungsmediums innerhalb des Kommunikationssystems werden hier festgelegt. Daten werden ohne Rücksicht darauf, ob sie ankommen oder nicht, gesendet. Die Protokolle V.24, FDDI-PHY und Busprotokolle wie z.B. IEC-Bus werden dieser Schicht zugeordnet.

15.1.1.2 2.Schicht = Sicherungsschicht (Data Link Layer)

Diese Schicht ermöglicht einen sicheren Transport von Daten zwischen zwei benachbarten Stationen. Sie unterstützt die Erkennung und Behebung von Übertragungsfehlern sowie die Flußkontrolle zwischen Sender und Empfänger. Das Protokoll HDLC kann hier als Beispiel der Sicherungsschicht genannt werden.

15.1.1.3 3.Schicht = Vermittlungsschicht (Network Layer)

Die dritte Schicht stellt eine Art Leitzentrale für den Datenverkehr dar. Sie ist dafür zuständig, die geographische Entfernung zwischen den Endsystemen mit Hilfe von Vermittlungseinrichtungen zu überwinden. Gleichzeitig wird den Endsystemen die Möglichkeit eröffnet, mit verschiedenen anderen Endsystemen zeitlich bzw. logisch getrennt zu kommunizieren. Protokolle dieser Schicht sind CLNS / CONS.

15.1.1.4 4.Schicht = Transportschicht (Transport Layer)

Innerhalb der vierten Schicht werden die Verbindungen zwischen Sender und Empfänger aufgebaut. Es findet eine fehlerfreie Punkt-zu-Punkt-Verbindung statt. Hierbei spielt es keine Rolle, wieviele physikalische Vermittlungsknoten zwischen den Parteien sind und in welcher Weise sie zusammenarbeiten. Die Schicht 4 ist die erste Schicht, die unabhängig von der Hardware arbeitet. Zwischen Schicht 3 und 4 befindet sich eine Abstraktionsgrenze. Während Schicht 3 noch lokale Gegebenheiten beachten muß, wird in Schicht 4 allein das abstrakte Datenmodell benutzt. Als Beispiel dieser Schicht sei das Transmission Control und das User Datagram Protocol zu erwähnen. Protokolle der Transportschicht sind TP-0 bis TP-4.

15.1.1.5 5.Schicht = Kommunikationssteuerungsschicht (Session Layer)

Aufgabe dieser Schicht ist es, den abstrakten Datentransport in Dialogen, die vom Anwender initiiert werden können, zu behandeln. Die Art der Kommunikation (wechselseitig oder gleichzeitig in zwei Richtungen), die Ausfallsicherheit durch Checkpoints im Datenstrom und die Dialogkontrolle werden hier festgelegt. Die Protokolle HTTP (Hypertext Transfer Protocol), FTP (File Transfer Protocol) und Telnet können dieser Schicht zugeordnet werden.

15.1.1.6 6.Schicht = Darstellungsschicht (Presentation Layer)

Diese Schicht stellt Mittel zum darstellungsunabhängigen Austausch von Daten sowie zur Datenkompression und Verschlüsselung zur Verfügung, wie z.B. MIME (Multipurpose Internet Mail Extensions) oder HTML (Hypertext Markup Language). In dieser Schicht wird auch die Reihenfolge der Bytes (big endian bzw. little endian) bzw. des verwendeten Zeichencodes (EBCDIC, ASCII) berücksichtigt.

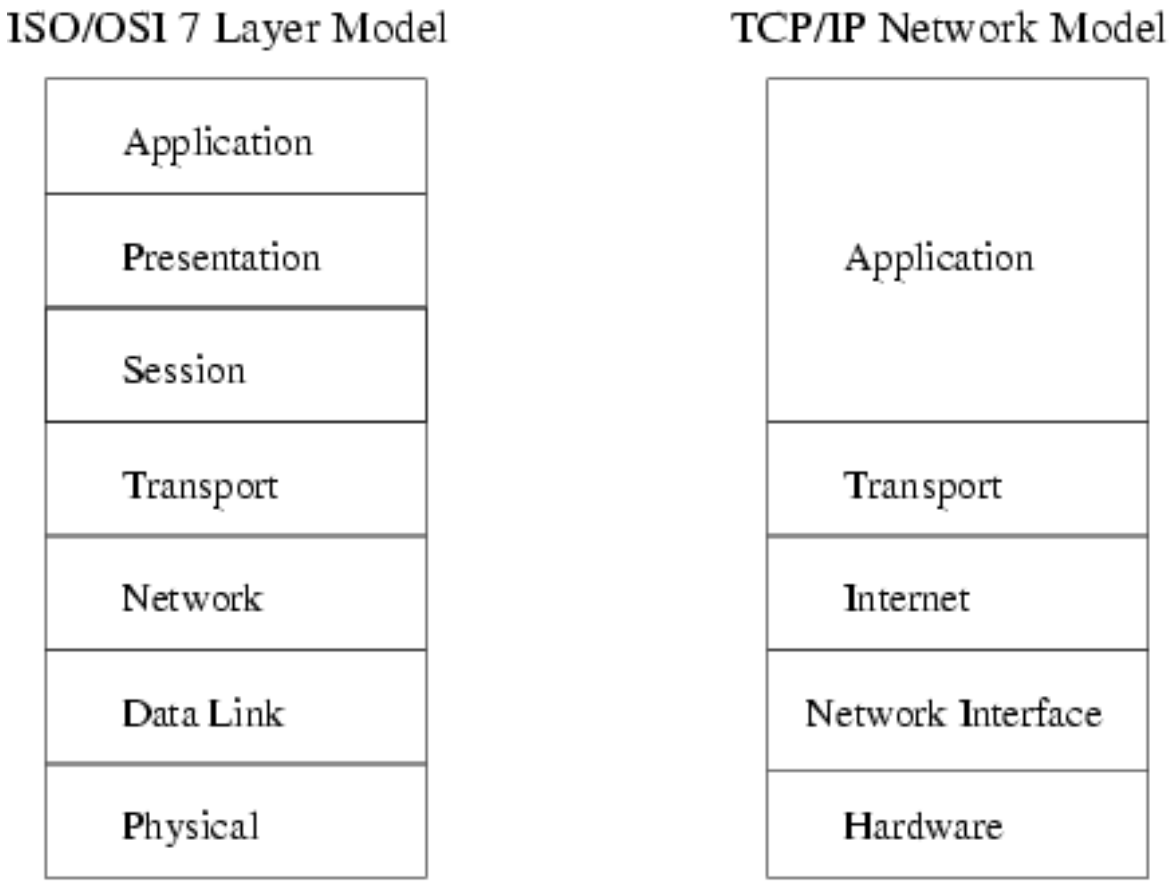
15.1.1.7 7.Schicht = Anwendungsschicht (Application Layer)

Die oberste Schicht ist für die Steuerung der untergeordneten Schichten und die Anpassung an die Anwendung verantwortlich. Die Anwendungsschicht bildet für die eigentliche Anwendung ein Fenster zum korrespondierenden System und dient dem Anwendungsprogramm als Verbindung zur Außenwelt. Bekannte Anwendungen dieser Schicht sind der Browser und FTP-Client.

In der Realität findet man zwar - zumindest bei den stark verbreiteten Implementationen - keine völlige Übereinstimmung mit diesem 7 Schichten Modell, trotzdem leistet es gute Dienste für das Verständnis der Datenübertragung.

15.1.2 TCP/IP Netzwerk Modell

Die ersten Netzwerke basierend auf TCP/IP wurden schon lange vor der Definition des OSI-Modells betrieben und können deshalb auch nicht exakt darauf abgebildet werden. Ein sinnvoller Vergleich ist nur bei den untersten 4 Schichten möglich. Die ISO/OSI-Schichten 5 bis 7 werden im TCP/IP-Netz meist zu einer Applikationsschicht zusammengefasst.



15.1.2.1 Hardware Layer

Aufgaben: physikalischer Transport von Signalen über ein Medium

Prinzip: elektrotechnische Normierung von Pegeln, Pinbelegung, Kabelqualität, ...

Protokoll: Ethernet-Spezifikation, Token-Ring, FDDI, ...

Verbindungseinrichtungen: Repeater

wo implementiert: LAN-Karte, Dfü-Kontroller, Verkabelung, ...

15.1.2.2 Network Interface Layer

Aufgaben: Fasst Bits in Dateneineiheiten (Frames) zusammen und kontrolliert die Übertragung

Prinzip: eindeutige Adressierung, genormtes Zugriffsverfahren (CSMA/CD, TokenRing, ...)

Protokoll: Ethernet V2, IEEE 802.2, 802.3, 802.5, FDDI, ...

Verbindungseinrichtungen: Bridge, Hub, Switch, ...

wo implementiert: LAN-Karte, evt. Transceiver, Kernel: Treibermodul

wichtige Kommandos: ok: banner
ok: .enet_addr
ifconfig -a
snoop
netstat -i
ping <*host*>

wichtige Dateien: /etc/ethers

15.1.2.3 Internet Layer

Aufgaben: Aufbau eines logischen Netzes, Verbindung von physikalischen Netzen, Routing, Fragmentieren von Daten, Fehler-Erkennung und -Behandlung, ...

Prinzip: global eindeutige Adressierung, Next-Hop Routing (dynamisch oder statisch)

Protokoll: IP, ICMP, ARP, RARP, RIP

Verbindungseinrichtungen: Router, Hosts

wo implementiert: Kernel-Modul: /dev/ip, /dev/icmp, /dev/arp
Dämon-Prozesse: routed, gated

wichtige Kommandos: # ifconfig <*interface*> inet <*addr*> netmask <*mask*> [*broadcast*] up|down
snoop
netstat -r[n]
ping <*host*>
traceroute <*host*>
ndd -set /dev/ip ip_forwarding 1

wichtige Dateien: /etc/protocols (Protokollnummer für Weitergabe an Schicht 4)
/etc/hostname.<*interface*>
/etc/inet/hosts
/etc/netmask
/etc/networks
/etc/netconfig
/etc/defaultrouter
/etc/notrouter
/etc/gateways (für RIP)

15.1.2.4 Transport Layer

Aufgaben: Herstellung einer expliziten Verbindung (TCP); bei verbindungslosen Diensten (UDP) vor allem Routing auf der Zielmaschine zur richtigen Anwendung (anhand Portnummer);

Prinzip: TCP: Handshake Verbindungsaufbau, -abbau, Keep-alive Pakete, Flusskontrolle, Sequenznumerierung

Protokoll: TCP, UDP

wo implementiert: Kernel-Modul: /dev/tcp, /dev/udp

wichtige Kommandos: # ndd -set /dev/<*prot*> <*parameter*> <*value*>
snoop

wichtige Dateien: /etc/services (Portnummern zur Weitergabe an oberen Layer)

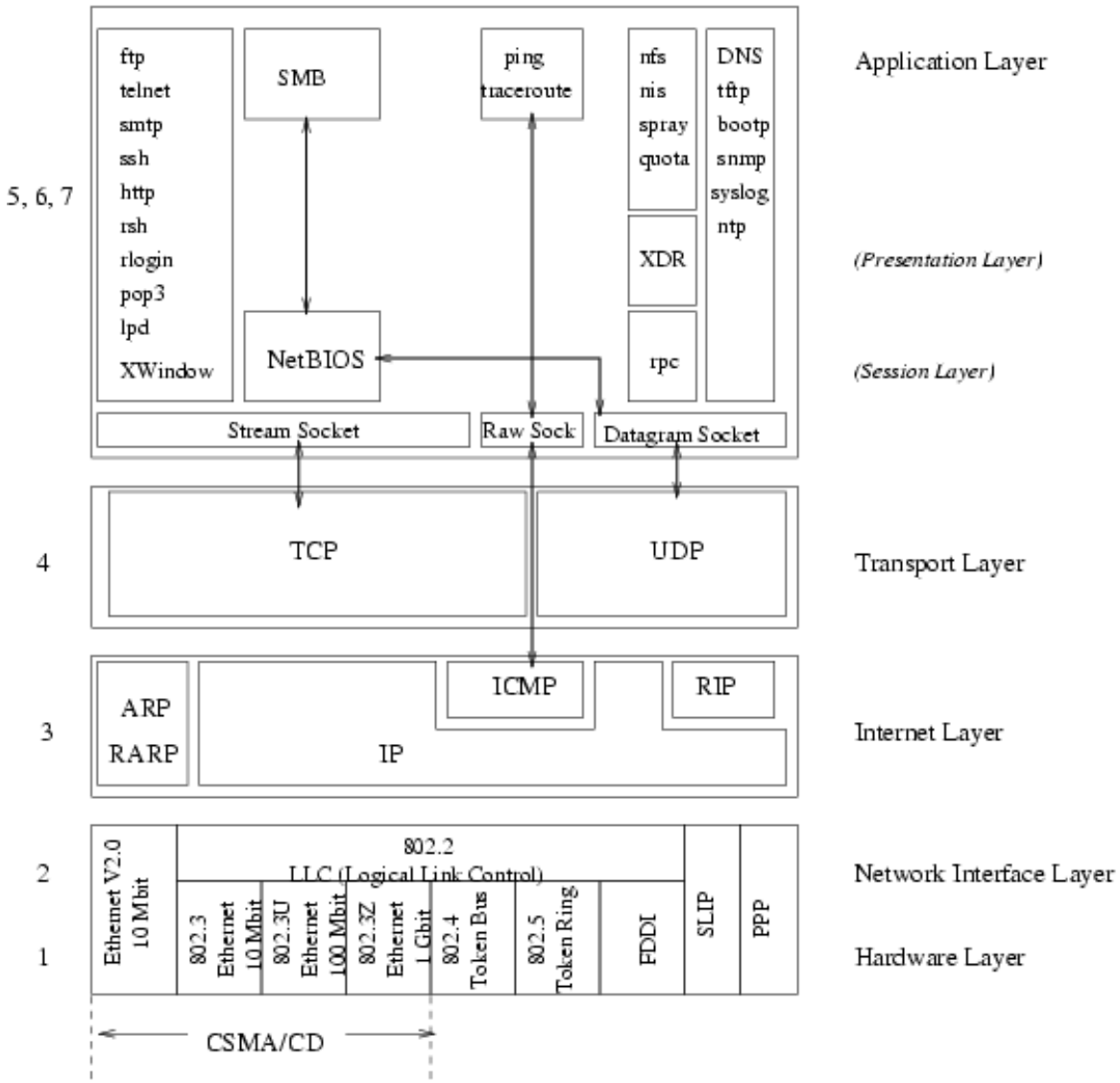
Nähere Informationen sind im Kapitel 16.5 auf Seite 207 zu finden.

15.1.2.5 Application Layer

Im TCP/IP Netzwerk Modell werden die obersten Schichten (5, 6 und 7) zusammengefasst. Nur teilweise lassen sich einzelne Funktionen einer ISO/OSI-Schicht zuordnen:

Zum Beispiel kann RPC (remote procedure call) etwa der Schicht 5 (Session Layer) zugeordnet werden.
XDR (external data representation) setzt auf RPC auf und entspricht der Schicht 6 (Presentation Layer) im ISO/OSI Model.

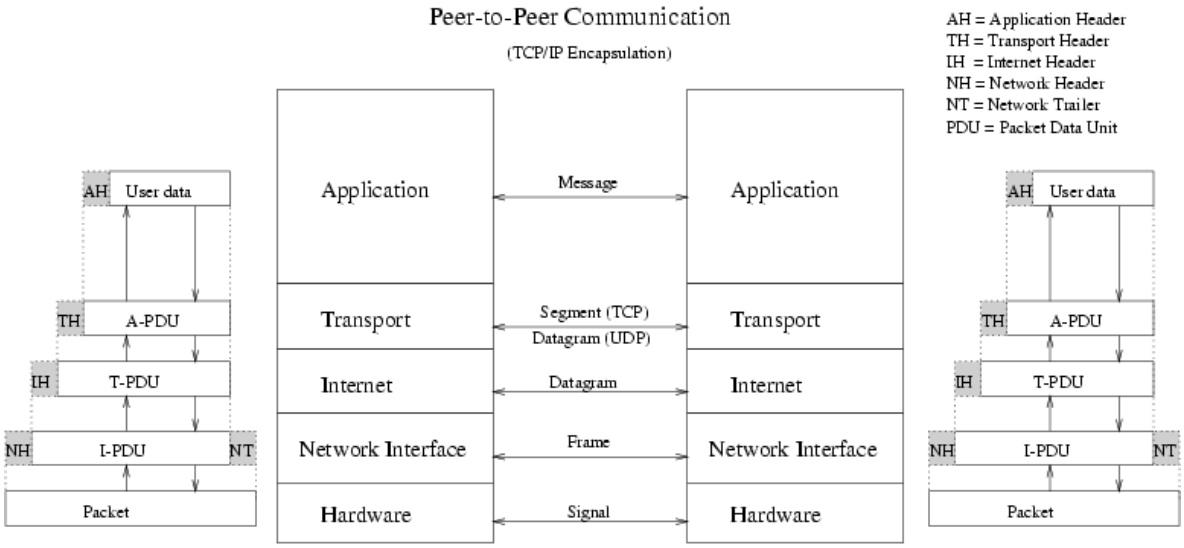
Die meisten Applikationen (telnet, ftp, tftp, rlogin, rsh, smtp, http, ...) der „TCP/IP-Welt“ setzen direkt auf der Socket-Schnittstelle auf, welche sich oberhalb der Schicht 4 befindet.


5, 6, 7
ftp
telnet
smtp
ssh
http
rsh
rlogin
pop3
lpd
XWindow
SMB
NetBIOS
ping
traceroute
nfs
nis
spray
quota
XDR
rpc
DNS
tftp
bootp
snmp
syslog
ntp
Application Layer
(Presentation Layer)
(Session Layer)
Stream Socket
Raw Sock
Datagram Socket
4
TCP
UDP
Transport Layer
3
ARP
RARP
IP
ICMP
RIP
Internet Layer
2
1
Ethernet V2.0 10 Mbit
802.2 LLC (Logical Link Control)
802.3 Ethernet 10 Mbit
802.3U Ethernet 100 Mbit
802.3Z Ethernet 1 Gbit
802.4 Token Bus
802.5 Token Ring
FDDI
SLIP
PPP
Network Interface Layer
Hardware Layer
CSMA/CD

15.1.3 Peer-to-Peer Kommunikation

Wenn Systeme den Datenaustausch mittels TCP/IP-Protokoll durchführen, so findet eine *Peer-to-Peer* Kommunikation statt. Das bedeutet, dass eine bestimmte Schicht mit der ihr korrespondierenden Schicht am anderen Ende des Übertragungskanals kommuniziert.

Jede Schicht fügt den Daten, welche sie von der übergeordneten Schicht erhält einen Header hinzu (*encapsulation)*. Die Information, welche in diesem Header hinzugefügt wird, wertet die korrespondierende Schicht am anderen Ende aus, um die darin enthaltenen Daten wieder korrekt an die gewünschte obere Schicht weiterzuleiten (*de-encapsulation*). In der Netzwerkschicht wird bei Verwendung von Ethernet (CSMA/CD) nicht nur ein Header, sondern auch ein Trailer (CRC) hinzugefügt.



15.1.4 Netzwerk Protokolle

Ein Protokoll ist eine Zusammenstellung von Regeln, wie der Datenaustausch zwischen Endpunkten zu erfolgen hat. Diese Regeln beinhalten:

Syntax Datenformat und Kodierung

Semantik Kontroll-Information und Fehlerbehandlung

Timing Übertragungsgeschwindigkeit

Wer sich intensiver mit Protokollen auseinandersetzen will oder muss, der findet alle Details im Internet: http://RFC.net/

15.1.4.1 TCP/IP Netzwerk Interface Layer Protokolle

SLIP *Serial Line IP* ermöglicht das Übertragen von IP-Paketen über serielle Schnittstelle (veraltet, wird durch PPP ersetzt)

PPP *Point-to-Point Protocol*; Übertragung von IP-Paketen über serielle Schnittstelle

15.1.4.2 TCP/IP Internet Layer Protokolle

ARP *Address Resolution Protocol* dient dazu, 32-Bit IP-Adressen in 48-Bit Ethernet-Adressen zu übersetzen

RARP *Reverse Address Resolution Protocol* ist das Gegenstück zu ARP; damit werden Ethernet-Adressen in IP-Adressen übersetzt

IP *Internet Protocol* dient dazu den Weg zum Ziel-Rechner mit der IP-Adresse zu finden

ICMP *Internet Control Message Protocol* liefert Fehlermeldungen und wird für Kontrollfunktionen bei IP-Paketen verwendet

15.1.4.3 TCP/IP Transport Layer Protokolle

TCP *Transmission Control Protocol* ist ein verbindungsorientiertes Protokoll, welches Datenübertragung voll duplex ermöglicht. Viele Applikationen setzen auf dieses Protokoll auf.

UDP *User Datagram Protocol* ist ein verbindungsloses Protokoll um 'Datagramme' zu versenden.

15.1.4.4 TCP/IP Application Layer Protokolle

DNS *Domain Name System* - Dabei handelt es sich um eine hierarchisch strukturierte, verteilte Datenbasis, welche Domänennamen und dazugehörige Mail-Server, sowie Rechnernamen und IP-Adressen verwaltet.

FTP *File Transfer Protocol* wird verwendet, um Dateien von einem Rechner zu einem anderen zu kopieren.

telnet Ermöglicht es, von einem Terminal aus eine Verbindung zu einem fernen Rechner herzustellen und wie an einem lokal angeschlossenen Terminal zu arbeiten.

rlogin Dies dient demselben Zweck wie telnet. Unterschiede bestehen vor allem in der Authentifizierung.

DHCP *Dynamic Host Configuration Protocol* automatisiert die Zuweisung von IP-Adressen und sonstigen Konfigurationsdaten in einem Netzwerk.

SMTP *Simple Mail Transfer Protocol* überträgt e-mails von einem Rechner zu einem Mail-Server.

SNMP *Simple Network Management Protocol* ermöglicht die Überwachung und Kontrolle von Geräten in einem Netzwerk.

POP-3 *Post Office Protocol, Version 3*- ermöglicht es e-mails von einem Mail-Server abzuholen.

HTTP *Hypertext Transfer Protocol* wird im WWW (world wide web) verwendet, um Texte, Bilder und sonstige Informationen mit einem sogenannten Web Browser darzustellen.

15.1.5 LAN Topologie

Ein *LAN* (*Local Area Network*) ist ein Kommunikationssystem, welches Geräte der elektronischen Datenverarbeitung miteinander verbindet. LAN's verbinden PC's, Workstations, Server, Drucker u.ä. Geräte, um deren Dienste gegenseitig nützen zu können. Die Geräte können sich in unterschiedlichen Räumen oder sogar Gebäuden befinden. Erfolgt die Verbindung über 'fremde' Leitungen und großen Distanzen, spricht man von einem WAN (Wide Area Network).

Ein LAN kann unterschiedliche Struktur aufweisen:

15.1.5.1 Bus Konfiguration

Der Bus war die ursprüngliche LAN Topologie für das Ethernet. Dabei handelte es sich um ein gelbes Koaxialkabel („Yellow Cable“ oder „RG8“), welches eine maximale Länge von 500 m hatte. Geräte können mittels eines sogenannten Transceivers mit dem Kabel verbunden werden. Der Anschluss dieses Transceivers erfolgt durch eine Bohrung in das 'Yellow Cable'. In das so entstandene Loch wird eine dünne Nadel zum Leiter in der Kabelmitte geführt. Die Bohrung muss an den markierten Stellen des Kabel erfolgen.

Später wurde das teure 'Yellow Cable' immer häufiger durch ein billigeres Koaxialkabel („Cheapernet“ oder RG58) ersetzt. Der Anschluss erfolgt nicht mehr durch eine Bohrung in das Kabel, sondern mittels eines T-Anschlussstückes direkt auf die LAN-Karte. Der Transceiver befindet sich bei dieser Anschlussart auf der LAN-Karte selbst.

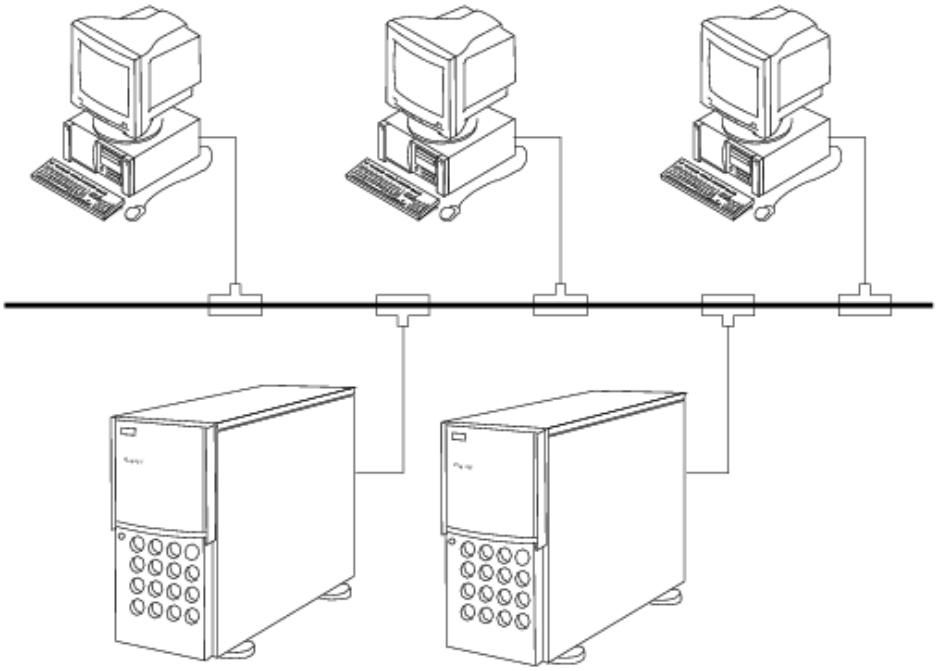
15.1.5.2 Stern Konfiguration

Bei dieser Art der Verkabelung befindet sich an zentraler Stelle ein sogenannter „HUB“, von dem aus die Anschlussleitungen sternförmig zu den angeschlossenen Geräten führen. Vorteil dieser Verkabelung ist die wesentlich höhere Zuverlässigkeit. Im Gegensatz zur Bus-Konfiguration kann ein einzelnes defektes Gerät oder Kabel nicht gleich das gesamte Netz lahmlegen. Ein weiterer Vorteil ist, dass bei dieser Topologie auch höhere Übertragungsgeschwindigkeiten unterstützt werden. Die Stern-Konfiguration setzt sich in den letzten Jahren immer mehr gegenüber der Bus-Konfiguration durch.

15.1.5.3 Ring Konfiguration

In einer Ring Konfiguration wird jeweils der Ausgang eines Gerätes mit dem Eingang des nächsten Gerätes verbunden. Das letzte Geräte ist wiederum mit dem ersten Gerät der Kette verbunden. Die

Abbildung 15.1: Bus Konfiguration



LAN Bus Topologie

Abbildung 15.2: Stern Konfiguration

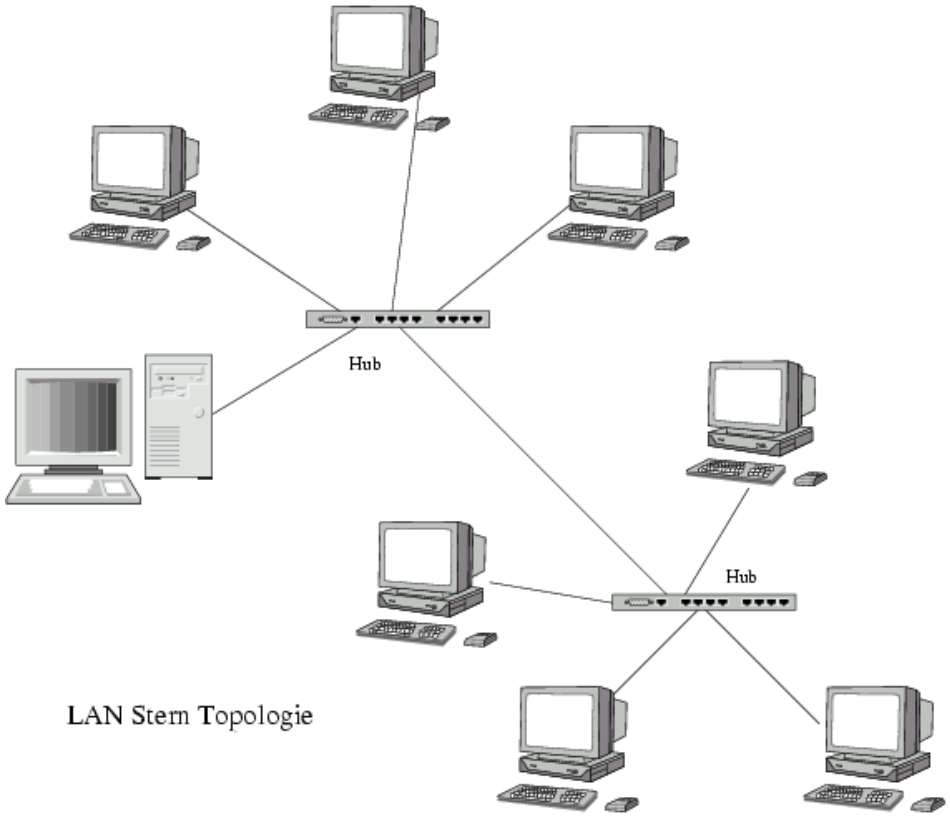


Abbildung 15.3: Ring Konfiguration

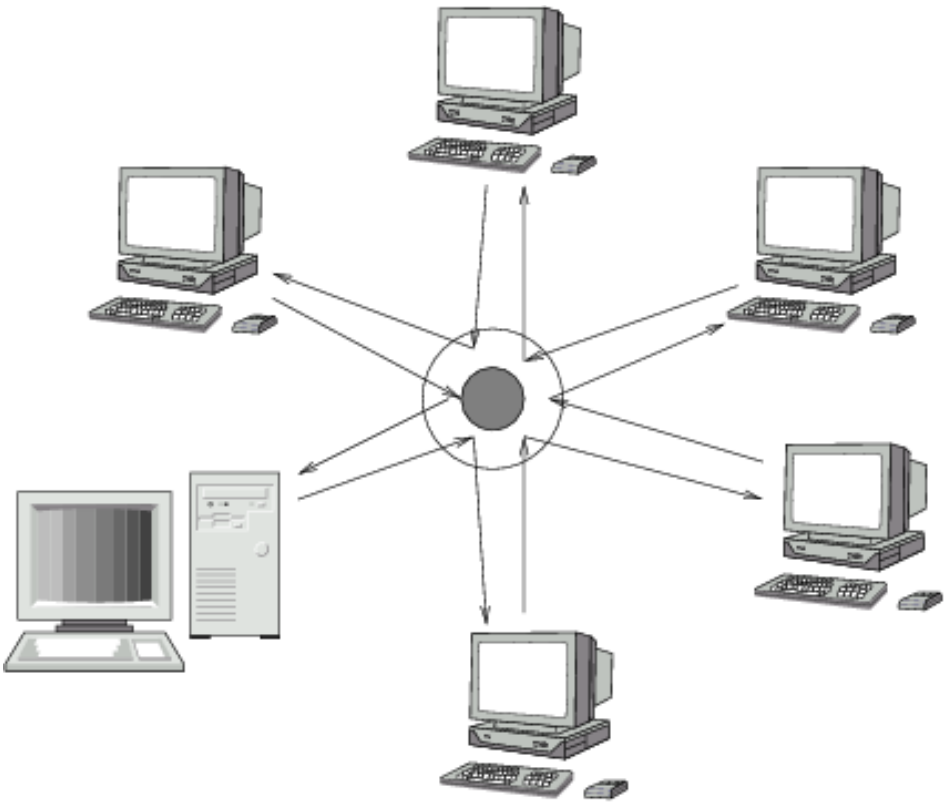
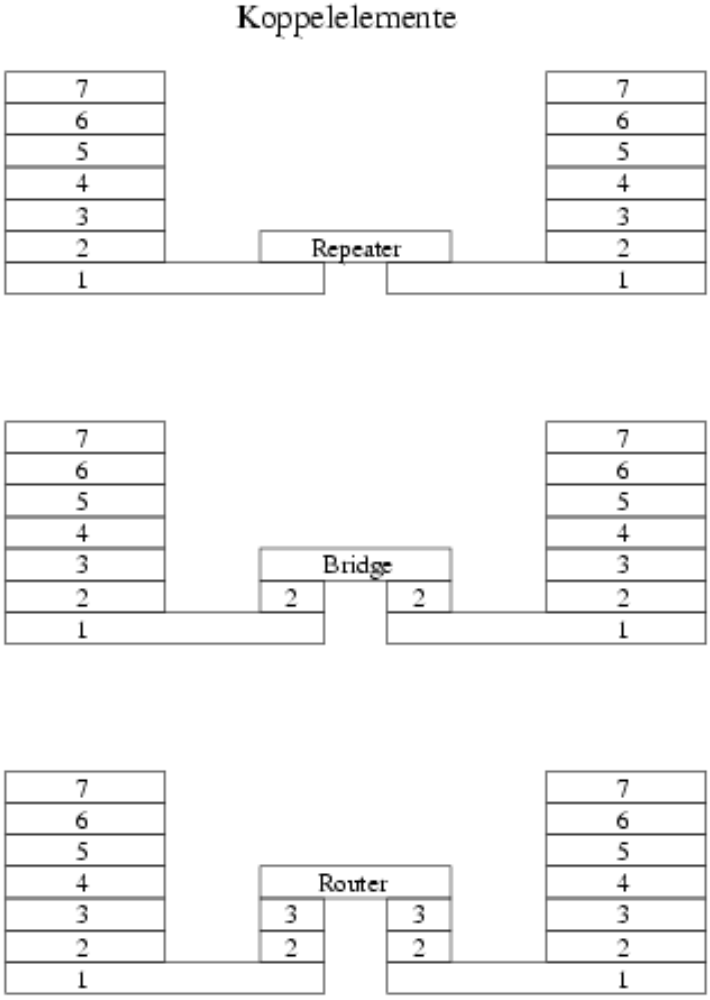


Abbildung 15.4: Koppelelemente



Daten werden also durch alle Teilnehmer des Ringes gereicht. Fällt ein Gerät in der Kette aus, steht die Übertragung.

In der Praxis sieht es jedoch meist so aus, dass sich zentral ein intelligenter Hub befindet, von dem aus jeweils eine Sende- und eine Empfangsleitung zum Gerät führt. Dies sieht dann wie eine Sternverkabelung aus, die Daten werden jedoch in einem Ring übertragen. Mit Hilfe der Intelligenz des Hubs wird auch das Problem entschärft, dass bei Ausfall eines Teilnehmers das gesamte Netz steht. Der Hub ist in der Lage ein defektes Gerät zu erkennen und dieses aus dem Ring zu schalten.

15.1.5.4 LAN Komponenten

Backbone Hauptverbindungs-Einrichtung in einem Netzwerk

Segment Eine zusammenhängende Verbindungsleitung woran im Allgemeinen Netzwerkkomponenten in einen Punkt zu Punkt-Verbindung angeschlossen sind.

Repeater Ein Gerät, das Datensignale bitweise verstärkt und regeneriert um die Strecke einer Datenübertragung zu vergrößern. Ein Repeater arbeitet am Hardware-Layer und interpretiert die übertragenen Daten nicht.

Hub Das zentrale Gerät, woran alle Rechner über „Twisted Pair Verkabelung“ angeschlossen sind.

Bridge Ein Gerät, das 2 oder mehrere Netzwerk-Segmente miteinander verbindet. Die Bridge arbeitet am Link Layer und interpretiert Ethernetadressen (MAC-Adresse), um die Pakete dann entsprechend zu filtern oder weiterzuleiten.

Switch Ein Gerät mit mehreren Anschlüssen, das die Verbindung zwischen Segmenten automatisch herstellt und trennt. Der Switch verbindet nur jene Segmente, welche für den Datenaustausch benötigt werden. Dadurch werden Kollisionen vermieden und der Durchsatz verbessert.[2]

Router Der Router hat 2 oder mehr Netzwerkschnittstellen. Das Gerät arbeitet am Network Layer und wertet IP-Adressen aus und kümmert sich um die Weiterleitung der Daten über die richtige Schnittstelle.

Gateway Ein Verbindungsgerät zwischen 2 oder mehr Netzwerken mit unterschiedlichen Protokollen. Der Gateway führt die Protokoll-Umsetzung durch.

Konzentrator Ein Gerät worüber unterschiedlichste Datenpakettypen fließen können. Ein Konzentrator ist meist ein Gerät, welches mehrere Geräte wie Repeater, Hub, Switch, Router und Gateway in sich vereint.[3]

15.1.5.5 Ethernet Komponenten

Ethernet Kontroller Die Hardware-Komponente, die Ethernet-Frames erzeugt oder liest. (die LAN-Karte)

Transceiver Eine aktive Komponente, welche die Daten vom Ethernet-Kabel zum Kontroller und umgekehrt transportiert.

Transceiver Kabel Verbindungskabel zwischen Transceiver und Computer (AUI-Schnittstelle). Falls sich der Transceiver am Ethernet Kontroller befindet (wie z.Bsp. bei Cheapernet), entfällt dieses Kabel.

Thick Ethernet RG8 Koaxialkabel, 50 Ω „Yellow Cable“; Ein dickes, hochwertiges Koaxial-Kabel für Ethernet. Am Kabel befinden sich Markierungen für mögliche Transceiver-Verbindungen alle 2,5 Meter. Die maximale Länge eines Segmentes beträgt 500 Meter. Durch Einsatz von Repeatern (max. 4) kann die maximale Länge des Ethernet-Netzwerkes auf 2500 Meter verlängert werden.

Thin Ethernet RG58 Koaxialkabel, 50 Ω Ein dünnes, im Vergleich zu RG8 niederwertiges Koaxial-Kabel für Ethernet. Es ist wesentlich billiger („Cheapernet“) und einfacher zu verlegen als das Thick Ethernet Kabel. Die maximale Länge beträgt 180 Meter, mit Repeatern kann die Länge auf 540 Meter ausgeweitet werden.

[2] Auf welchem Layer ein Switch arbeitet, ist unterschiedlich je nach Gerät (Layer 2 oder 3).

[3] Mir scheint, dass diese Bezeichnung nur für die Solaris - Zertifizierungsprüfung von Bedeutung ist. In der Praxis nennt man die Komponenten eher beim richtigen Namen - also Hub oder Router etc.

Abschluss Widerstände, 50 Ω Widerstände, die an den Enden der Ethernet-Kabeln angebracht werden, um die Reflexion von Signalen zu verhindern.

Twisted Pair Kabel Das Kabel enthält 4 Leiter (je 2 verdrillte Paare). Die Verdrillung der Leiterpaare bewirkt eine Reduzierung von Störeinflüssen. Diese Kabel werden für die Stern Konfiguration mit Hub's verwendet. Die größtmögliche Entfernung vom Hub beträgt 100 Meter. Es gibt unterschiedliche Qualitäten: Kabel der Kategorie 3 werden verwendet für 10BaseT- und 100BaseT4-Netzwerke.[4] Die Litzen sind 2 bis 3 mal je Fuß verdrillt. Kabel der Kategorie 5 sind 2 bis 3 mal je Zoll verdrillt und werden für 10BaseT- und 100BaseTx-Netzwerke verwendet.

15.1.5.6 SUN Communications Controller

Zugriffsmethode	Kontroller-Name	Device-Name
ATM	SunATM-155 SBus fiber controller	ba0
	SunATM-155 SBus UTP5 controller	ba0
	SunATM-622 SBus fiber controller	ba0
Ethernet	Lance Ethernet SBus controller	le0
	Quad Lance Ethernet SBus controller	qe0
Fast Ethernet	Sun Fast Ethernet 1.0 SBus controller	be0
	Sun Quad Fast Ethernet 1.0 SBus controller	qfe0
	Sun Quad Fast Ethernet 2.0 SBus controller	hme0
	Sun Fast Ethernet 2.0 SBus/PCI controller	hme0
FDDI	Sun FDDI/S SAS Fiber SBus controller - single	nf0
	Sun FDDI/S DAS Fiber SBus controller - double	nf0
Token Ring	SunTri/S 4/16 Mbps SBus controller	tr0
Gigabit Ethernet	Vector Gigabit Ethernet V1.1	vge0
	GEM Gigabit Ethernet V2.0	ge0

15.1.6 Übertragungsmethoden

15.1.6.1 Ethernet – IEEE 802.3

Diese Zugriffsart ist weiter unten genauer beschrieben und wird hier deshalb nur als Schlagwort erwähnt.

15.1.6.2 ATM – Asynchronous Transfer Mode

ATM wurde entwickelt mit der Idee unterschiedlichste Daten (Sprache, Bild, ...) über einen Übertragungsweg über weite Strecken zu transportieren. Da je nach Art der Daten die Übertragung zeitkritisch ist (z. Bsp. Ton), können diese unterschiedlich priorisiert werden. ATM wird vor allem von größeren Firmen eingesetzt, um Filialen miteinander über einen schnellen Datenweg miteinander zu verbinden.

ATM wurde definiert für die Übertragungsgeschwindigkeiten von 45, 100, 155 und 622 MBit/Sekunde.

[4] 100BaseT4 benötigt 8 Adern. Für 100 MBit-Netzwerke werden deshalb üblicherweise Kabel der Kategorie 5 verwendet, welche mit nur 4 Adern auslangen finden.

15.1.6.3 Token Ring – IEEE 802.5

Der Token Ring war ursprünglich eine Entwicklung von IBM und ist immer noch sehr verbreitet. Die Geräte sind in einem logischen Ring[5] miteinander verbunden.

Im Token Ring läuft ständig ein sogenanntes Token von einem Gerät zum Nächsten die Runde. Das Token ist ein leerer kleiner Datenframe. Ist ein Gerät im Besitz eines leeren Tokens, kann es von ihm in ein Datenframe umgewandelt, mit Daten gefüllt und weitergesendet werden.

Der große Vorteil gegenüber Ethernet ist, dass Daten nicht an alle Netzteilnehmer gesandt wird. Der Datenframe läuft nur so lange am Ring, bis der Empfänger erreicht ist - nur im schlimmsten Fall ist dies die Station genau 'hinter' der Sendestation. Aufgrund der kollissionsfreien Übertragungmethode ist auch eine bessere Auslastung als am Ethernet zu erreichen. Während auf einem Ethernet schon bei 40% Last Probleme auftreten, kann der TokenRing weit höher ausgelasten werden.

Nachteilig ist, dass diese Übertragungsart weit aufwändiger ist als Ethernet. Die Netzwerkkarten sind im Allgemeinen teurer. Der 'Overhead' durch die Token ist größer, was dazu führt, dass Ethernet bei schwächerer Last performanter ist. Zudem kommt, dass es inzwischen Fast-Ethernet mit 100MBit oder gar Gigabit Ethernet gibt, hingegen TokenRing mit 4 bzw. 16 MBit Übertragungsrate arbeitet[6].

15.1.6.4 FDDI – Fiber Distributed Data Interface

FDDI ist ein Doppelring auf Basis des Token Ringes. Ursprünglich nur für Glasfaser vorgesehen, gibt es inzwischen auch Kupferkabel. Diese sind viel billiger, sind aber nur auf viel kürzere Distanzen einsetzbar. Man verzichtet dabei auch auf den Vorteil, dass Glasfaser unempfindlich ist auf elektromagnetische Einstrahlung und nicht[7] abhörbar ist.

Durch die doppelte Ausführung des Ringes ist ein Ausfallschutz gegeben – fällt ein Ring aus, können die Daten über den 2. Ring transportiert werden. Es ist auch eine Lastteilung möglich. Z.Bsp. kann ein Gerät Daten auf beiden Ringen in entgegengesetzte Richtungen gesendet werden. Jene Daten, die rascher beim Empfänger ankommen, werden verwendet.

FDDI unterstützt synchrone und asynchrone Datenübertragung. Die synchrone Übertragung beansprucht eine vorgegebene Bandbreite des FDDI-Netzwerkes, die asynchrone Übertragung verwendet den verbleibenden Rest. Synchrone Übertragung verwenden Stationen, welche einen kontinuierlichen Datenstrom (z.Bsp. für die Sprachdatenübertragung) benötigen.

[5] Um den Ausfall des gesamten Ringes durch einen defekten Teilnehmer zu verhindern, werden die Kabeln sternförmig von einer Verteilerleiste zu den Geräten geführt. In dieser Leiste kann ein Geräteanschluss überbrückt werden, wenn dort Fehler auftreten. Nichts desto trotz laufen die Daten von einem Gerät zum anderen quasi im Kreis.

[6] Es mag inzwischen auch schnellere TokenRing Installationen geben, diese sind aber kaum verbreitet.

[7] .. oder nur mit sehr hohem technischen Aufwand, der eher bemerkt wird

15.1.6.5 Datenübertragungs-Medien

Kabeltyp	Datenrate	max. Segmentlänge	Medium	Anschluss	Topologie	Bemerkung
10Base2	10MBit/s	185m	Koaxkabel, RG58	BNC-Stecker, T-Stücke	Bus	max. 30 Stationen; min. 0,5m Abstand
10BaseT	10MBit/s	100m	UTP, 2 Paare, Kat.3,4,5	8-polig RJ45	Punkt-Punkt	
100BaseTX	100MBit/s	100m	UTP, 2 Paare, Kat.5	8-polig RJ45	Punkt-Punkt	
100BaseT4	100MBit/s	100m	UTP, 4 Paare, Kat.3,4,5	8-polig RJ45	Punkt-Punkt	
10BaseFL	10MBit/s	2000m	Multi 62,5μm, 850nm;oder Monomode	meist ST-Stecker	Punkt-Punkt	
100BaseFX	100MBit/s	412m (2km Vollduplex)	Multimode 62,5μm, 1350nm; oder Monomode	SC-Stecker	Punkt-Punkt	
1000BaseCX	1GBit/s	25m	Twinax Kupferkabel		Punkt-Punkt	
1000BaseLX	1 GBit/s	550m 550m 5 km	Multimode 62,5μm, 1270nm Multimode 50μm, 1270nm Monomode 10μm, 1270nm		Punkt-Punkt	
1000BaseSX	1GBit/s	275m 550m	Multimode 62,5μm, 830nm Multimode 50μm, 830nm		Punkt-Punkt	

15.2 LAN Planung

15.3 Ethernet

Ethernet ist die am weitesten verbreitete LAN-Technologie. Ethernet wurde ursprünglich von Xerox, DEC und Intel entwickelt und ist nun in der Norm IEEE 802.3[8] beschrieben. Alle Geräte hängen direkt am Netz und konkurrieren um den Netzzugriff mittels CSMA/CD Protokoll.

15.3.1 CSMA/CD

CSMA/CD steht für „Carrier Sense - Multiple Access - Collision Detect“ und beschreibt den Ablauf, wie Konflikte durch gleichzeitigen Sendeversuch gelöst werden. Je mehr Geräte am gleichen Netzwerk betrieben werden, desto wahrscheinlicher ist es, dass zwei oder mehrere Geräte zugleich senden wollen.

Soll ein Datenpaket gesendet werden, so wird zunächst geprüft, ob das Übertragungsmedium (Koaxialkabel, UTP-Kabel, Glasfaser, ...) frei ist, oder ob das Signal eines anderen Netzwerkteilnehmers anliegt. Ist das Medium belegt, wird bis Freiwerden gewartet. Ist die Leitung frei, werden zusätzliche 9,6 μs (das entspricht etwa 12 Bytes) gewartet, damit zwischen den Datenpaketen eine kleine Zeitlücke entsteht (Interframe Spacing, Controller Recovery Time). Dann wird der Frame gesendet und zugleich geprüft, ob es zu einer Kollision kommt. Kam es zu keiner Kollision, ist die Übertragung erledigt und die Leitung ist frei für die nächste Übertragung. Wird aber eine Kollision erkannt, so werden noch mindestens 32 bis maximal 48 Bit gesendet (JAM-Signal), damit auch der am weitesten entfernte Netzwerk-Teilnehmer die Kollision erkennen kann. Nun warten alle Teilnehmer, die gerade senden wollten eine gewisse Zufallszeit und starten einen erneuten Sendeversuch. Die Wartezeit wird bis zu 10. Wiederholung jeweils verdoppelt, bei weiteren Wiederholungen bleibt sie dann gleich. Nach 16 erfolglosen Sendeversuchen wird an die nächsthöhere Schicht ein „Excessive Collision Error“ gemeldet.

Da der Ablauf sehr simpel ist, funktioniert diese Methode sehr gut bei geringer Auslastung des Netzwerkes. Es entstehen minimale Sendeverzögerungen und damit ist ein guter Durchsatz gewährleistet. Wird das Netz aber stark belastet, so kommt es durch die ständigen Wiederholungen zu einer starken Performanceeinbuße. In diesem Falle ist ein Token-Verfahren wie z.Bsp. beim Token-Ring trotz komplizierterem Protokoll eindeutig überlegen.

Um dem oben beschriebenen Problem zu begegnen, werden heute in Netzwerken, in denen mit höherer Last zu rechnen ist, vorwiegend Switches eingesetzt. Damit hängt jedes Gerät in einem eigenen Segment, Kollisionen werden vermieden. Daten werden vom Switch nur dorthin geleitet, wo sie laut MAC-Adresse hingehören. Das erhöht nicht nur den Durchsatz, sondern bringt auch zusätzlichen Schutz vor ungewolltem Mitlesen von Daten am Netz.[9]

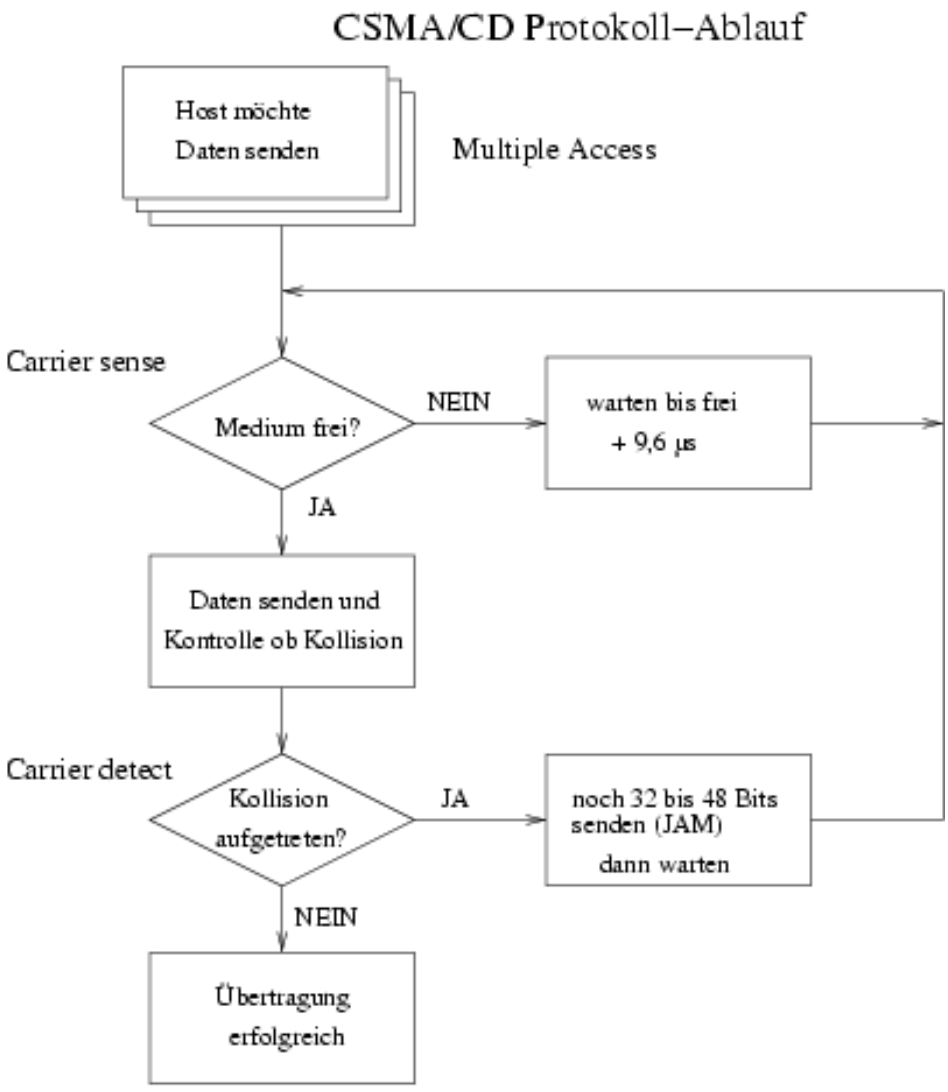
15.3.2 Ethernet Adresse / MAC Adresse

Die *Ethernet Adresse* ist eine 48-Bit Zahl. Sie wird in hexadezimaler Schreibweise angegeben, wobei die 6 Bytes jeweils durch einen Doppelpunkt voneinander getrennt werden (z.Bsp.: 00:10:A4:F4:DE:D7). Diese Adresse wird auch *hardware address* oder *media access control address* (*MAC Address*)

[8] Das ursprüngliche Ethernet V1.0 ist praktisch nicht mehr vorhanden. Ethernet V2.0 jedoch existiert parallel zu jenem Ethernet, welches in IEEE 802.3 beschrieben ist. Es unterscheidet sich leicht im Frameaufbau (siehe weiter unten).

[9] WARNUNG! Es garantiert aber keine Abhörsicherheit! Es ist nicht sehr schwierig Daten dennoch zu 'erschnüffeln' sodass sensible Daten unbedingt zusätzlich geschützt werden müssen!

Abbildung 15.5: CSMA/CD



genannt. Die Hersteller von Ethernetkarten garantieren, dass diese Addresse weltweit eindeutig ist. [10] Tatsächlich ist es aber nur innerhalb eines Netzes (dieselbe Netzadresse) nötig, dass die MAC-Adresse eindeutig ist. Sobald Daten zwischen verschiedenen Netzen ausgetauscht werden und 'geroutet' werden muss, hat die MAC-Adresse keine Bedeutung mehr, da diese außerhalb des eigenen Netzes nicht mehr mitgeschickt wird.

Alle SUN Workstations bzw. SUN Server haben diese Ethernet Adresse im NVRAM hinterlegt. Somit kann diese Adresse auch mit anderen Werten überschrieben werden.[11]

Es gibt 3 Typen von Ethernet Adressen:

Unicast Adresse Dies ist die soeben oben beschriebene eindeutige Adresse jeder Netzwerkkarte. Um ein Datenpaket gezielt an einen Host zu schicken wird dessen MAC-Adresse als Zieladresse ins Paket gepackt.

Broadcast Adresse Um eine Nachricht an alle im lokalen Netzwerk befindlichen Geräte zu senden, wird die Broadcast als Zieladresse eingefügt. Diese lautet FF:FF:FF:FF:FF:FF. Erhält ein Host ein Paket mit der Broadcastadresse, wird das Paket gleich durch den Network Interface Layer in den nächsthöheren Layer (Internet Layer) gereicht.

Multicast Adresse Man kann auch eine Gruppe von Rechnern ansprechen. Dies geschieht mit Hilfe einer Multicast Adresse. Die ersten 3 Bytes einer Multicast-Adresse lauten 01:00:5E. Mit den folgenden 3 Bytes kann man die entsprechende Gruppe identifizieren.

15.3.3 Ethernet Frame

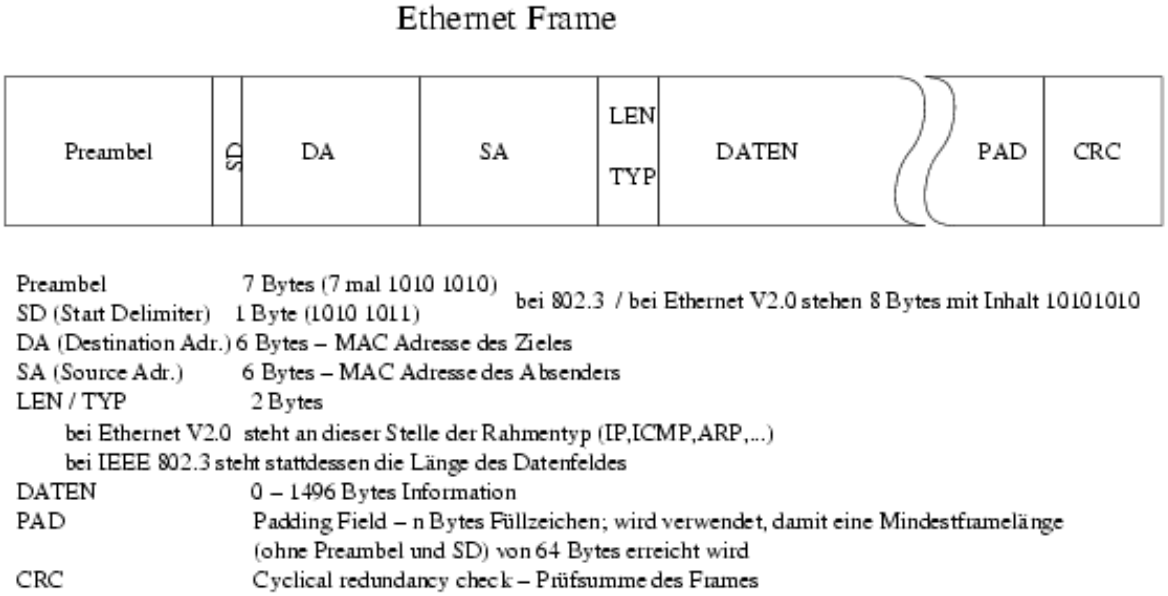
Als im Jahre 1985 die Normen IEEE 802.3 und 802.2 festgehalten wurden, war das Ethernet Protokoll V2.0 bereits sehr stark verbreitet und nicht mehr wegzudenken. Ethernet V2.0 deckt die beiden Schichten 1 und 2 (Physical Layer und Data Link Layer) ab und kann deshalb nicht einfach auf die Schichten der ISO-Norm abgebildet werden. In den Normen 802.3 und 802.2 wurden die Aufgaben der jeweiligen Schicht getrennt. Man nahm dabei Rücksicht darauf, dass die Koexistenz mit Ethernet V2.0 erhalten bleibt. Die Header von Ethernet V2.0 und 802.3 sind fast identisch. Bei 802.3[12] wird anstatt der Ethernet Typ-Kennung die Länge des Datenfeldes eingetragen. Da es bei Ethernet V2.0 keine Typ-Kennung gibt, deren Zahl kleiner ist als die maximale Framegröße (1500) gibt es keine Konflikte. Die 'eingepackten' Daten, welche im Datenfeld des Ethernetframes stecken unterscheiden sich aber grundlegend zwischen Ethernet V2.0 und 802.2. 802.2 ist das 'Logical Link Control Protocol' und wickelt Layer 2 ab.

[10] Die ersten 3 Bytes sind für den Hersteller reserviert. Dadurch ist es möglich den Hersteller der Netzwerkkarte zu ermitteln. Viele Programme mit denen das Mitlesen im Netzwerk möglich ist, sind in der Lage diese 3 Bytes bereits zu interpretieren.

[11] Sie werden diesem Umstand sehr dankbar sein, falls Sie eine Software besitzen, deren Lizenz an die MAC-Adresse gebunden ist und die Netzwerkkarte wegen eines Defektes getauscht werden muss.

[12] Zur 'totalen Verwirrung' hat Novell unter derselben Bezeichnung 802.3 wiederum anders aufgebaute Ethernet-Frames eingeführt.Diese von Novell verwendeten Frames werden jetzt auch *802.3raw* genannt. Novell verwendet inzwischen auch tatsächliche 802.3 Frames, die proprietäre Variante sollte langsam aussterben.

Abbildung 15.6: Ethernet Frame



15.3.4 MTU Maximum Transfer Unit

Die MTU ist die maximale Anzahl von Daten, welche in einem Paket über ein physikalisches Netzwerk übertragen werden können. Wie groß die MTU ist, hängt von der zugrundeliegenden Netzwerkphysik ab. Der administrative Overhead ist umso kleiner, je größer die Datenpakete sind. Der Durchsatz steigt also bei größerer MTU. Werden die Daten aber über ein unzuverlässiges Medium geschickt, muss aber bei jedem Übertragungsfehler das komplette Paket erneut übertragen werden.

Man wählt deshalb für ’bessere’ Übertragungsstrecken eine größere MTU als für ’schlechtere’. So beträgt die MTU für das Loopback-Interface bei Solaris 8232 Bytes, für die Ethernet-Schnittstelle aber nur 1500 Bytes. Bei Übertragungen über Telefonleitungen ist die MTU noch kleiner.

Sitzt man zu Hause am PC um im Internet zu ’surfen’ so hat man meist eine relativ langsame Modemverbindung zum nächsten Provider. In diesem Abschnitt wird mit einer geringen MTU übertragen. Bis zum eigentlichen Server wird die MTU unter Umständen mehrfach geändert. Überschreitet ein Datenpaket die MTU, muss das Paket ’fragmentiert’ werden und am Ziel wieder zusammengefügt werden.

15.3.5 Ethernet Fehlererkennung

Wird ein Datenpaket empfangen, so führt das Ethernet Interface eine Plausibilitätskontrolle durch. Folgende Überprüfungen werden durchgeführt:

Runts (auch *Short Frames*) Ist das empfangene Datenpaket kleiner als 46 Bytes so wird das Paket verworfen, da es zu kurz ist. Ursache sind i.A. fehlerhafte Netzwerkadapter.

Jabbers Ist das komplette Paket größer als die MTU, ist es zu groß und wird ebenfalls verworfen. Jabber kann durch defekte Netzwerkkarten oder auch durch fehlerhafte Treiber verursacht werden.

Bad CRC Stimmt die Prüfsumme am Ende des Paketes nicht, so sind die Daten nicht gültig und werden verworfen.

Es gibt auch noch folgende Arten von Störungen:[13]

Late Collision (*verspätete Kollission*) Kollissionen müssen normalerweise innerhalb 512 Bitzeiten auftreten. Ursache dafür können defekte Netzwerkkarten sein, aber häufig führen nicht eingehaltene Regeln für die jeweilige Topologie dazu. Es ist unbedingt darauf zu achten, dass maximale Segmentlängen (inklusive Leitungen zum Netzwerkadapter), maximale Anzahl von Repeatern, etc. eingehalten werden. Missachtung kann zu sporadischen bis dauerhaften Problemen führen.

Ghost Frames (*Geisterrahmen*) Damit sind Datenrahmen gemeint, die wie solche aussehen, aber sinnlosen Inhalt haben. Ursache dafür können Störeinflüsse (elektromagnetische Einstrahlungen), Potentialprobleme (Ausgleichsströme) oder fehlerhafte aktive Komponenten sein. Eventuell kann auch ein Repeater aus Störsignalen einen Rahmen aufbauen und versenden.

[13] Ich liste diese Störungen getrennt auf, da sie für die Zertifizierungsprüfung bei SUN nicht relevant sind (nicht in den Schulungsunterlagen genannt).

16 Netzwerk Protokolle

16.1 ARP und RARP

ARP (Address Resolution Protocol) dient dazu, anhand einer Adresse der OSI-Schicht 2 eine Adresse der OSI-Schicht 3 zu erfragen. *RARP* (Reverse Address Resolution Protocol) macht genau das Umgekehrte - die Schicht-3 Adresse ist bekannt und man benötigt die Schicht-2 Adresse.[1]

In der Praxis geht es meist um die MAC-Adressen (Ethernet) und die IP-Adressen. Wenn eine Station im Ethernet Daten zu einem anderen Rechner senden will, so ist ihr zu Beginn nur die IP-Adresse des Ziels bekannt[2]. Für die Übertragung der Daten in einem Ethernet-Paket wird aber die MAC-Adresse benötigt. Um dieses Problem zu lösen, wird ein 'ARP-Request Paket' mit der Zieladresse FF:FF:FF:FF:FF:FF (also eine Broadcast-Adresse) an alle Teilnehmer des Netzes geschickt. In diesem Paket ist die eigenen MAC- und IP-Adresse und die IP-Adresse des gewünschten Rechners enthalten. Jeder Rechner im Netz muss nun (außer die ARP-Funktion der Netzwerkkarte wurde explizit deaktiviert) das Paket untersuchen, ob es an ihn gerichet ist (IP-Adresse des Zieles). Jener Rechner, dessen IP-Adresse mit der Adresse im ARP-Paket übereinstimmt, antwortet mit einem 'ARP-Reply Paket'. Das 'ARP-Reply Paket' enthält nun ein mit allen Adressen (MAC und IP) vollständig ausgefülltes Paket. Nun erfährt die sendewillige Station die MAC-Adresse des gewünschten Zieles und kann das Datenpaket mit der richtigen MAC-Adresse versehen und auf die Reise schicken.

```
ARP-Request Paket wird von der sendewilligen Station geschickt:
Ethernet II
Destination: ff:ff:ff:ff:ff:ff
Source: 00:00:21:22:86:fe
Type: ARP (0x0806)
Trailer: 20202020202020202020202020202020...
Address Resolution Protocol (request)
Hardware type: Ethernet (0x0001)
Protocol type: IP (0x0800)
Hardware size: 6
Protocol size: 4
Opcode: request (0x0001)
Sender hardware address: 00:00:21:22:86:fe
Sender protocol address: 192.168.77.177
Target hardware address: 00:00:00:00:00:00
Target protocol address: 192.168.77.1
```

[1] Referenz für ARP und RARP sind RFC826 und RFC903.
[2] Die IP-Adresse wiederum erhält die Anwendung aus der Datei /etc/inet/hosts, über NIS oder DNS.

Tabelle 16.1: ARP Protokolldaten

0		14	31
Hardware-Typ (Schicht 2)			Protokolltyp (Schicht 3)
Adresslänge (Schicht 2)	Adresslänge (Schicht 3)		Operation
Absenderadress (Schicht 2)			
Absenderadresse (Schicht 3)			
Zieladresse (Schicht 2)			
Zieladresse (Schicht 3)			

Der oben beschriebene ARP-Datensatz wird im Nutzdatenbereich des MAC-Protokolls übertragen.

```
unaufbereitetes Datenpaket:
0000  ff ff ff ff ff ff 00 00 21 22 86 fe 08 06 00 01   ........!"......
0010  08 00 06 04 00 01 00 00 21 22 86 fe c0 a8 4d b1   ........!"....M.
0020  00 00 00 00 00 00 c0 a8 4d 01 20 20 20 20 20 20   ........M.
0030  20 20 20 20 20 20 20 20 20 20 20 20

---------------------------------------------------------------------

ARP-Reply Paket sendet die 'angesprochene' Station zurück
Ethernet II
Destination: 00:00:21:22:86:fe
Source: 00:c0:95:ee:ca:04
Type: ARP (0x0806)
Address Resolution Protocol (reply)
Hardware type: Ethernet (0x0001)
Protocol type: IP (0x0800)
Hardware size: 6
Protocol size: 4
Opcode: reply (0x0002)
Sender hardware address: 00:c0:95:ee:ca:04
Sender protocol address: 192.168.77.1
Target hardware address: 00:00:21:22:86:fe
Target protocol address: 192.168.77.177

unaufbereitetes Datenpaket:
0000  00 00 21 22 86 fe 00 c0 95 ee ca 04 08 06 00 01   ..!"............
0010  08 00 06 04 00 02 00 c0 95 ee ca 04 c0 a8 4d 01   ..............M.
0020  00 00 21 22 86 fe c0 a8 4d b1                     ..!"....M.
```

Damit nicht vor jedem Verbindungsaufbau ein ARP-Request verschickt werden muss, existiert ein '*ARP-Cache*' in dem die Adresszuordnungen gespeichert werden. Sowohl der Rechner, der ein ARP-Reply erhält, als auch jener Rechner, der auf einen ARP-Request antwortet, tragen sich die Daten in die ARP-Tabelle des Kernels ein. Damit diese Tabelle nicht zu groß wird und auch damit nicht

veraltete Daten[3] in der Tabelle verbleiben, entfernt der Kernel längere Zeit (5 Minuten üblicherweise) nicht benützte Einträge. IP-Adressen, für die bereits ein ARP-Request versandt, aber noch kein Reply erhalten wurde, stehen unvollständig in der ARP-Tabelle.

Mit Hilfe des Kommandos '*arp*' kann man die ARP-Tabelle auslesen, Eintragungen löschen oder statische Eintragungen durchführen.

arp -a | -s *hostname mac-addr* | -d *hostname* | -f *filename*

-a zeige alle Einträge der ARP-Tabelle
Die Flags in der ausgegebenen Tabelle haben folgende Bedeutung:

S statischer Eintrag

P 'published' Eintrag (siehe Option '-s')

M 'mapped' Eintrag (typisch für Multicast-Eintrag)

U unvollständiger, noch nicht aufgelöster Eintrag

-s *hostname mac-addr [temp|pub]* trägt Zuordnung IP- zu MAC-Adresse statisch ein

Optional kann noch eine der Zusätze *temp* oder *pub* angegeben werden.

-d *hostname* löscht ARP-Eintrag für *hostname*

-f filename liest statische Adresszuordnungen aus der Datei *filename* (z.Bsp. aus /etc/ethers)

```
# arp -a
Net to Media Table
Device IP Address Mask             Flags  Phys Addr
------ ---------- ---------------  ------ ------------------
le0    rhino      255.255.255.255         08:00:20:75:8b:59
le0    mule       255.255.255.255  SP     08:00:20:75:6e:6f
le0    horse      255.255.255.255  U
le0    224.0.0.0  240.0.0.0        SM     01:00:5e:00:00:00
```

Wann wird '*RARP*' benötigt? Eine Diskless-Station, welche ihr Betriebssystem nicht lokal gespeichert, sondern über das Netzwerk lädt, kennt nach dem Einschalten nur ihre MAC-Adresse, aber nicht die eigene IP-Adresse. Um diese zu erfahren, schickt es ein sogenanntes '*REVARP-Request* Paket' mit der Broadcastadresse FF:FF:FF:FF:FF:FF und der eigenen MAC-Adresse an alle. Ein Rechner muss nun als RARP-Server konfiguriert sein, damit solche Request-Pakete beantwortet werden können.

Am RARP-Server läuft ein Dämonprozess namens *in.rarpd*, welcher eingehende REVARP-Requests beantwortet. Die dafür nötigen Daten entnimmt der Prozess der Datei */etc/ethers*. */etc/ethers* enthält eine Tabelle, die MAC-Adressen und deren zugehörige IP-Adressen enthält.

in.rarpd [-a] [-d]

-a beantwortet revarp-Requests an allen verfügbaren Netzwerk-Schnittstellen

-d aktiviert Debug-Modus für Fehlerdiagnose

[3] Zum Beispiel ändert sich die MAC-Adresse eines Rechners, wenn dessen Netzwerkkarte wegen eines Defektes getauscht werden muss.

16.1.1 Fehlersuche bei RARP-Server:

Falls eine Diskless-Station nicht startet oder die Installation mittels ’Jump Start’ nicht funktioniert, müssen folgende Voraussetzungen überprüft werden:

- Läuft der Dämon in.rarpd?
- Ist die MAC-/IP-Adresse der Station in der Datei /etc/ethers eingetragen?
- Ist die Station in der Datei /etc/inet/hosts eingetragen?
- Ist die Ursache noch nicht gefunden, ist es am effizientesten mitzulesen z.Bsp. mit ’*snoop -v rarp*’:
  Schickt die Station ein REVARP-Request Paket? Wird dieses Paket vom RARP-Server beantwortet?

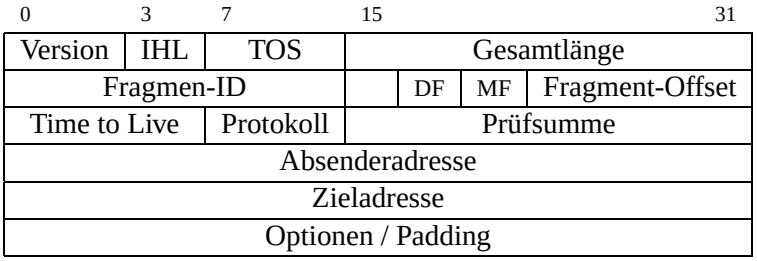
16.2 Internet Layer

Der *Internet Layer* entspricht im ISO/OSI 7-Schichtenmodell dem *Network Layer* (Schicht 3). Im Internet wird auf dieser Schicht das IP-Protokoll zur Datenübertragung verwendet. Protokolle wie NETBEUI oder IPX, die sich ebenfalls auf dieser Ebene befinden, haben im Internet keine Bedeutung. Spricht man von *IP*, so ist meist das Internet Protocol Version 4 (*IPv4*) gemeint. Seit Jahren schon gibt es aufgrund von Mängeln bzw. Einschränkungen[4] von IPv4 Diskussionen, das Protokoll durch das *Internet Protocol - Next Generation* (*IPnG*) bzw. (*IPv6*) zu ersetzen.

Das Internet Protokoll ist in den Kernel integriert. Es stellt 2 Dienste zur Verfügung:

- Fragmentierung und Wiederzusammensetzung (*Reassembly*) von Fragmenten für Protokolle in darüberliegende Schichten. Siehe auch Kapitel 15.3.4auf Seite 164.
- Routing-Funktion um Daten zu senden. Siehe Kapitel 16.3 auf Seite184.

16.2.1 IP-Paket Header



Version Zur Zeit sind nur die beiden Versionen IPv4 und IPv6 definiert. Eine Unterscheidung der beiden Protokolle IPv4 oder IPv6 erfolgt allerding schon auf MAC-Ebene anhand der Protocol-Identifier (0x800 für IPv4 bzw. 0x86dd für IPv6).

IHL Internet Header – Da der IP-Header aufgrund von Optionen unterschiedlich lang sein kann, wird im Internet Header die Headerlänge in Vielfachen von 32 Bit angegeben. Der kleinste möglich Wert ist 5 (keine Option – also 20 Byte großer Header), der größte Wert beträgt 15 (60 Bytes langer Header).

TOS Type of Service – Dies 8 Bits werden für Informationen wie Dringlichkeit und Zuverlässigkeit des Datagramms verwendet. In RFC 2474 wurde die Bedeutung in *Differentiated Services Codepoint* geändert und bestimmt das Weiterleitungsverhalten.

Gesamtlänge Die Gesamtlänge gibt die Größe des gesamten IP-Paketes an. Die 16 Bits beschränken die maximale Länge auf 65 535 Bytes. RFC 791 schreibt vor, dass jeder IP-fähige Rechner zumindest eine Länge von 576 Bytes unterstüzten muss.

[4] Eines der Probleme ist, dass der Adressraum (4 Bytes für Netz- und Host-Adresse) knapp wird. Andere Mängel treten immer wieder in Form von Sicherheitslücken zu Tage, welche von Hackern ausgenützt werden können, um sich Zugang zu Rechnern zu verschaffen.

Fragment-ID Mussten Datenpakete auf dem Weg zum Zielrechner fragmentiert werden, so kann der Zielrechner anhand der Fragment-ID und der Absenderadresse das ursprüngliche Datagramm wieder korrekt zusammenfügen. Alle Fragmente eines IP-Datagrammes enthalten dieselbe Fragment-ID.

Flags 3 Bits stehen für Flags zur Verfügung, wobei 1 Bit derzeit unbenützt bleibt.

DF (Don't Fragment) Damit wird eine Fragmentierung verhindert mit dem Risiko, dass das Datagramm aufgrund einer niedrigeren MTU (Maximale Transfer Unit) nicht mehr weitertransportiert werden kann.

MF (More Fragments) Dieses Bit zeigt an, dass fragmentiert wurde und diesem Datenpaket noch weitere Fragmente folgen. Wurde fragmentiert, ist also in allen Fragmenten außer dem letzten dieses Bit gesetzt.

Fragment-Offset Der Fragment-Offset gibt an, an welcher Stelle die Nutzdaten dieses Paketes vor der Fragmentierung im ursprünglichen Datagramm standen, damit der Zielrechner wieder das Original-Datagramm zusammenfügen kann. Das Feld hat eine Größe von 8 Bytes und erlaubt deshalb eine maximale Anzahl von 8192 Fragmenten. Die Länge aller Fragmente außer dem Letzten muss ein Vielfaches von 8 Bytes betragen.

Time To Live Damit Datenpakete in einem komplexen und ev. falsch konfigurierten Netz nicht endlos kreisen, wurde ein Mechanismus geschaffen, der ein Paket altern lässt und am Ende seiner Lebensdauer verworfen wird. Dies wird durch die *TTL* (*Time to Live*) realisiert. Der Absender setzt einen bestimmten Wert (max. 255). Wenn das Paket durch einen Router geschleust wird, vermindert der Router die TTL jeweils um 1. Erreicht die TTL den Wert 0, so wird das Paket verworfen.

Protokoll Diese 8 Bits dienen der Identifikation des Protokolls in der nächsthöheren Schicht. Einige typische Werte dafür sind: 0x01 für ICMP, 0x02 für Gateway-Protokoll, 0x06 für TCP, 0x11 für UDP, 0x22 für XNS etc.

Prüfsumme Die *Header Checksum* (16 Bits) dient dazu, die korrekte Übertragung des Headers zu überprüfen (die Nutzdaten werden damit noch nicht geprüft). Da die TTL von jedem Router verändert wird, muss auch die Prüfsumme neu errechnet werden. Als Prüfsumme wird das Einerkomplement der Summe aller 16-Bit Werte gebildet.

Absender- und Zieladresse Dies sind die jeweils 32 Bit langen IP-Adressen des Quell- und des Zielrechners.

Optionen / Padding An dieser Stelle können mögliche Optionen des IP-Protokolls hinterlegt werden. Die Optionen werden immer mit Binär 0 bis zu einer Wortgrenze (32 Bits) aufgefüllt.

16.2.2 Internet Adresse / IP Adresse

Die *Internet Adresse* ist eine 32-Bit Zahl. Sie wird im Allgemeinen durch vier durch einen Punkt voneinander getrennten Dezimalzahlen angegeben. Die Zahlen können Werte von 0 bis 255 betragen. Sie wird auch *IP Adresse* genannt. Diese Adresse wird bei der Installation des Betriebssystemes vergeben.

Die IP-Adresse beinhaltet eine Netzwerk-Adresse und eine Host-Adresse. Die Netzwerk-Adresse identifiziert ein lokales Netzwerk. Sie wird durch die linken Zahlen der IP-Adresse beschrieben. Will man mit dem Internet direkt verbunden werden, so muss diese Netzwerk-Adresse eindeutig sein. Es darf weltweit kein anderes Netzwerk mit derselben Netzwerk-Adresse im Internet geben.

Die Host-Adresse wird durch die rechten Zahlen der IP-Adresse beschrieben. Sie identifizieren jeden Rechneranschluss innerhalb eines Netzwerkes. Die Hostadresse darf weder alle Bits gleich Null (0) noch alle Bits gleich (1) haben, da diese Werte eine Sonderfunktion besitzen (*Broadcast Address*).

Um verschieden große Netze sinnvoll adressieren zu können hat man unterschiedliche Netzwerk-Klassen definiert:

16.2.2.1 Class A - Netzwerk

Das *Class A Network* wird bei extrem großen Netzwerken verwendet. Jedes Netzwerk kann 16 Millionen Rechner umfassen. Die Netzwerkadresse umfasst 8 Bits, während das erste Bit (ganz links) den Wert ’0’ hat. Diese Netzwerkklasse kann somit 127 (1 – 127) Netze und je Netz 16 Millionen Hosts adressieren.

16.2.2.2 Class B - Netzwerk

Ein *Class B Network* kann etwa 16 000 Netze und 65 000 Hosts je Netz adressieren. Die Netzadresse umfasst 16 Bits, wobei der Wert der ersten beiden Bits ’10’ lautet.

16.2.2.3 Class C - Netzwerk

Ein *Class C Network* ist für kleine Netze gedacht. Es erlaubt die Adressierung von bis zu 254 Hosts je Netz. Dafür stehen damit Netzwerkadressen für über 2 Millionen Class-C Netze zur Verfügung. Die ersten 3 Bits der Netzwerkadresse sind fest definiert: Deren Inhalt lautet immer ’110’.

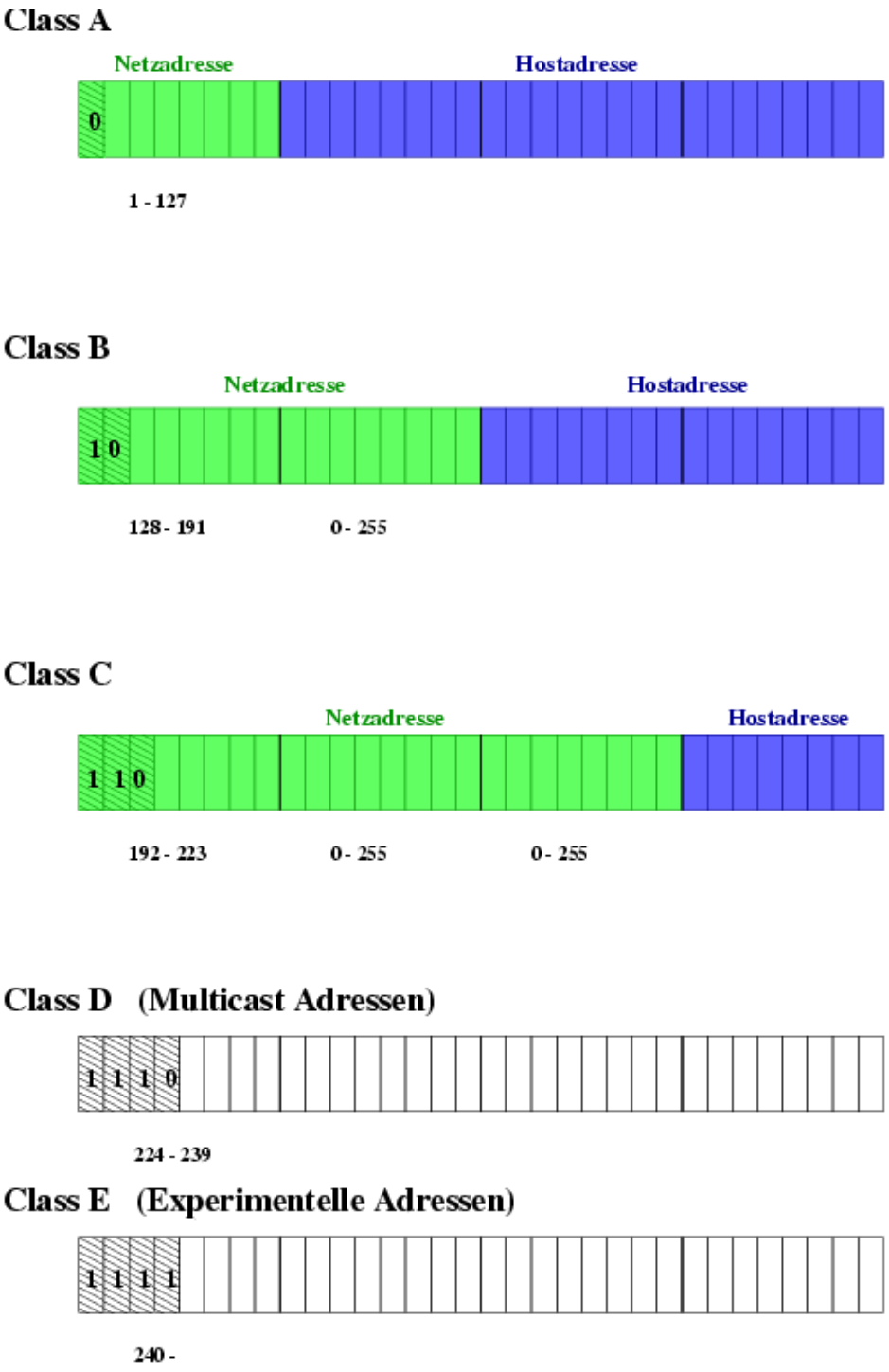
In den RFC’s (Request for Comment) finden sich noch weitere Vorschläge bzw. bereits definierte zusätzliche Klassen.

Class D Netzadressen werden für *Multicasting* verwendet. Man kann damit mehrere Hosts zugleich ansprechen. Solche Adresssen haben in den erste 4 Bits den Wert ’1110’. Weiters gibt es noch **Class E** Adressen, welche für Experimentelle Zwecke reserviert sind. Diese Adressen beginnen mit ’1111’.

16.2.2.4 Reservierte Netzwerk- und Host-Adressen

IPv4 Adresse	Beschreibung
127.x.x.x	reserviert für Loopback
x.0.0.0	alle Bits der Hostadresse ’0’ ’alte’ Broadcastadresse
x.255.255.255	alle Bits der Hostadresse ’1’ Broadcastadresse
0.0.0.0	wird verwendet, wenn System die eigene IP-Adresse noch nicht kennt z.Bsp. bei RARP und BOOTP
255.255.255.255	’Generic’ Broadcast

Abbildung 16.1: Internet Netzwerk Klassen



Durch die rasche Verbreitung des Internets kam es zu Engpässen von Netzwerkadressen. Eine rasche Umstellung auf das neue Protokoll IPv6 war nicht in Sicht und so wurde mitte der 90-er Jahre der Vorschlag gemacht, in jeder Adress-Klasse (A, B und C) einen bestimmten Bereich für private Netze zu reservieren.[5] Die vorgeschlagenen Adressen können in privaten Netzen verwendet werden und kollidieren nicht im Internet, da diese im Internet nicht weitergeroutet werden. Durch Anwendung von *NAT* (Network Address Translation)[6] ist es dennoch möglich aus einem Privatnetz Zugriff ins Internet zu erlangen. Als Nebeneffekt sind die Rechner des Privatnetzes auch besser vor unerwünschten Zugriffen aus dem Internet geschützt (die IP-Adressen der Rechner im Privatnetz werden ja nicht geroutet).

Reservierte Adressen für Privatnetze:

Klasse	Netz/Subnetmask	niedrigste / höchste Adresse
A	10/8	10.0.0.0 — 10.255.255.255
B	172.16/12	172.16.0.0 — 172.31.255.255
C	192.168/16	192.168.0.0 — 192.168.255.255

16.2.3 IPv4 Netzmasken

Um Datenpakete korrekt routen zu können, muss der Netzanteil einer IP-Adresse exakt definiert sein. Dies geschieht durch die Netzwerkmaske. Die Netzwerkadresse wird bestimmt indem die IP-Adresse durch logisches UND mit der Netzmaske errechnet wird. Für die 3 oben genannten Netzwerk-Klassen lautet die Netzmaske also:

Klasse A – 255.0.0.0

Klasse B – 255.255.0.0

Klasse C – 255.255.255.0

Beispiel für die Errechnung der Netzadresse:

```
IPv4 Adresse (dezimal)    171.63.14.3
IPv4 Adresse (binär)      10101011 00111111 00001110 00000011
Klasse B Netzmaske (dez.) 255.255.0.0
Klasse B Netzmaske (bin.) 11111111 11111111 00000000 00000000
logische UND-Verknüpfung:
IPv4 Adresse (binär)      10101011 00111111 00001110 00000011
UND Netzmaske             11111111 11111111 00000000 00000000
-----------------------------------------------------------
Netzwerk-Adresse (binär)  10101011 00111111 00000000 00000000
Netzwerk-Adresse (dezimal)   171      63       0        0
```

[5] Details können dem RFC1918 entnommen werden. http://www.rfc.net/

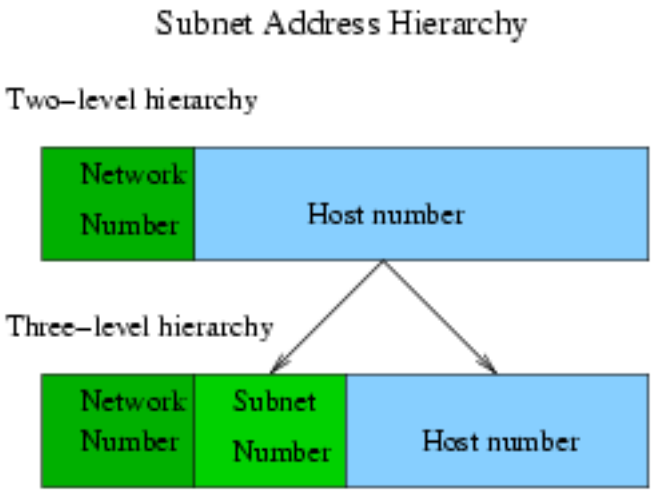
[6] Bei NAT passiert grob gesagt Folgendes: Es wird nur jenem Rechner, der den direkten Zugang zum Internet besitzt eine 'offizielle' Internet-Adresse vergeben. Alle Rechner im Privatnetz erhalten eine Adresse laut RFC1918. Beim Zugriff auf einen Host im Internet wird die Adresse des Rechners im Privatnetz in die offiziell zugewiesene Adresse übersetzt (= NAT). Das heißt, alle Rechner im Privatnetz arbeiten mit derselben 'Internet'-Adresse.

In vielen Fällen ist eine feinere Abstufung als es durch die Klassen A, B und C möglich ist, erforderlich. Das erreicht man, indem in der Netzmaske zusätzliche Bits auf '1' gesetzt werden. Auf diese Art 'verkleinerte' Netze nennt man '*Subnetze*'.[7]

Warum setzt man Subnetze ein:

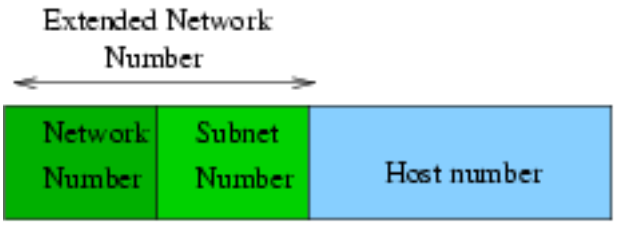
- Subnetting ermöglicht, eine Netzwerkadresse einer Klasse in viele unterschiedliche Netze zu unterteilen. Das ist zwingend erforderlich, wenn Router zwischen den physikalischen Netzen eingesetzt werden.
- Reduzierung des Netzwerk-Verkehrs durch Verkleinerung der Broadcast-Domänen
- Erhöhung der Sicherheit durch Einschränkung der Zugriffsmöglichkeit.[8]
- Strukturierung des Netzes in Bezug auf die Geographie, Abteilung oder Netzwerk-Protokolle

Vor allem Adressen der Klasse A bieten die Möglichkeit unrealistisch viele Hosts je Netz zu definieren. Netze mit 'zig-tausenden Hosts sind in der Praxis nicht sinnvoll. Beim *Subnetting* wird die Hostadresse durch Setzen von zusätzlichen Bits der Subnetzmaske verkleinert und stattdessen die Netzadresse erweitert.Dadurch erhält man eine hohe Flexibilität die Größe des Netzwerkanteiles an der Adresse nach eigenen Bedürfnissen zu bestimmen.



Internet-Router verwenden nur die *Network number* der Zieladresse um den Weg zum Ziel zu finden (auch wenn dort dann Subnetze verwendet werden). Router, die sich in einer Subnet-Umgebung befinden, verwenden die *Extended Network Number*(Erweiterte Netzwerk Adresse).

[7] Nähere Details findet man in RFC950.
[8] Die Erhöhung der Sicherheit ist nur bedingt, da dies nur für Laien ein Hindernis darstellt.



Beispiel für die Errechnung der Extended Subnet Number:

```
                          172      16       25        3
Two-level hierarchy:    10101100 00010000 00011001 00000011
Subnet mask (dez.)        255      255      252        0
Subnet mask (bin.)      11111111 11111111 11111100 00000000
                        |---------------| |----|
                          Default Mask  Ext.Mask
Three-level hierarchy:  10101100 00010000 00011001 00000011
                        |-Extended network nr.-|
Extended Network Number   172      16       24        0

Mögliche Hostadressen in diesem Beispielnetz:
172.16.24.1 bis 172.16.27.254
```

Im obigen Beispiel wurde der für diese Klasse–C Adresse üblichen Netzmaske ’255.255.0.0’ die Netzmaske ’255.255.252.0’ gewählt. Der Anteil für die Netzadresse wurde dadurch um 6 Bits vergrößert. In jedem der definierbaren Netze können mit den verbleibenden 10 Bits 1022 Hosts[9] addressiert werden.

[9] $2^{10} = 1024$. Davon müssen die beiden Adressen abgezogen werden, bei denen kein Bit bzw. alle Bits gesetzt sind. Dies sind reservierte Adressen für Broadcast.

Übliche Werte für Subnetzmasken sind (siehe auch Tabellen auf Seite 178):

128 1 zusätzliches Bit für die Netzadresse
192 2 Bits
224 3 Bits
240 4 Bits
248 5 Bits
252 6 Bits
254 7 Bits
255 8 Bits

Die Broadcast-Adresse ist jene Adresse, bei der sich alle Hosts eines Netzes angesprochen fühlen. Sie wird errechnet durch logisches NOT der Netzwerkmaske und logisches OR der Netzwerkadresse.

Für obiges Beispiel ergibt sich damit:

```
IPv4 address    172.16.25.3    10101100 00010000 00011001 00000011
AND Subnetmask  255.255.252.0  11111111 11111111 11111100 00000000

Network number  172.16.24.0    10101100 00010000 00011000 00000000
NOT subnetmask  255.255.252.0  11111111 11111111 11111100 00000000

                               00000000 00000000 00000011 11111111
OR Network number 172.16.24.0  10101100 00010000 00011000 00000000

Broadcast number 172.16.27.255 10101100 00010000 00011011 11111111
```

Es gibt 2 unterschiedliche Schreibweisen, um Netzwerkadressen anzugeben:
172.16.24.0/255.255.252.0 oder 172.16.24.0/22
Die erste Schreibweise ist offensichtlich – der erste Teil gibt die Netzadresse an, der zweite Teil (hinter dem Schrägstrich) die Netzmaske. Die zweite Art beschreibt die Netzmaske durch Angabe der Anzahl der gesetzten Bits.

RFC 950 empfiehlt die Verwendung von zusammenhängenden Netzmasken. Das sind Masken, deren Bits von links nach rechts durchgehend auf ’1’ gesetzt sind. Es wäre denkbar, Netzmasken zu verwenden, welche nicht zusammenhängend sind. Zum Beispiel wäre die folgende Netzmaske möglich:

```
dezimal: 255.255.210.0
binär  : 11111111 11111111 11010010 00000000
```

Allerdings erschweren solche Netzmasken die Arbeit unnötig, da die Berechnung von Host- und Broadcastadressen komplizierter wird. Außerdem werden solche Subnetzmasken nicht von allen Betriebssystemen unterstützt.

16.2.3.1 Variable Subnetmasken

Die Abteilungen einer Organisation sind selten alle gleich groß. Es ist deshalb eine schwierige Aufgabe, die ’richtige’ Größe der Subnetzmaske zu wählen. Um dieses Problem zu lösen, ist es möglich

unterschiedliche Maskenlängen zu wählen. Mit Hilfe dieser *Variable Length Subnet Masks* (*VLSM*) kann die jeweils ideale Netzwerkgröße gewählt werden. Für Abteilungen mit vielen Hosts, wird eine kürzere Maske gewählt (Hostanteil ist größer), für Netze mit nur wenigen Hosts wählt man eine lange Maske.

VLSM erlaubt auch eine rekursive Unterteilung eines Netzwerk-Adressbereiches. Angenommen, ein Konzern hat mehrere Niederlassungen in unterschiedlichen Städten. Aufgrund der Größe wird die Netzadresse 10.0.0.0/8 der Klasse A gewählt. Die europäischen Niederlassungen bekommen das Subnetz 10.64.0.0/10. Dieses Subnetz wird nun noch weiter unterteilt, um die Städte abzubilden: Madrid – 10.76.0.0/14, München – 10.92.0.0/14 etc. In München gibt es ein Produktionswerk mit der Netzadresse 10.93.0.0/17, ein Bürogebäude mit der Netzadresse 10.94.0.0/17 und ein Verkaufshaus mit der Adresse 10.92.128.0/17. Das Netz des Bürogebäudes wird in weitere Subnetze für Forschung (10.94.16.0/20), Buchhaltung (10.94.32.0/20) usw. unterteilt.[10]

Die Vorteile der variablen Subnetzmasken sind

- Effizientere Nutzung der vorhandenen IP-Adressen.
- Geschickte Wahl der Subnetzadressen kann die Menge der Routing-Informationen im Backbone erheblich reduzieren.

Moderne Routing-Protokolle wie *Open Shortest Path First* (*OSPF*) und *Intra-Domain Intermediate System to Intermediate System* (*IS-IS*) unterstützen die Verwendung von variablen Subnetzmasken. Ältere Protokolle wie *RIP* unterstützen dies nicht.

16.2.3.2 Empfohlene Subnetzmasken für Klasse-B Netze:

Maske - dezimal	Maske - binär	Anzahl Subnetze	Anzahl Hosts je Subnetz
255.255.0.0	11111111 11111111 00000000 00000000	1	65534
255.255.128.0	11111111 11111111 10000000 00000000	2	32766
255.255.192.0	11111111 11111111 11000000 00000000	4	16382
255.255.224.0	11111111 11111111 11100000 00000000	8	8190
255.255.240.0	11111111 11111111 11110000 00000000	16	4094
255.255.248.0	11111111 11111111 11111000 00000000	32	2046
255.255.252.0	11111111 11111111 11111100 00000000	64	1022
255.255.254.0	11111111 11111111 11111110 00000000	128	510
255.255.255.0	11111111 11111111 11111111 00000000	256	254
255.255.255.128	11111111 11111111 11111111 10000000	512	126
255.255.255.192	11111111 11111111 11111111 11000000	1024	62
255.255.255.224	11111111 11111111 11111111 11100000	2048	30
255.255.255.240	11111111 11111111 11111111 11110000	4096	14

[10] Im Beispiel habe ich bewusst Subnetzmasken gewählt, welche nicht genau an Bytegrenzen gebunden sind, um die Möglichkeiten noch deutlicher zu machen. Da die Rechenarbeit mit solchen Netzmasken aber lästig ist, habe ich mir einen 'Netzmasken-Rechner' mit Hilfe von Perl geschrieben. Das Programm ist im Anhang (Seite 288) aufgelistet.

Maske - dezimal	Maske - binär	Anzahl Subnetze	Anzahl Hosts je Subnetz
255.255.255.248	11111111 11111111 11111111 11111000	8192	6
255.255.255.252	11111111 11111111 11111111 11111100	16384	2

16.2.3.3 Empfohlene Subnetzmasken für Klasse-C Netze:

Maske - dezimal	Maske - binär	Anzahl Subnetze	Anzahl Hosts je Subnetz
255.255.255.0	11111111 11111111 11111111 00000000	1	254
255.255.255.128	11111111 11111111 11111111 10000000	2	126
255.255.255.192	11111111 11111111 11111111 11000000	4	62
255.255.255.224	11111111 11111111 11111111 11100000	8	30
255.255.255.240	11111111 11111111 11111111 11110000	16	14
255.255.255.248	11111111 11111111 11111111 11111000	32	6
255.255.255.252	11111111 11111111 11111111 11111100	64	2

16.2.3.4 Konfiguration eines Subnetzes

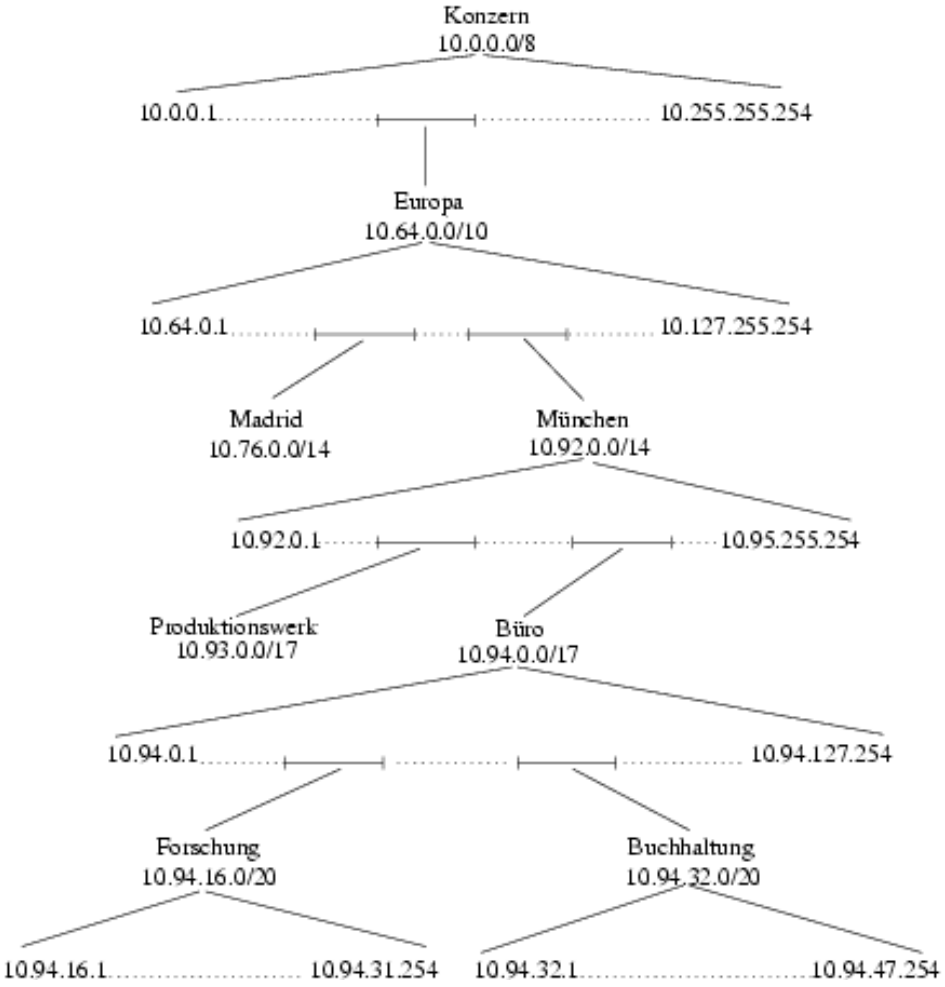
Bei Einsatz von Subnetzen müssen sowohl die betroffenen Rechner als auch Router entsprechend konfiguriert werden. Läuft der Router auf einem Rechner mit Solaris-Betriebssystem, kann folgende Anleitung verwendet werden.

Router Setup

1. Datei */etc/hostname.*xxn für das jeweilige Ethernet-Interface erzeugen und darin den Hostnamen eintragen.
2. In der Datei /etc/inet/hosts Hostnamen (und eventuelle zusätzliche Alias-Namen) und zugehörige IP-Adresse eintragen.
3. In die Datei */etc/inet/netmasks* die Netzwerk-Adresse und die dazugehörige Subnetzmaske eintragen. Zum Beispiel kann ein Eintrag wie folgt aussehen:
   `10.35.11.0  255.255.255.0`
4. (Optional) Falls gewünscht, kann den Subnetzen ein Name zugewiesen werden durch Eintrag in die Datei */etc/inet/networks*.
5. Reboot des Routers (bzw. Stop und Restart des Netzwerkes über Run-Scripte).
6. Kontrolle mittels Kommando '*ifconfig -a*'.

Host Setup Die Konfiguration der Hosts erfolgt durch Netzmasken-Einträge in der Datei */etc/inet/-netmasks*. Je nachdem, ob eine zentralisierte Administration mittels Name-Service (NIS, NIS+) im Einsatz ist oder nicht, muss jeder Rechner extra konfiguriert werden, oder die Änderung erfolgt nur einmal zentral am Masterrechner des Nameservices.

Abbildung 16.2: Beispiel für Variables Subnetting



Subnet-Konfiguration bei vorhandenem NIS Alle folgenden Schritte werden am NIS Master vorgenommen.

1. Alle Rechner in die Datei */etc/inet/hosts* mit zugehörigen IP-Adressen eintragen. Es ist darauf zu achten, dass Router mehrere IP-Adressen und Hostnamen haben.
2. Datei */etc/inet/netmasks* anpassen.
3. (Optional) Falls sprechende Namen für die Netzwerke gewünscht sind, könne die Namen in die Datei */etc/inet/networks* eingetragen werden.
4. Wechsel ins Verzeichnis */var/yp* und aufruf von *make*.
5. Restart des NIS-Masters.
6. Nachdem der NIS-Master wieder bereit ist, alle NIS-Slaves und NIS-Clients durchstarten.
7. Überprüfen durch das Kommando '*ifconfig -a*'

Subnet-Konfiguration bei vorhandenem NIS Alle folgenden Schritte werden am NIS+ Master vorgenommen.

1. Konfiguration der Hosts-Tabelle mittels '*Admintool*'. Zusätzliche Adressen und Hostnamen von Routern nicht vergessen.
2. Netmasks-Tabelle mit *Solstice AdminSuite* anpassen
3. (Optional) Netzwerknamen in die Networks-Tabelle mittels *Solstice AdminSuite* eintragen.
4. Restart des NIS+ Master.
5. Nachdem der NIS+ Master wieder bereit ist, alle NIS+ Slaves und NIS+ Clients durchstarten.
6. Überprüfen durch das Kommando '*ifconfig -a*'

Subnet Setup ohne Verwendung eines Nameservices Die folgenden Schritte müssen an allen Rechnern im Netz durchgeführt werden.

1. Datei */etc/inet/hosts* anpassen.
2. Eintrag der Netzwerkadresse und zugehöriger Netzmaske in die Datei */etc/inet/netmasks*.
3. (Optional) Eintrag eines Netzwerknamens in die Datei */etc/inet/networks*.
4. Reboot des Rechners.
5. Überprüfen durch das Kommando '*ifconfig -a*'.

16.2.3.5 Manuelle (temporäre) Konfiguration von Subnetzen für Testzwecke

Es ist möglich, ein Subnetz ohne Restart und ohne Änderung von Dateien zu konfigurieren. Das kann für Testzwecke sinvoll sein, ist aber nur temporär bis zum nächsten Restart wirksam. Es kann dabei zu Problemen mit den gerade laufenden Prozessen kommen, welche das Netzwerk benützen.

Die Änderung erfolgt durch folgende Kommandos (im Beispiel für das Netzwerk-Interface *le0*):

```
# ifconfig le0 down
# ifconfig le0 inet ip-addr -trailers netmask netmask broadcast + up
```

16.2.4 Netzwerk Interface Konfiguration

Die Konfiguration der Netzwerk-Schnittstellen erfolgt unter Solaris während des Systemstarts durch den Solaris-Kernel und durch Skripte, die vom init-Prozess aufgerufen werden.

Das Run-Script */etc/rcS.d/S30rootusr.sh* läuft während des Startups in der Phase des Single-User Modus. Es liest die zur Konfiguration der LAN-Schnittstelle benötigten Werte aus den Dateien */etc/hostname.xxn* und */etc/inet/hosts*[11] und aktiviert die Schnittstelle mittels *ifconfig*-Kommando.

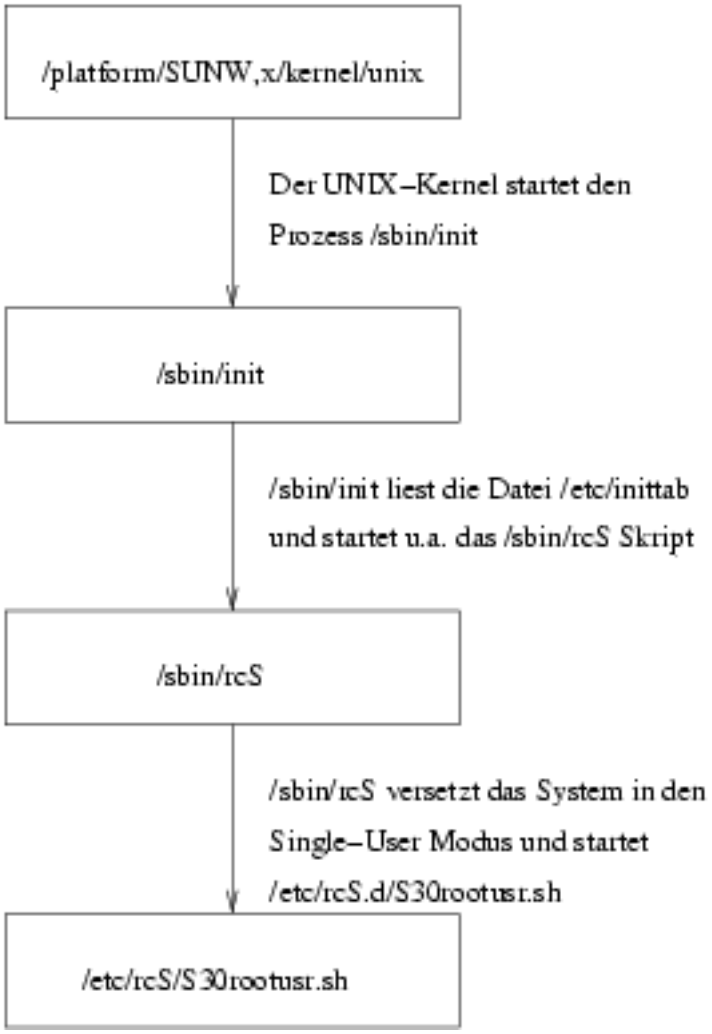
Die Netzwerk-Schnittstellen sind also auch im Single-User Modus aktiv und konfiguriert.

16.2.5 IPv6 (Internet Protokoll Version 6)

16.2.6 PPP (Point to Point Protokoll)

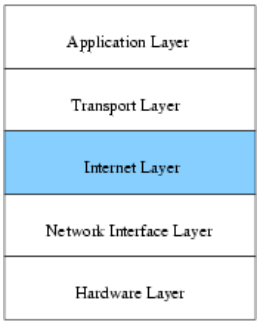
[11] Die Datei */etc/hosts* ist ein Link auf */etc/inet/hosts* und existiert aus Gründen der Kompatibilität zu früheren Systemen bzw. Programmen, die diese Datei noch im Verzeichnis */etc* erwarten.

Abbildung 16.3: Netzwerk Interface Konfiguration



16.3 Routing

Routing ist jener Mechanismus, der dafür sorgt, dass Datenpakete von einem Netzwerk in ein anderes weitergeleitet werden. Routing wird im Internet-Layer abgewickelt.



16.3.1 Routing Schemen

16.3.1.1 Table-Driven Routing

Jeder Rechner unterhält eine Tabelle im Kernel, die erreichbare Devices und Hosts enthält. Diese Routing-Tabelle des Kernels kann mit '*netstat -r*' aufgelistet werde.

16.3.1.2 Static Routing

Statische Routen sind bleiben permanent erhalten, bis sie entweder manuell entfernt werden, oder das System durchgestartet wird. Der geläufigste statische Eintrag ist jener, der die Verbindung in das lokale Netzwerk über das Netzwerkinterface beschreibt. Diese Routen in die lokalen Netze werden durch das Kommando *ifconfig* automatisch beim Konfigurieren der Netzwerkschnittstellen eingetragen. Dadurch ist gewährleistet, dass alle direkt ans lokale Netzwerk angeschlossenen Rechner erreichbar sind (sogar im Single-User Modus).

Statische Routen können auch manuell hinzugefügt werden. Diese Routen beschreiben Netzwerk-Ziele, die nicht direkt, sondern nur über andere Hosts (Router) erreicht werden können.

16.3.1.3 Dynamic Routing

Dynamisches Routing bezeichnet veränderliche Routingeinträge in der Routing-Tabelle. Verändert sich die Umgebung, werden die Routing-Tabellen der Hosts automatisch an die Veränderungen angepasst. Dazu laufen 2 Dämon-Prozesse, die sich um Empfang und Versand von Routing-Informationen sowie die Aktualisierung der Routing-Tabellen kümmern. Sie werden im Runlevel 2 im Runscript */etc/rc2.d/S69inet* gestartet:

- RIP ist implementiert im Prozess *in.routed* (Network Routing Dämon)

- RDISC ist implementiert im Prozess *in.rdisc* (Network Routing Discovery)

Die Idee des dynamischen Routings ist die, dass Router die ihnen bekannten Ziele (Rechner und Netzwerke) am Netz periodisch bekanntgeben während andere Router auf solche Routing-Informationen hören und deren Routing-Tabellen entsprechend aktualisieren.

16.3.1.4 Internet Control Messaging Protocol (ICMP) Redirects

ICMP ist ein eigenes Protokoll, das Kontroll- und Fehler-Informationen weiterleitet. Tritt ein Problem bei der Weiterleitung eines Paketes auf, so sendet der Router per ICMP-Paket eine Nachricht an den Absender-Host zurück.

Die bekanntesten Nachrichten, die per ICMP übertragen werden, sind *echo request* und *echo reply*. Diese werden im Kommando *ping* verwendet.

ICMP Redirects sind Informationen, die ein Host erhält, wenn der angesprochene Router einen 'besseren' Weg zum Ziel kennt. Dies tritt im Allgemeinen dann auf, wenn an einem Host eine *Default Route* (siehe Kapitel 16.3.1.5) eingetragen ist, der 'ideale' Weg zum gewünschten Ziel aber über einen anderen Router führen würde. Der Host, der diese ICMP-Redirects erhält, kann nun seine Routing-Tabelle aktualisieren und in Zukunft werden Pakete an die Zieladresse direkt über den Router am 'besten' Weg geschickt. Zu beachten ist, dass dieser Mechanismus zu sehr großen Routing-Tabellen führen kann.

Eine detaillierte Auflistung aller möglichen ICMP-Nachrichten ist im Kapitel 16.4 auf Seite 201 zu finden.

16.3.1.5 Default Routing

Es ist nicht erforderlich für alle Ziele einen Eintrag in die Routing-Tabelle vorzunehmen. Es kann ein *Default Router* angegeben werden, über den alle jene Pakete geschickt werden, die zu einem Ziel führen, dessen Weg dem Absender-Host nicht bekannt ist.

Ein *Default Router* wird in der Datei */etc/defaultrouter* durch Eintrag des Hostnamens oder der IP-Adresse des Routers bekanntgegeben. ACHTUNG! Existiert die Datei */etc/defaultrouter*, so verhindert ihre Existenz den automatischen Start der beiden dynamischen Routingprozessen *in.routed* und *in.rdisc*.

Vorteile des Default Routing

- Die Existenz der Datei */etc/defaultrouter* verhindert das Starten von zusätzlichen Routing-Prozessen.[12]
- Die Angabe eines Default Routers ist ideal, wenn nur ein Router existiert über den alle indirekten Ziele erreichbar sind.

[12] Ich bin mir nicht ganz sicher, ob das immer einen Vorteil bringt. Ich denke, das ist in Einzelfall zu unterscheiden. Jedenfalls wird es in den Schulungsunterlagen von SUN als Vorteil gesehen und ist ev. auch bei der Zertifizierungsprüfung als solcher zu nennen.

- Ein einzelner Default Routing Eintrag hält die Routing-Tabelle kleiner.
- Der Eintrag von mehreren Default Routern kann die Verfügbarkeit erhöhen, da ein 'single point of failure' eliminiert wird.
- Es können andere Routing Protokolle laufen.

Nachteile des Default Routing

- Der Default Routing Eintrag ist immer präsent, auch wenn dieser Router ausgefallen ist. Der Host ist nicht in der Lage andere mögliche Wege zu lernen.
- Alle Hosts im Netz müssen die Datei */etc/defaultrouter* korrekt konfiguriert haben. Das kann ein Problem in großen Netzen mit sich bringen.
- ICMP Redirects entstehen, wenn mehr als ein Router im Netz vorhanden ist.

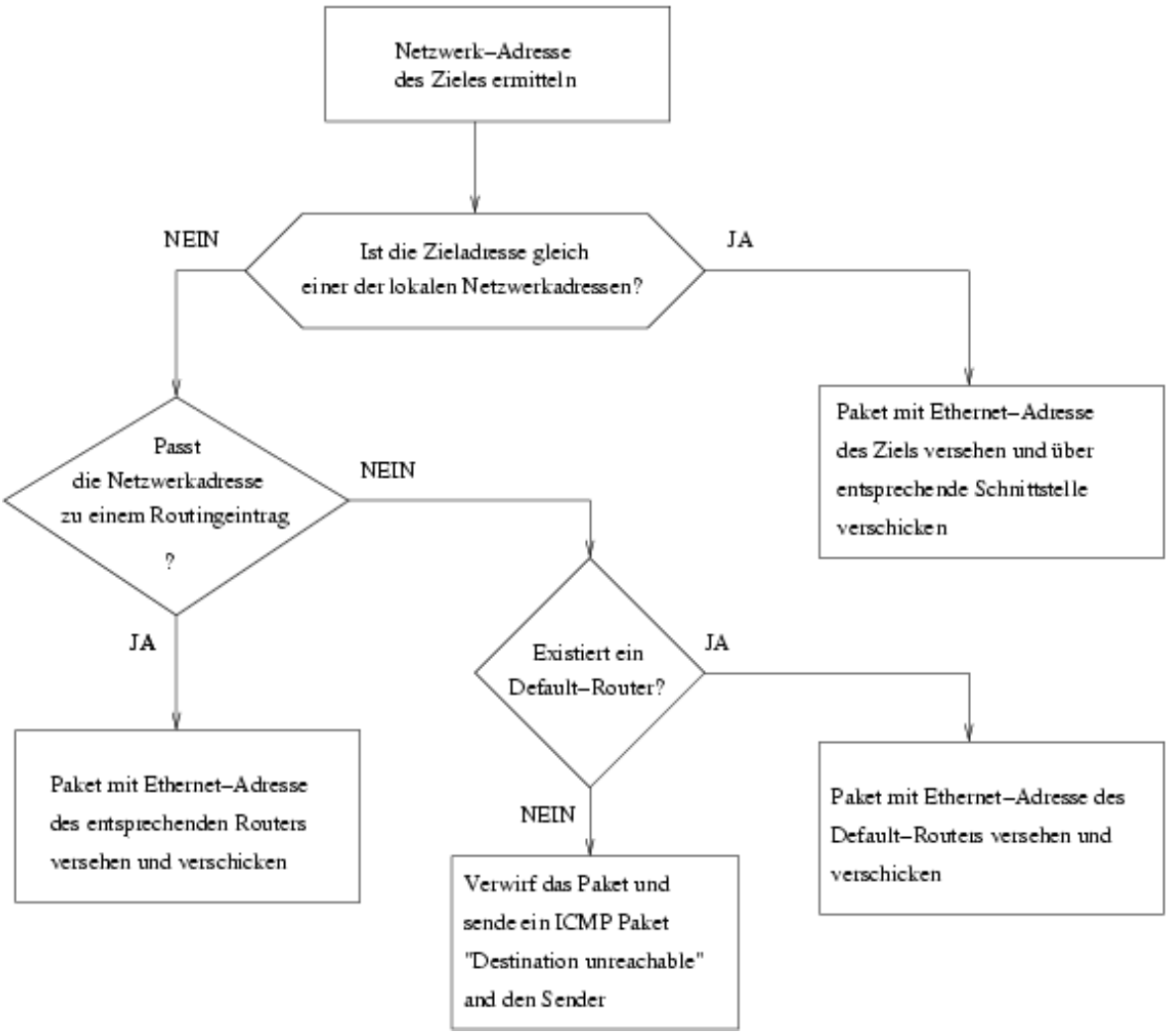
16.3.2 Routing Algorithmen

Ein Host, der als Router im Einsatz ist, muss beim Weiterleiten eines Datenpaketes die Zieladresse untersuchen und einige Entscheidungen bezüglich des weiteren Weges treffen. Dies geschieht im Solaris-Kernel.

- Überprüfung der Zieladresse, ob in einem lokalen Netz
  Der Kernel löst die Zieladresse aus dem Datenpaket und errechnet die Netzadresse. Diese Netzadresse wird mit allen vorhandenen lokalen Netzen verglichen. Wird ein Netz mit identischer Netzadresse gefunden, wird das Paket am entsprechenden Interface rausgesendet.
- Routing-Tabelle nach passender Hostadresse durchsuchen
  Passt keine der lokalen Netzwerkadressen mit der Zieladresse überein, so wird die Routing-Tabelle nach einer übereinstimmenden Hostadresse durchsucht.
- Routing-Tabelle nach passender Netzwerkadresse durchsuchen
  Existiert kein identischer Hosteintrag in der Routing-Tabelle, so wird nach einer übereinstimmenden Netzwerkadresse durchsucht. Wird eine passende Netzwerkadresse gefunden, so ersetzt der Kernel die Ethernet-Zieladresse des Paketes duch die Ethernet-Adresse des gefundenen Routers. Die IP-Adresse des Ziels bleibt unverändert.
- Suche nach einem Default Routing Eintrages in der Routing-Tabelle
  Waren alle bisherigen Suchen erfolglos, wird in der Routing-Tabelle des Kernels nach einem Default-Router gesucht. Existiert ein Default-Router, so wird die Ethernet-Zieladresse des Datenpaketes durch jene des Default-Routers ersetzt und das Paket verschickt.
- Existiert keine Route zum Ziel, wird eine ICMP Fehlernachricht generiert
  Wurde keine passende Route gefunden und es ist auch kein Default-Router vorhanden, so kann das Datenpaket nicht weitergeleitet werden. Der Kernel generiert ein ICMP Pakete, welches den Absender-Host über die Fehlersituation benachrichtigt. Der Absender erhält die Meldung '*No route to host or network is unreachable*'.

Der oben beschriebene Ablauf ist in der Abbildung als Flussdiagramm ersichtlich.

Abbildung 16.4: Flussdiagramm des Routing-Prozesses



16.3.3 Autonomous System (AS)

Eine Ansammlung von Netzwerken und Routern, die unter der Kontrolle einer einzigen Administration stehen, wird ein *Autonomous System* (*AS*) genannt. Einem AS wird von der INTERNIC eine eindeutige 16-bit Adresse zugewiesen. Aufgrund der unterschiedlichen Anforderungen werden innerhalb einer AS andere *Routing-Protokolle* (oder auch *Gateway Protokolle*) verwendet als zwischen den AS'en. Router, die sich zwischen den Autonomen Systemen befinden, kommunizieren untereinander mittels *Exterior Gateway Protocol* (*EGP*), um ihre Routingtabellen aufzubauen. Router innerhalb einer AS verwenden *Interior Gateway Protocols* (*IGP*).

16.3.4 Gateway Protokolle

Wie schon erwähnt, werden die Gateway Protokolle in 2 Klassen unterteilt – Exterior Gateway Protocol (EGP) und Interior Gateway Protocol (IGP).

16.3.4.1 Exterior Gateway Protocol (EGPs)

EGP stammt aus den frühen 80-iger Jahren. EGP organisiert den Informationsaustausch über 3 Protokolle:

Neighbour acquisition EGP beinhaltet einen Mechanismus, der es benachbarten autonomen Systemen ermöglicht, den Austausch von Routing-Informationen auszuverhandeln.

Neighbour reachability Sobald ein EGP Gateway[13] die 'Nachbarschaft bestätigt', senden sich die beiden Nachbarn zyklisch *keep alive* Informationen, um die Verfügbarkeit des jeweils Anderen zu überprüfen.

Network reachability Die Liste der von einem Autonomen System erreichbaren Netzwerke wird periodisch an den Nachbarn vermittelt. EGP limitiert diese Routen mittels eines *Distance Vector* (255).

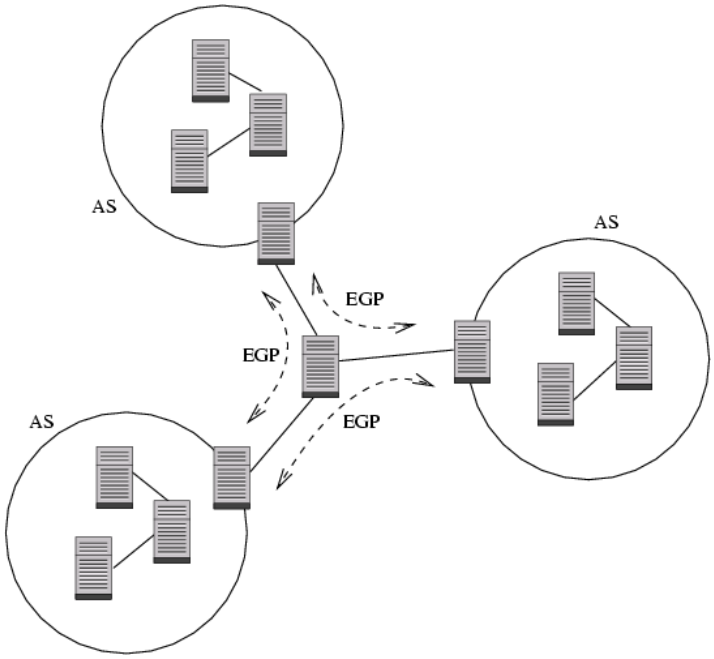
16.3.4.2 Border Gateway Protocol (BGPs)

BGP wurde entwickelt, um einige Mängel desEGP zu beseitigen. Dies erfolgt durch zusätzliche Attribut Flags (welche u. a. auch EGP Kommunikation interpretiert) und durch die Ersetzung des Distanz-Vektors durch einen *Path Vector*.

Der *Path Vector*, der in BGP implementiert wurde, führt dazu, dass der komplette Pfad von der Quelle zum Ziel in den Routingtabellen hinterlegt wird. Das verhindert die Probleme von Schleifen (*looping problem*), wie sie in einem komplexen Netzwerk wie dem Internet möglich sind. Einzige Möglichkeit einer Schleife wäre, wenn in einem Pfad eine AS doppelt vorkäme. In diesem Falle generiert BGP eine Fehlersituation.

[13] Der Begriff *Gateway* wird in diesem Zusammenhang für einen Router verwendet, der an der Grenze eines Autonomen Systems steht und eine Verbindung 'nach außen' hat.

Abbildung 16.5: Exterior Gateway Protocol (EGP)



BGP reduziert auch die Zeit, um festzustellen, dass ein bestimmtes Netzwerk nicht erreichbar ist. Nachteil von BGP ist, dass der Path Vector durch die größere Informationsmenge mehr Speicher erfordert.

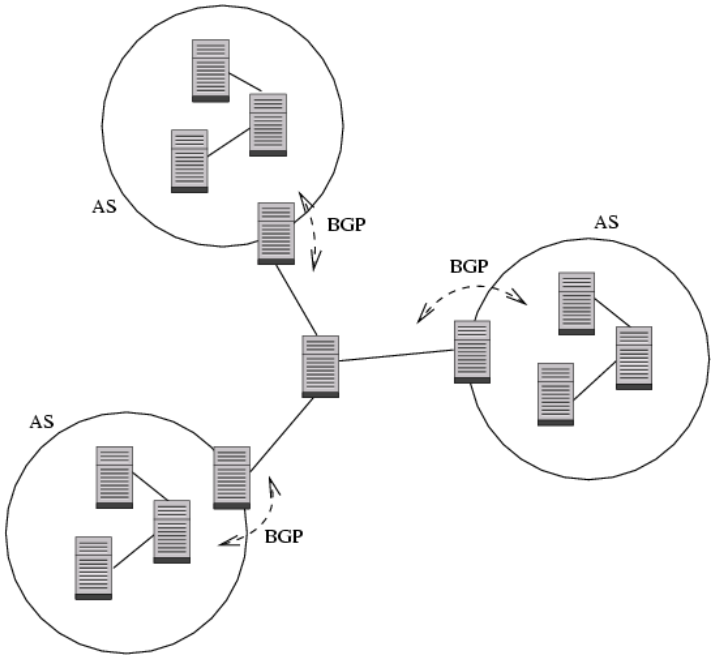
BGP ist EGP ähnlich. Es verwendet eine *keep-alive* Prozedur und handelt mit andern BGP-Routern einen Informationsaustausch aus. Anstatt einer regelmäßigen Benachrichtigung, erfolgt der Informtionsaustausch immer bei einer Änderung von Routen.

BGP ist im Internet weiter verbreitet als EGP. Zusätzlich hat BGPv4 Erweiterungen für die Unterstützung von *classless interdomain routing* (*CIDR*).

Classless Inter-Domain Routing (CIDR) *CIDR* wurde 1992 entwickelt, als Antwort auf die zu Ende gehenden Klasse-B Adressen und die überquellenden Routing-Tabellen. Es sollte eine Übergangslösung sein, bis IPv6 flächendeckend eingesetzt ist. CIDR begegnet diesen Problemen auf folgende Art:

- Effizientere Nutzung des IP-Adressraumes
  Mit *classfull* Routing Protokollen werden nur Netzwerke für 254, 65534 oder 16777214 Hosts unterstützt. Classfull Protokolle werten die ersten 3 Bits der IP-Adresse aus, um die Klasse (A, B oder C) zu bestimmen.
  CIDR kompatible Protokolle verwenden nicht die ersten 3 Bits, stattdessen wird die Netzmasken mit jeder Routing-Information mitgeschickt. Dadurch ist es möglich, Netzwerke exakt nach Bedarf zu konfigurieren.
- Gruppierung von Routing Tabellen, um die Größe der Tabellen auf den Backbone-Routern zu reduzieren. Die Basis für eine Gruppierung ist, dass ISPs (Internet Service Provider) ein großer

Abbildung 16.6: Border Gateway Protocol (BGP)



zusammenhängender Block von IP-Adressen zugewiesen wird. Alle Netzwerk-Adressen des ISP teilen sich dieselben ersten 3 Bits. Dadurch kann durch einen einzelnen Eintrag eines Backbone-Routers eine Menge darunterliegender Netzwerke repräsentiert werden.

16.3.4.3 Interior Gateway Protocols (IGPs)

Für die Verteilung von Routing-Informationen innerhalb eines Autonomen Systems existieren mehrere Protokolle. Im Folgenden werden die beiden Protokolle *RIP* und *RDISC* genauer behandelt.

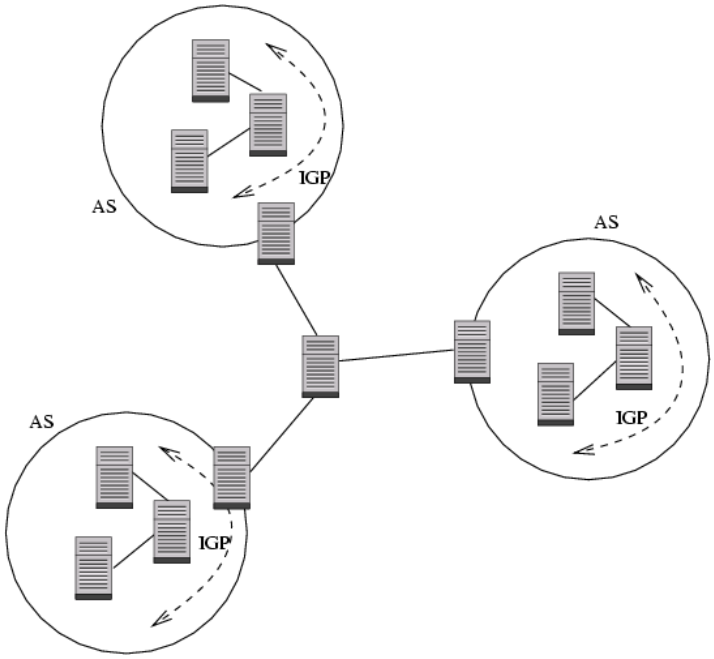
Open Shortest Path First (OSPF) Das *Open Shortest Path First* (*OSPF*) Protokoll ist ein sogenanntes *link-state* Protokoll. Die Routen werden nicht, wie bei RIP anhand von Distanz Vektoren ermittelt, sondern OSPF verwaltet einen Plan der Netzwerk-Topologie. Das unterstützt eine globalere Sicht des Netzwerkes und demzufolge ermöglicht es die Auswahl des *shortest path* (kürzesten Weges) zum Ziel. Diese Pläne werden regelmäßig aktualisiert.

Hauptvorteile eines *link state* Protokolls sind:

Schnelle Konvergenz ohne Schleifen Vollständige Ermittlung von Wegen passiert lokal und macht es somit schneller. Dadurch, dass die vollständige Netzwerkabbildung lokal gehalten wird, ist die Bildung von Schleifen unterbunden.

Unterstützung von *multiple metrics* OSPF erlaubt die Bewertung von mehreren Eigenschaften wie kürzeste Verzögerung, größter Durchsatz und höchste Zuverlässigkeit. Das ermöglicht eine flexiblere Wegewahl.

Abbildung 16.7: Interior Gateway Protocol (IGP)



Multiple paths In komplexeren Netzen, wo es mehrere Wege zum selben Ziel gibt, ist OSPF in der Lage Lastverteilung (*load-balancing*) zu ermöglichen.

Intra-Domain Intermediate System to Intermediate System (IS-IS) *Das Intra-Domain Intermediate System to Intermediate System* (*IS-IS*) Protokoll ist ein dem OSPF sehr ähnliches *link-state* Protokoll. Es wurde speziell für OSI-konforme Netze entwickelt.

Routing Information Protocol (RIP) Das *Routing Information Protocol* (*RIP*) ist ein *distance-vector* Protokoll. Der Name Distanz-Vektor rührt daher, dass der Abstand zu den erreichbaren Netzwerken beschrieben wird.

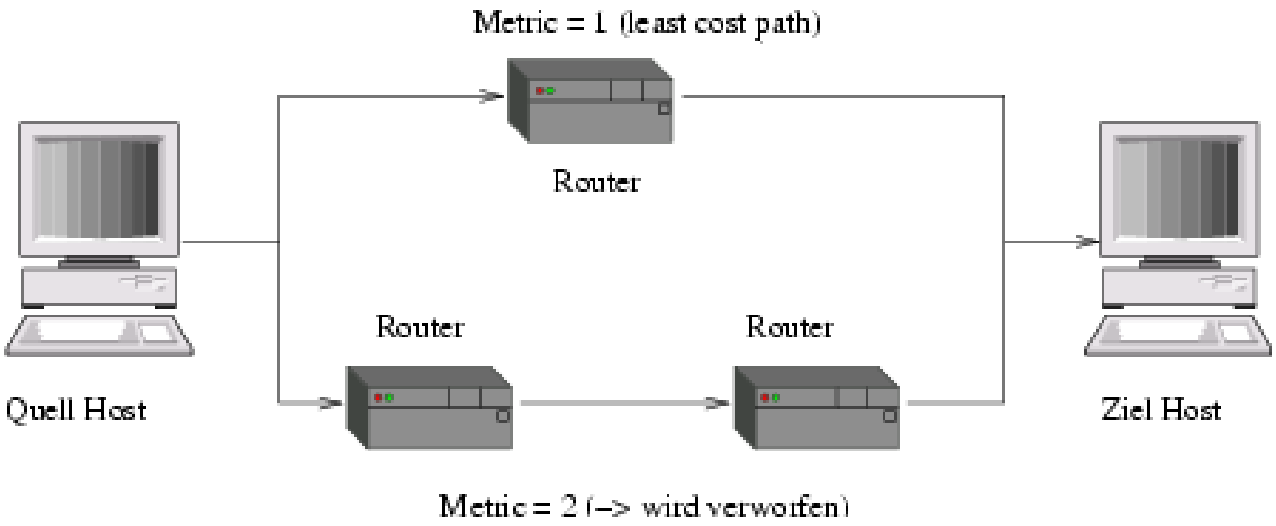
Vorteile von RIP:

- weite Verbreitung, stabil und einfach zu implementieren;
- Routing Tabellen werden alle 30 Sekunden durchgeführt
- es macht manuelle Administration der Routing-Tabellen überflüssig; die Aktualisierung erfolgt automatisch;

Nachteile von RIP:

- aufgrund der regelmäßigen Broadcasts kann eine erhöhte Netzlast erzeugt werden

Abbildung 16.8: Least Cost Path



- Multiple Metriken werden nicht unterstützt (Bewertung von mehreren Eigenschaften)
- Lastverteilung (load-balancing) wird nicht unterstützt
- wird eine Anzahl von mehr als 15 Hops[14] erreicht, ist gilt das Ziel als 'unendlich weit entfernt' und ist unerreichbar

Least Cost Path Die Effizienz einer Route wird bestimmt durch die Distanz zwischen Quelle und Ziel. Dazu wird die Anzahl der Router dazwischen gezählt. Der ermittelte Wert wird *hop count* genannt. Existieren mehrere Wege zum Ziel, entscheidet sich RIP für jenen Weg mit den 'geringeren Kosten' (*least cost*) – also den Weg mit dem kleineren *hop count*–Wert.

Stabilitäts–Merkmale RIP besitzt einige Eigenschaften, welche auch in einem sich rasch ändernden Netzwerk Stabilität garantieren. Dazu gehören *hop-count limit*, *hold-downs*, *split horizons* und *poison reverse updates*.

Hop-count limit RIP erlaubt einen maximalen *hop count* von 15. Jedes Ziel, welches eine größere Distanz aufweist, gilt als unerreichbar. Dieses Maximum schränkt die Einsetzbarkeit von RIP ein, verhindert aber andererseits die Bildung von Routing-Schleifen.

Hold-down state *Hold-downs* werden eingesetzt, um zu verhindern, das Routing-Nachrichten einer ungültigen Wiederverfügbarkeit einer Route verbreitet werden. Geht ein Router außer Betrieb, so bemerken das dessen Nachbarrouter. Diese ermitteln eine andere Route und verbreiten die neue Routing-Information. Es entsteht so eine 'Welle' von Routing-Updates im Netzwerk.
Da die Information nicht von allen Routern im Netzwerk sofort erhalten wird, kann es passieren,

[14] Ein *hop* ist gleich der Anzahl Router zwischen Quelle und Ziel.

dass ein Gerät eine veraltete Information verbreitet, während der tatsächliche Zustand aber ein anderer ist.
Hold downs veranlassen einen Router dazu, Statuswechsel, die eine soeben entfernte Route betreffen, gewisse Zeit zurückzuhalten. Diese *Hold-down Periode* muss gerade etwas größer sein, als die Zeit, die benötigt wird, um alle Router in einem Netz über einen Statuswechsel zu informieren. *Hold down* verhindert das sogenannte *count-to-infinity Problem*.

Split horizons Es ist nicht sinnvoll, Routing-Informationen in dieselbe Richtung zurückzusenden, von der sie kam. Die *split-horizon* Regel verhindert dies.

Triggered updates with poison reverse Während split horizons Endlosschleifen zwischen benachbarten Routern verhindert, sollen *poison reverse updates* längere Routing-Schleifen vereiteln. Die Idee dahinter ist die, dass wachsende Routing-Metriken auf eine Schleife hindeuten. In diesem Fall werden sogenannte *poison reverse updates* verschickt, um diese Routen zu entfernen und in den *hold-down Status* zu versetzen.

in.routed Prozess Der Prozess *in.routed* implementiert RIP. Er erstellt und verwaltet die dynamischen Routing-Informationen. Sobald mehr als nur ein Netzwerkinterface existiert, verschickt per Broadcast alle 30 Sekunden eigene Routing-Informationen. Alle Rechner empfangen diese Broadcasts. Auf allen Rechnern, auf denen auch *in.routed* läuft, werden diese Informationen verarbeitet. Auf Routern wird *in.routed* mit der Option '*-s*' gestartet, was das Versenden der Broadcasts bewirkt. Auf Rechnern ohne Routing-Funktion wird *in.routed* mit der Option '*-q*' gestartet. Diese Rechner werten die empfangenen Routing-Informationen lediglich für ihre eigenen Routing-Tabellen aus und verhalten sich ansonsten still.

Der *in.routed* Dämon wird beim Systemstart durch das Run-Script */etc/init.d/inetinit* gestartet.

Syntax:

/usr/sbin/in.routed [-gqstv] [logfile]

-q *quiet modus* – keine Broadcasts werden gesendet; Prozess hört nur auf Routing-Information

-s *server modus* – Routing-Informationen werden per Broadcast verteilt

-v *logfile* *verbous modus* – Meldungen werden in *logfile* geschrieben

-g -t Logging wird auf den Bildschirm ausgegeben

Network Router Discovery *RDISC* ist ebenfalls ein Prozess, der in der Lage ist Routing-Informationen zu senden, zu empfangen und entsprechend zu verarbeiten. Sind mehrere alternative Routen möglich, können mit Hilfe von RDISC dynamisch Default-Gateways konfiguriert werden. Bei Ausfall des Gateways wird automatisch auf einen anderen Router gewechselt. Windows-Clients sind leider nicht in der Lage dieses Feature zu nützen.[15]

[15] Falls doch, bin ich um Hinweise dankbar.

in.rdisc Prozess Router, auf denen *in.rdisc -r* läuft, senden alle 10 Minuten Routing-Informationen an die Multicast-Adresse 224.0.0.1. *in.rdisc -s* hört auf Rechnern ohne Routing-Funktion bei der Adresse 224.0.0.1 auf eingehende Routing-Nachrichten.

Vorteile von RDISC

- Unabhängig von Routing-Protokollen
- Verwendung von Multicast-Adressen
- erzeugt kleinere Routing-Tabellen
- unterstützt Redundanz durch mehrfache Default-Routing Einträge

Nachteile von RDISC

- Eine Periode von 10 Minuten kann sogenannte *black holes* verursachen. Ein *black hole* ist die Zeit, die eine Route in der Tabelle verbleibt, obwohl die Strecke bereits ausgefallen ist. Die Standard-Lebenszeit einer Route beträgt 30 Minuten.
- Auf Routern *muss* zusätzlich ein Routing-Protokoll laufen (z. Bsp. RIP), um andere Netzwerke kennenzulernen. RDISC (in.rdisc) unterstützt lediglich Default-Routen zu Hosts, nicht zwischen Routern.
- ICMP Redirects können generiert werden, sobald mehr als ein Default-Router auf einem Host eingetragen ist.

Syntax

/usr/sbin/in.rdisc [-a] [-s] [send-address] [receive-address]

/usr/sbin/in.rdisc -r [-p preference] [-T interval] [send-address] [receive-address]

-s sende bei Start drei 'solicitation messages' um rasch alle Router zu entdecken

-r arbeite als Router und versende Routing-Informationen

-T *intervall* setzen des Zeit-Intervalls in dem die Informationen verteilt werden (Standard ist 600 Sekunden)

16.3.5 Multihomed Host

Erkennt Solaris beim Startup mehr als nur eine Netzwerkkarte, so erhält der Rechner automatisch Routing-Funktionen. Es ist jedoch möglich, einen Rechner zu einem *Multihomed Host* zu machen – einen Rechner mit mehrern LAN-Schnittstellen, aber ohne Forwarding-Funktion und ohne Routing-Protokoll.

Folgende Maschinen-Typen können als *Multihomed Host* konfiguriert werden:

NFS Server Um den Zugriff auf große Data-Centers aufzuteilen, haben solche Server oft mehrere Netzwerkkarten.

Database Server Wie auch bei NFS-Servern verbessern mehrere Netzwerk-Interfaces den Zugriff auf Datenbank-Server.

Firewall Gateway Firewalls sind Geräte, welche an der Grenze zwischen privaten und öffentlichen Netzen wie z. Bsp. dem Internet sitzen und unerlaubte Zugriffe verhindern.

Wenn ein Rechner startet, sieht das Run-Script */etc/init.d/inetinit* nach der Datei */etc/notrouter*. Existiert diese Datei, werden die beiden Routing-Prozesse *in.routed -s* sowie *in.rdisc -r* nicht gestartet und *IP forwarding* wird nicht aktiviert. Der Prozess *in.rdisc -s* wird in jedem Fall gestartet. Der detaillierte Ablauf ist im Flussdiagramm ersichtlich.

Konfiguration eines Multihomed Hosts:

1. Datei /etc/hostname.*interface* erstellen und mit Editor einen Hostnamen für dieses Interface eintragen
2. Zusätzlichen Hostnamen und zugehörige IP-Adresse in /etc/inet/hosts eintragen.
3. Leere Datei mit dem Namen /etc/notrouter erzeugen (z.Bsp. mit '`touch /etc/notrouter`')
4. Leere Datei /etc/reconfigure erzeugen, damit bei nächstem Reboot eine Hardware-Erkennung durchgeführt wird.
5. Shutdown mit '`init 5`'
6. Einbau der zusätzlichen Netzwerkkarten
7. Boot des Systemes
8. Kontrolle mit '`ifconfig -a`', ob alle Netzwerkkarten vorhanden und deren Adressen richtig sind.
9. Kontrolle, ob der Rechner wirklich ein 'Multihomed Host' ist:
   `# ps -ef | grep in.r`
   Es darf weder der Prozess *in.routed* noch *in.rdisc -r* laufen (*in.rdisc -s* ist OK)

16.3.6 Das Kommando netstat -r

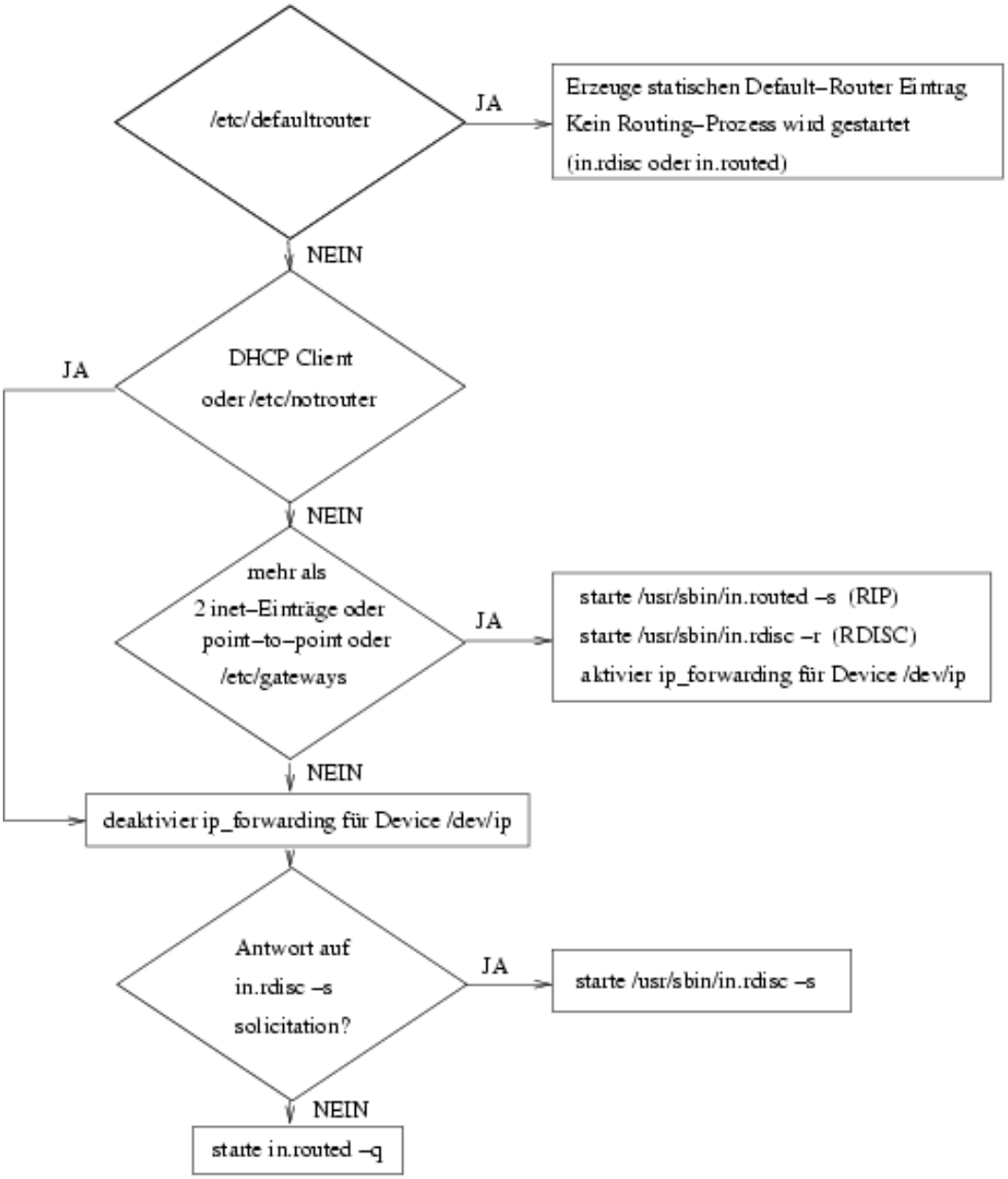
netstat -r listet die Routing-Tabelle des Kernels auf. Siehe auch 16.7.9 auf Seite 222!

Syntax

Beispiel:

```
Routing Table:
 Destination     Gateway       Flags  Ref  Use  Interface
------------- ----------------- ----- ----- ------ ---------
193.16.214.0  193.16.214.28      U        4    124  hme0
224.0.0.0     193.16.214.28      U        4      0  hme0
default       193.16.214.250     UG       0   1286
127.0.0.1     127.0.0.1          UH       0      6  lo0
```

Abbildung 16.9: Router Initialisierung – */etc/init.d/inetinit*



Bedeutung der Spalten:

Destination	Zielrechner bzw. Zielnetz
Gateway	Der Rechner, der die Datenpakete weiterleitet
Flags	Status dieser Route U – Interface ist 'up' H – Zieladresse ist ein Host G – Gatewayadresse ist ein anderer Host (indirekte Route) D – Route wurde durch ein ICMP Redirect verursacht
Ref	Anzahl der Routen, die sich dieselbe Interfaceadressse teilen (Ethernet)
Use	Anzahl der Pakete, die über diese Route geschickt wurden. Beim *localhost* ist es die Anzahl der empfangenen Pakete
Interface	Interface, das zum Ziel führt

16.3.7 /etc/inet/networks Datei

Um einer Netzwerkadresse einen Namen zuzuordnen, kann die Datei */etc/inet/networks* editiert werden. Die Datei hat folgenden Aufbau:

```
NETZWERK-NAME  NETZWERK-ADRESSE  ALIAS-NAMEN
```

Beispiel:

```
vie      193.16.214    Wien
mch      224.0.0       Muenchen
loopback 127.0
```

Mit obiger *networks*-Datei würde das Kommando '*netstat -r*' folgende Ausgabe liefern:

```
Routing Table:
 Destination     Gateway       Flags  Ref  Use  Interface
-------------- ---------------- ----- ----- ------ -------
vie            193.16.214.28     U         4    124  hme0
mch            193.16.214.28     U         4      0  hme0
default        193.16.214.250    UG        0   1286
loopback       127.0.0.1         UH        0      6  lo0
```

/etc/inet/networks wird auch vom Kommando *route* ausgewertet.

16.3.8 Das Kommando route

Mit Hilfe von *route* kann die Routing-Tabelle des Kernels manipuliert werden.

ACHTUNG! Werden Routing-Einträge manuell aus der Tabelle gelöscht, müssen ev. vorhanden Routingprozesse wie *in.routed* und *in.rdisc* gestoppt und wieder gestartet werden, da sonst keine automatische Verteilung und Aktualisierung von Routen mehr erfolgt.

Syntax

route [-fn] add|delete [host|net] ziel-adresse [gateway [metric]]

-f *flush* – löscht alle Routing-Einträge aus Routing-Tabelle

-n keine Namensauflösung[16]

add host *host-ipadr gateway-ipadr* fügt Route für einen Host zur Routingtabelle hinzu

add net *net-ipadr gateway-ipadr* fügt Route für ein Netz zur Routingtabelle hinzu

delete host *host-ipadr gateway-ipadr* entfernt Route für einen Host

delete net *net-ipadr gateway-ipadr* entfernt Route für ein Netz

metric Der optionale Parameter *metric* gibt die Anzahl der vorhandenen Router zwischen Quelle und Ziel an.[17]

Beispiele:

```
Hinzufügen einer Route für das Netz 128.50.3.0 -- erreichbar
über den Router mit der Adresse 128.50.3.254
# route add net 193.16.214.0 193.16.214.254 1

Hinzufügen einer Route für ein Netz. Der Netzwerkname ist
in /etc/inet/networks und der Hostname des Routers in
/etc/inet/hosts eingetragen
# route add net vie gw 1

Löschen der Route aus der Tabelle
# route delete net 193.16.214.0 gw

Löschen der gesamten Routing-Tabelle
# route -f

Hinzufügen einer Mulitcast-Route ('myhost' ist eigener
Rechnername)
# route add 224.0.0.0 myhost 0
```

[16] Falls die Namensauflösung nicht über die lokalen Konfigurationsdateien, sondern über einen anderen Nameserver erfolgt, dieser aber nicht erreichbar ist, sollte der Schalter '*-n*' verwendet werden.

[17] RIP wählt immer jenen Weg mit dem kleinsten Wert, falls mehrere Wege zum Ziel führen. Häufig wird für wichtige Strecken neben der Standleitung ein 2. Weg (häufig eine ISDN-Strecke) eingerichtet. Damit bevorzugt die Standleitung verwendet wird, wendet man in Routern einen Trick an: Es wird die Metrik für die ISDN-Strecke auf einen höheren Wert als für die Metrik der Standleitung gesetzt. Dadurch wird die Standleitung bevorzugt und erst dann auf die ISDN-Strecke ausgewichen, wenn die Standleitung ausfällt.

16.3.9 Datei /etc/gateways

Der Prozess *in.routed* liest während der Initialisierung die optional vorhandene Datei */etc/gateway* um seine Routing-Tabelle zu erzeugen. Durch Editieren dieser Datei können permanente Routen eingetragen werden. Der Aufbau dieser Datei ist folgender:

```
net ziel-netz gateway router metric cnt [passive | active]
```

Beispieleintrag:

```
net 128.50.0.0 gateway 128.50.0.254 metric 1 passive
```

Der Parameter *passive* bewirkt, dass dieser Routing-Eintrag durch ein Routing-Protokoll überschrieben werden kann.

16.3.10 Router Konfiguration

Um einen Solaris-Rechner zu einem Router zu machen, sind unten aufgelistete Schritte nötig. Es wird davon ausgegangen, dass zusätzliche Netzwerkkarten bereits eingebaut wurden. Entweder muss vor dem Einbau der Karte die Datei */reconfigure* erzeugt werden (mit Kommando *touch /reconfigure*) oder es muss im OBP mit *boot -r* gestartet werden, damit die neue Hardware erkannt wird.

1. Erzeugen einer Datei /etc/hostname.*interface* für jede zusätzlich eingebaute Netzwerkkarte. Als Inhalt der Datei muss der dem Interface zugeordnete Hostname eingetragen werden.
2. Obigen Hostname und zugehörige IP-Adresse in die Datei */etc/inet/hosts* eintragen.
3. Reboot der Maschine und anschließende Kontrolle mit *ifconfig -a*.[18]

16.3.10.1 Fehlersuche bei der Router-Konfiguration

Bei Fehlern sind mehrere Ursachen möglich:

- Wurden alle Netzwerkkarten korrekt erkannt?
  # prtconf
  Wird das neue Gerät gelistet? Gerätenamen und Instance-Number notieren.
- **# ifconfig -a**
  Werden alle Karten aufgelistet? Sind deren Adressen korrekt und ist der Zustand 'UP'?
- **# ls -l /etc/hostname.*interface***
  Existiert die Datei? Anstatt *interface* muss der Name des Gerätes und dessen Instanz-Nummer angegeben werden.

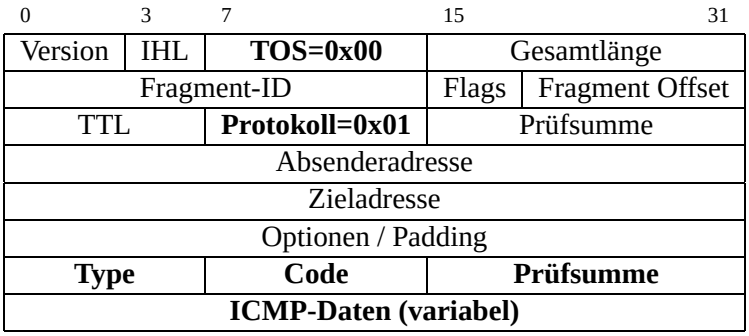
[18] Selbstverständlich können die Dateien auch vor dem Einbau der Netzwerkkarte(n) editiert werden um ein zweimaliges Durchstarten des Rechners zu umgehen.

- # **cat /etc/hostname.*interface***
  Ist der eingetragene Hostname richtig?
- **# cat /etc/inet/hosts**
  Ist der in der Datei /etc/hostname.interface eingetragene Name auch in /etc/inet/hosts eingetragen und ist dessen zugewiesene IP-Adresse richtig?

16.4 ICMP-Nachrichten

Im Kapitel 16.3 wurde die Bedeutung der ICMP-Nachrichten im Zusammenhang mit Routern bereits erwähnt. Hier folgt nun eine detailliertere Beschreibung dieses Protokolls.

16.4.1 Das ICMP-Paket



Charakteristisch für ein ICMP-Paket ist im IP-Header zum einen, dass das *Type of Service* Feld auf 0x00 und der *Protokolltyp* auf 0x01 gesetzt ist. Im Nutzdatenbereich des IP-Paketes befinden sich die eigentlichen ICMP-Informationen.

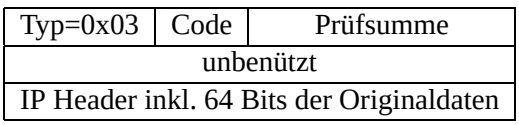
Type Dieses Feld ist 1 Byte groß und gibt an um welche ICMP-Nachricht es sich handelt. Der Aufbau der Daten im ICMP-Datenfeld hängt von diesem Typ ab.

Code Der Informationsgehalt des Code-Feldes ist auch vom ICMP-Typ abhängig. Dort werden zusätzliche Informationen hinterlegt (siehe weiter unten).

Prüfsumme Die Prüfsumme wird über alle Felder einschließlich ab dem ICMP-Typ, also den gesamten Nutzdatenbereich des IP-Paketes.

Im RFC 792 werden 11 verschiedene ICMP-Typen beschrieben. Im folgenden Abschnitt werden diese genauer erläutert.

16.4.2 Destination Unreachable



Die Nachricht *Destination unreachable* wird von einem Router in folgenden Fällen an den Absender eines Datenpaketes zurückgeschickt. In allen Fällen enthält die ICMP-Nachricht den Header und die ersten 64 Bits des ursprünglichen IP-Paketes, welches nicht zugestellt werden konnte.

Net unreachable Code=0x00 – Das Netzwerk, des Zielhosts ist nicht erreichbar. Zum Beispiel dann, wenn die Distanz der Zieladresse in der Routing-Tabelle den Wert 'unerreichbar' enthält.

Host unreachable Code=0x01 – Der gewünschte Zielrechner kann im Zielnetz nicht erreicht werden.

Protocol unreachable Code=0x02 – Diese Nachricht kann von einem Router oder auch einem Endsystem generiert werden. Und zwar dann, wenn am Zielport, welches im TCP-Header angegeben ist, nicht das erwartete Protokoll läuft.

Port unreachable Code=0x03 – Auch diese Meldung kann sowohl von einem Router als auch von einem Endsystem kommen. Sie wird an den Absender geschickt, wenn hinter dem angegebenen Port keine Applikation läuft.

Fragmentation needed Code=0x04 – Weiter oben haben wir gehört, dass das Flag *Don't fragment* verhindert, dass ein Datenpaket fragmentiert wird. Stellt aber ein Router fest, dass eine Fragmentierung zwingend nötig ist, um das Paket weiterleiten zu können, wird das Paket verworfen und die ICMP-Nachricht *Fragmentation needed* an den Sender geschickt.

Source Route failed Code=0x05 – Ist im IP-Paket die Option *Source Routing* gesetzt und es tritt dabei ein Fehler auf, erhält der Absender diese Nachricht.

16.4.3 Source Quench

Typ=0x04	Code=0x00	Prüfsumme
unbenützt		
IP Header inkl. 64 Bits der Originaldaten		

Bei hoher Netzlast, kann es vorkommen, dass ein Router nicht mehr in der Lage ist alle ankommenden Pakete korrekt weiterzuleiten und diese dann verwerfen muss. Der Router kann dem Absender dieses Ereignis mittels *Source Quench* Nachricht mitteilen (RFC 792). Der Sender sollte dann darauf mit einer geringeren Datenrate reagieren. Es ist auch möglich schon vor dem Eintritt dieser 'Notsituation' Source Quench Pakete zu verschicken um dem vorzubeugen.

Da in einer Überlastsituation dadurch aber zusätzliche Netzlast erzeugt wird und der Missbrauch durch künstlich erzeugte Source Quench Nachrichten ein Netzwerk lahmlegen kann, geht man heute eher davon ab, diese Funktionalität zu implementieren (sieh auch RFC 1812).

16.4.4 Redirect

Typ=0x05	Code	Prüfsumme
IP Adresse des Routers		
IP Header inkl. 64 Bits der Originaldaten		

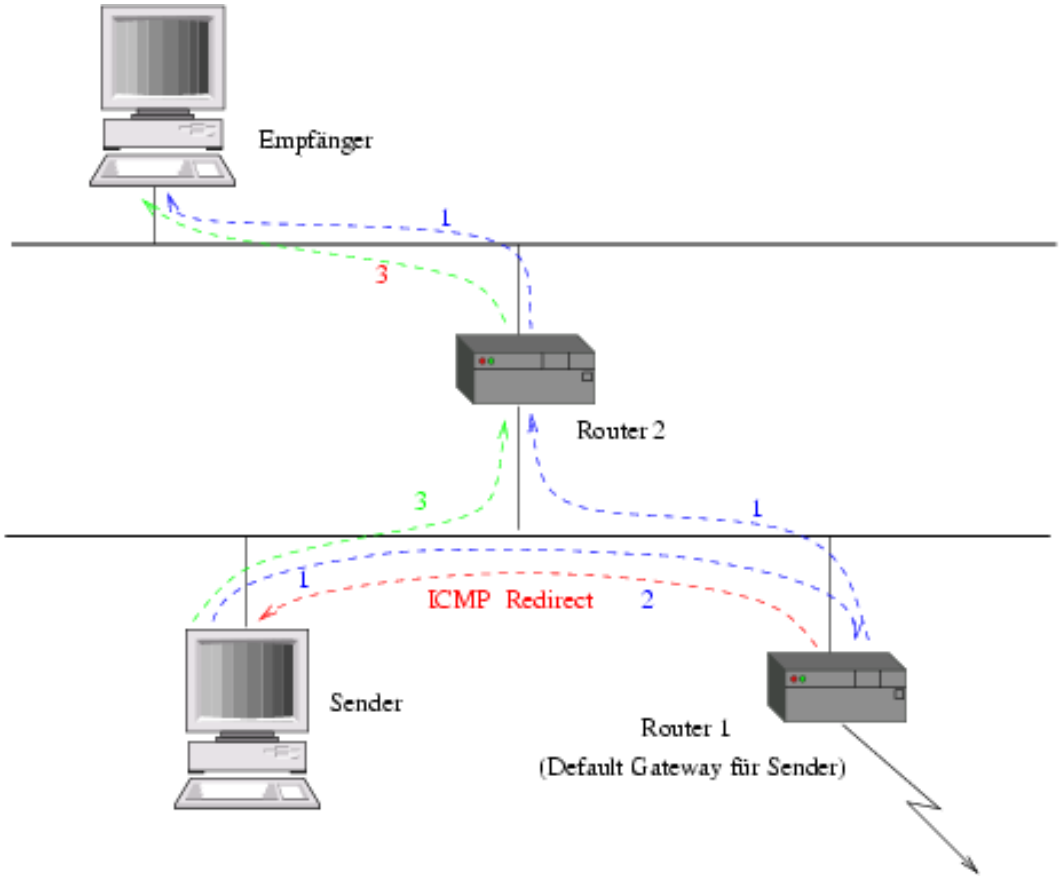
Code = 0x00 Umleitung aller IP-Pakete, die gleiche Ziel-Netzadresse tragen wie das soeben gesendete.

Code = 0x01 Umleitung der IP-Pakete, die die gleiche Empfangs-IP-Adresse wie das aktuelle Paket tragen.

Code = 0x02 Umleitung nur der IP-Pakete, die dieselbe Ziel-Netzadresse und denselben Type of Service (TOS) wie das aktuelle Paket enthalten.

Code = 0x03 Umleitung der IP-Pakete, die gleiche Empfangs-IP-Adresse und denselben Type of Service (TOS) wie das aktuelle Pakete enthalten.

Abbildung 16.10: ICMP Redirect



Die *Redirect* Nachricht soll die Routingwahl durch Netzwerke optimieren. Angenommen wir hätten ein Netzwerk mit mehreren Routern um in andere Netze gelangen zu können (siehe Abbildung). Der Sender hat in seiner Routingtabelle keine Route zum gewünschten Zielrechner, aber er findet einen Default-Gateway Eintrag. Dieser Default-Gateway sei 'Router 1'. Der Sender wird das Paket deshalb an 'Router 1' schicken, damit es an den Zielhost weitergeleitet wird. 'Router 1' kennt den Weg zum Zielhost, nämlich über 'Router 2' und schickt das Paket dorthin. Er bemerkt aber auch, dass es effizienter wäre, wenn der Absender sich gleich an 'Router 2' gewandt hätte und teilt dies dem Sender mittels ICMP-Redirect Nachricht mit. Der Sender erhält diese Nachricht und kann aufgrund dieser Information seine Routingtabelle um diese Route erweitern, sodass in Zukunft die Pakete nicht 2 mal (einmal vom Sender zu 'Router1', dann von 'Router 1' zu 'Router 2') im lokalen Netz übertragen werden.

Eine Redirect-Nachricht wird aber keinesfalls generiert, falls die IP-Option 'Source Route' gesetzt ist.

16.4.5 Echo Request und Echo Reply

Echo Reply - Nachricht:

Typ=0x00	Code=0x00	Prüfsumme
Identifier		Sequenz Nummer
Daten		

Echo Request - Nachricht:

Typ=0x08	Code=0x00	Prüfsumme
Identifier		Sequenz Nummer
Daten		

Sendet ein Rechner eine *Echo Request* Nachricht an einen Zielrechner, so antwortet dieser mit einer *Echo Reply* Nachricht. Angewendet wird das u. a. beim Kommando 'ping', um festzustellen, ob ein Zielrechner erreichbar ist. Die beiden Nachrichten unterscheiden sich im Typ-Feld. Das Code-Feld trägt den Wert '0', für die Felder *Identifier*, *Sequence Number* und *Daten* sind in RFC 792 keine bestimmten Werte vorgesehen. Allerdings muss der Zielrechner den Inhalt dieser Felder in die Reply-Nachricht kopieren. Häufig verwendet das Kommando 'ping' das Feld 'Identifier' als fortlaufenden Zähler. Das Kommando 'ping' ist in Kapitel 16.7.7 auf Seite 221 beschrieben.

16.4.6 Time Exceeded

Typ=0x0B	Code	Prüfsumme
unbenützt		
IP Header inkl. 64 Bits der Originaldaten		

Code = 0x00 Diese Nachricht wird von einem Router gesendet, falls die *Time to Live* (*TTL*) eines Paketes den Wert '0' erreicht und das Paket verworfen wird.

Code = 0x01 Kann ein Endsystem nicht innerhalb einer vordefinierten Zeit alle Fragmente einer Nachricht zusammensetzen, weil Fragmente fehlen, werden alle Fragmente verworfen und diese Nachricht an den Sender geschickt.

16.4.7 Parameter Problem

Typ=0x0C	Code=0x00	Prüfsumme
Zeiger	unbenützt	
IP Header inkl. 64 Bits der Originaldaten		

Empfängt ein Router oder Endsystem ein IP-Paket, dessen Header nicht interpretiert werden können, so wird das Paket verworfen und der Sender kann mit der ICMP-Nachricht *Parameter Problem* informiert werden. Im Zeiger-Feld steht die Position im IP-Header an der ein Problem auftrat. Beträgt der Wert des Zeigers z. Bsp. 0x01, so bezieht sich das auf die Version (normalerweise 0x04 für IPv4).

16.4.8 Timestamp Request und Timestamp Reply

Timestamp Request:

Typ=0x0D	Code=0x00	Prüfsumme
Identifier		Sequenz Nummer
Originate Timestamp		
Receive Timestamp		
Transmit Timestamp		

Timestamp Reply:

Typ=0x0E	Code=0x00	Prüfsumme
Identifier		Sequenz Nummer
Originate Timestamp		
Receive Timestamp		
Transmit Timestamp		

Diese ICMP-Nachricht dient dazu, die Systemzeit eines Rechners oder Routers zu erfahren. Der Sender schickt ein *Timestamp Request* Paket an ein Ziel. Im Feld *Originate Timestamp* wird die aktuelle Zeit des Senders eingetragen. Empfängt der Zielhost diese Nachricht, wird eine *Timestamp Reply* Nachricht an den Absender zurückgeschickt. Im Feld *Receive Timestamp* ist dann die Zeit zu finden, zu der das Request-Paket vom Zielrechner empfangen wurde. Im Feld *Transmit Timestamp* steht die Zeit, an der das Paket verschickt wurde.

Die 32 Bit große Zahl einer Zeitmarke gibt die Millisekunden seit Mitternacht (GMT) an.

16.4.9 Information Request und Information Reply

Information Request

Typ=0x0F	Code=0x00	Prüfsumme
Identifier		Sequenz Nummer

Information Reply

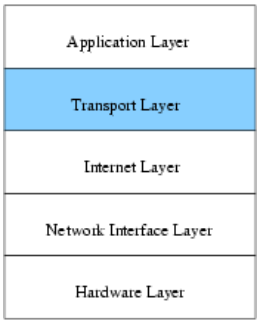
Typ=0x10	Code=0x00	Prüfsumme
Identifier		Sequenz Nummer

Mit dieser Nachricht lässt sich die Subnetzmaske und somit die Netzadresse eines Zieles erfahren. Wird als Zieladresse die IP-Adresse 0.0.0.0 angegeben, so antworten alle Rechner, die dieses Feature

unterstützen mit den Informationen des lokalen Netzes.[19] Man kann auf diese Weise die korrekte Einstellung der Netzmaske ferner Rechner überprüfen.

[19] Nicht alle Betriebssysteme unterstützen diesen Nachrichtentyp. Die Aufgabe, die dieser Funktion ursprünglich zugedacht wurde, wird inzwischen von anderen Protokollen wie RARP gelöst.

16.5 Transport Layer



Die *Transport Schicht* stellt die Schnittstelle zu den Applikationen dar und wird deshalb auch mit dem Begriff *end-to-end communication* beschrieben. Sie stellt der Applikation ein Transport-Service zur Verfügung – die Art des Services kann durch die Applikation bestimmt werden.

Im Header der Transport Schicht befinden sich u. a. eine *Destination-Portnummer* (*Ziel-Port)*, welche die Applikation im Zielrechner spezifiziert, und eine *Source-Portnummer* (*Quell-Port*), welche die Zuordnung zur lokalen Anwendung definiert.

Die Transport-Instanz teilt den Datenstrom in übertragbare kleinere Einheiten und leitet sie, versehen mit einer Zieladresse, an die nächste Schicht (Internet Layer) weiter. Die Transport-Schicht kümmert sich auch um die Fehlererkennung und deren Korrektur, sowie um die Datenflusskontrolle. Wie diese Funktionen im Detail aussehen, hängt von der Wahl des Protokolles ab. Es existieren 2 Protokolle in dieser Schicht: TCP und UDP. Sowohl TCP als auch UDP sind Bestandteil des Solaris-Kernels.

16.5.1 Protokoll-Typen der Transportschicht

Connection-Oriented Protocols (Verbindungsorientierte Protokolle) — Bei den verbindungsorientierten Protokollen muss vor der Übertragung der Daten eine Verbindung hergestellt werden.[20] Diese Methode bietet hohe Zuverlässigkeit, erfordert aber einen höheren 'Verwaltungsaufwand'. Zu diesem Protokolltyp wird TCP gezählt.

Connectionless Protocols (Verbindungslose Protokolle) — Mit verbindungslosen Protokollen ist ein vorheriger Verbindungsaufbau nicht nötig. Die Verwaltung ist sehr gering, die Methode deshalb sehr schnell. Nachteilig ist, dass die Übertragung keine Garantie bietet, dass die Daten vollständig ankommen. Falls trotz Verwendung eines verbindungslosen Protokolles Zuverlässigkeit gefordert wird, muss die Applikation selbst dafür Sorge tragen.[21] Ein Vertreter dieser Protokolle ist UDP.

[20] Man kann sich das vorstellen, wie eine Telefonverbindung. Bevor ich mit meinem Gegenüber sprechen kann, muss ich dessen Nummer wählen und abwarten, bis abgehoben wird. Dann können (Sprach-) Daten nach Belieben ausgetauscht werden.

[21] Ein Beispiel für die Verwendung eines verbindungslosen Protokolls trotz geforderter Zuverlässigkeit ist *tftp* (Trivial File Transfer Protocol). tftp wird z. Bsp. bei Diskless Systemen verwendet. Durch die Verwendung von UDP (einem 'primitiven', verbindungslosen Protokoll) findet die nötige Software auf einem PROM leicht Platz. Die Übertragungssicherheit wurde in den Programmcode des tftp gepackt.

Stateful Protocols Bei einem *stateful protocol* kennen beide 'Teilnehmer' den jeweiligen Zustand des anderen.

Stateless Protocols Beim *stateless protocol* kennt der Server den Zustand des Clients nicht. Der Server ist nicht in der Lage sicherzustellen, dass Daten vollständig und in der richtigen Reihenfolge ankommen. Die Vorteile eines *stateless protocols* sind die Einfachheit und ein gewisser Grad an Isolierung zwischen Server und Client. *Connectionless* Protokolle sind *stateless*.

Reliable Protocols Zuverlässige Protokolle wie z. Bsp. TCP verlangen, dass jedes Datenpaket, welches übertragen wurde, vom Empfänger bestätigt werden muss. Der Sender wiederholt die Übertragung bei Bedarf.

Unreliable Protocols Unzuverlässige Protokolle wie UDP fordern vom Empfänger keine Bestätigungen ein.

16.5.2 UDP – User Datagram Protocol

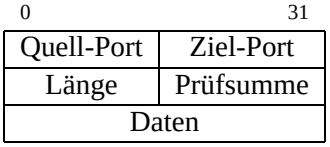
Das *User Datagram Protocol* ist ein connectionless, stateless Protokoll. Es ist für kleine Datenmengen und für Daten, deren korrekte Übertragung nicht zwingend notwendig ist, gedacht.

UDP Pakete können verloren gehen und die Reihenfolge bei der Ankunft der Daten kann durcheinander sein. Es ist Aufgabe der Applikation dies zu berücksichtigen. Über UDP kann die Applikation durch bloße Angabe von Quell- und Zielport auf die Internet-Schicht zu greifen.

Da UDP keine Bestätigung der einzelnen Datenpakete einfordert (*Non Acknowledged*), stellt es eine äußerst schnelle Datenübertragung zur Verfügung.

UDP teilt von der Applikationsschicht entgegengenommene Daten in *Datagramme*. Diese Datagramme bestehen aus einem Header, welcher Kontrollinformationen, Quell- und Zielport beinhaltet und einem Datenbereich. Die fertig aufbereiteten Datagramme werden an die Internet-Schicht (meistens handelt es sich dabei um das IP-Protokoll) weitergereicht.

16.5.2.1 UDP-Header (RFC 768)



Quell-Port Referenz für die lokale Applikation

Ziel-Port Referenz für die Applikation am Empfangsrechner

Länge Länge des kompletten Datagrammes inklusive Kopf in Bytes

Prüfsumme Prüfsumme des Datagramms

16.5.3 TCP – Transmission Control Protocol

Das *Transmission Control Protocol* ist ein 'verbindungsorientiertes', 'stateful' Protokoll. Es wurde konzipiert für die Übertragung von großen Datenmengen, welche auch über Router hinweg Zuverlässigkeit garantiert.

TCP ist durch folgende Eigenschaften gekennzeichnet:

- Explizite und bestätigter Verbindungsaufbau und -beendigung (*Virtual Circuit Connection*)
- Vollduplex-fähige virtuelle Verbindung
- Transparente Datenübertragung (*unstructured stream orientation*)
- Zuverlässigkeit durch
  - Sequenznummern
  - Prüfsumme
  - Quittierungszeitüberwachung
  - Segmentwiederholung
- Flusskontrolle durch Empfangsfensterangabe (*sliding-window* Verfahren)
- Effizienz durch Datenbuffer
- Urgent Data Feature (*Out of band* Daten)
- Applikationsadressierung durch 16-Bit Portnummer

Unstructured Stream Orientation Der TCP-Stack erhält von der Applikation einen Datenstrom. Dieser wird von TCP in Pakete zerteilt, welche von der Internetschicht übertragen werden können. Der Inhalt der Daten wird nicht interpretiert, sondern transparent in mit Headerinformation in ein TCP-Paket verpackt und in derselben Reihenfolge, wie sie von der Applikation erhalten wurden an die Internet-Schicht weitergeleitet.

Virtual Circuit Connection Bevor eine Datenübertragung zwischen den Applikationen am Sender und Empfänger stattfinden kann, müssen diese beim über das Betriebssystem einen Verbindungskanal schaffen. Dies ist vergleichbar mit dem Herstellen einer Verbindung beim Telefonieren bevor gesprochen werden kann.

Buffered Transfer Da der Datenfluss zwischen den Applikationen nicht gleichmäßig sein muss, sondern einmal schnell und dann wieder langsam erfolgt, werden im System Buffer verwendet, die die Effizienz steigern.

Vollduplex Verbindung Aus der Sicht der Applikation wird von einer Vollduplex Verbindung gesprochen, wenn Senden und Empfangen völlig unabhängig voneinander ablaufen. TCP ist in der Lage das zu gewährleisten. Bestätigungen für den Empfang von Segmenten können zusammen mit neuen Daten in einem TCP-Paket an die Gegenstelle geschickt werden (auch '*piggy backing*' genannt, da die Quittung praktisch 'huckepack' mit den Daten mitgenommen wird).

16.5.3.1 TCP Fluss-Kontrolle

TCP leistet mehr, als nur ein simples Senden, Empfangen und Bestätigen von Datenpaketen. Es enthält auch spezielle Algorithmen, um den Datenfluss zu optimieren.[22] Realisiert wird das durch das sogenannte *sliding window* Verfahren. Jedes TCP-Paket mit einer Bestätigung enthält eine *Fenster Größe*, die angibt wie viele Bytes der Empfänger noch bereit ist entgegenzunehmen. Sinkt die Fenstergröße auf Null, so bedeutet das für den Sender, dass er mit den folgenden Daten etwas warten muss.

Beim Verbindungsaufbau wird zwischen Sender und Empfänger die maximale Segmentgröße (mittels TCP-Option '*maximal segment size*') ausgehandelt[23] und der Empfänger teilt dem Sender einen initialen Sendekredit (im Feld 'Fenster Größe') mit. Nun schickt der Sender mehrere Datensegmente bis der Sendekredit aufgebraucht ist. Der Empfänger bestätigt die empfangenen Pakete indem er die Sequenznummer des zuletzt erhaltenen Segmentes um 1 erhöht und als 'Acknowledge Nummer' zurückschickt. War der Empfänger noch nicht in der Lage die empfangenen Pakete zu verarbeiten und den Puffer zu leeren, so schickt er im 'Quittungspaket' einen Sendekredit mit dem Wert Null an den Sender. Erst wenn der Sender wieder einen Sendekredit größer Null empfängt, darf er wieder Daten schicken.

Das im TCP-Header vorgesehene Feld für die Fenstergröße ist 16 Bits groß und erlaubt deshalb eine maximalen Wert von $2^{16} = 65535$. Solaris unterstützt die in RFC 1323[24] vorgeschlagene Methode, um größere Fenster (max. $2^{30} = 1GByte$) zuzulassen. Dadurch kann der Durchsatz in Hochgeschwindigkeitsnetzen wie ATM oder FDDI erheblich gesteigert werden.

Stauvermeidung, Congestion Window: Um Datenstaus zu vermeiden, unterstützt TCP ein sogenanntes Staukontrollfenster (*congestion window*). Es bestimmt, wie viele Segmente unbestätigt auf die Leitung geschickt werden dürfen. Um den Durchsatz zu erhöhen, ist es gewünscht nicht nach jedem Paket auf eine Quittung zu warten. Das TCP-Protokoll erlaubt ja die Bestätigung von mehreren Paketen indem es das letzte einer ganzen Serie von Segmenten bestätigt. Allerdings eröht sich die Netzlast, wenn durch wiederholte 'Staus' laufend Sendewiederholungen nötig sind. Zu Beginn einer Übertragung wird das Staukontrollfenster auf den Wert '1' gesetzt, d. h. es darf nur ein Segment gesendet werden, bis die Bestätigung empfangen wird. Mit jeder empfangenen Bestätigung wird das Staukontrollfenster um ein Segment vergrößert ('*slow start process*'). Kommt es zu einem Stau, wird das Staukontrollfenster um die Hälfte reduziert und der Timer für Sendewiederholungen exponentiell vergrößert.

```
Quelle Ziel      Info
-------------------------------------------------------------------
SRV01  pc07  [SYN] Seq=3608979481 Ack=0 Win=5840 Len=0 mss=1460
pc07   SRV01 [SYN, ACK] Seq=3594982721 Ack=3608979482 Win=32120 \
                          Len=0 mss=1460
SRV01  pc07  [ACK] Seq=3608979482 Ack=3594982722 Win=5840 Len=0
SRV01  pc07  [ACK] Seq=3608979482 Ack=3594982722 Win=5840 Len=1448
pc07   SRV01 [ACK] Seq=3594982722 Ack=3608980930 Win=31856 Len=0
```

[22] Der Grund für eine nötige Flusskontrolle muss nicht ein überlasteter oder leistungsschwacher Empfangsrechner sein. Eine Applikation, die z. Bsp. Multimediadaten (Audio- und/oder Filmdaten) ausgibt, ist gezwungen, dies in der entsprechenden Geschwindigkeit zu tun, auch wenn der Rechner in der Lage wäre die Daten um ein Vielfaches schneller auszugeben.

[23] Standardmäßig(ohne Angabe der Option) beträgt die maximale Segmentgröße für TCP-Pakete 536 Bytes.

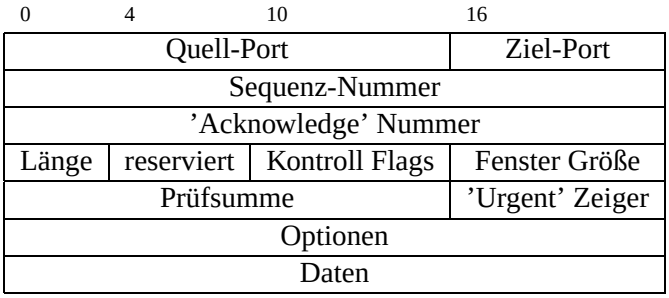
[24] RFC 1323 beschreibt eine Reihe von Maßnahmen, welche die Performance von TCP steigern.

```
SRV01  pc07  [ACK] Seq=3609040714 Ack=3594982722 Win=5840 Len=1448
SRV01  pc07  [ACK] Seq=3609042162 Ack=3594982722 Win=5840 Len=1448
pc07   SRV01 [ACK] Seq=3594982722 Ack=3609043610 Win=1448 Len=0
SRV01  pc07  [ACK] Seq=3609043610 Ack=3594982722 Win=5840 Len=1448
pc07   SRV01 [ACK] Seq=3594982722 Ack=3609045058 Win=0 Len=0
SRV01  pc07  [ACK] Seq=3609045057 Ack=3594982722 Win=5840 Len=0
pc07   SRV01 [ACK] Seq=3594982722 Ack=3609045058 Win=0 Len=0
pc07   SRV01 [ACK] Seq=3594982722 Ack=3609045058 Win=4344 Len=0
SRV01  pc07  [ACK] Seq=3609045058 Ack=3594982722 Win=5840 Len=1448
SRV01  pc07  [ACK] Seq=3609046506 Ack=3594982722 Win=5840 Len=1448
pc07   SRV01 [ACK] Seq=3594982722 Ack=3609047954 Win=2896 Len=0
```

Obiges Listing zeigt einen Ausschnitt einer Datenübertragung per TCP-Protokoll zwischen den beiden Rechnern mit den Namen ’SRV01’ und ’pc07’. Auf ’pc07’ wartete ein Prozess auf einem TCP-Port auf ankommende Datenpakete. ’SRV01’ beginnt mit einem Paket, in dessen TCP-Header das Flag ’SYN’ gesetzt ist. Der Sendekredit, den ’SRV01’ an seinen Kommunikationspartner ’pc07’ vergibt, beträgt 5840 Bytes. Außerdem wird als Option ’mss=1460’ (maximal segment size) mitgeschickt. Da ein wartender Prozess für das gewünschte Port existiert, antwortet ’pc07’ mit gesetztem ’SYN’ und ’ACK’-Flag. Die Acknowledge-Nummer ist um 1 höher als die empfangene Sequenz-Nummer und bestätigt somit den korrekten Empfang. Auch ’pc07’ teilt die maximale Segmentgröße in den Optionen mit und vergibt einen Sendekredit von 32120 Bytes im Feld ’Window’. Durch diese Bestätigung mit den beiden Flags ’SYN’ und ’ACK’ weiß ’SRV01’, dass die TCP-Verbindung erfolgreich aufgebaut wurde.[25]

’SRV01’ beginnt nun, Datensegmente an ’pc07’ zu senden. Schon bei der Bestätigung der ersten beiden Segmente, die von ’SRV01’ empfangen wurden, ist erkennbar, dass der Sendekredit langsam sinkt (Win=31856 gegenüber Win=32120 bei der vorherigen Antwort). Nach einiger Zeit sinkt der Sendekredit bis auf die Größe von Null Bytes. Sobald ’pc07’ wieder bereit war, Daten anzunehmen, wurde ein Paket mit der Fenstergröße von 4344 Bytes geschickt. Nun darf ’SRV01’ wieder Daten übertragen.

16.5.3.2 TCP-Header (RFC 793)



Quell Port (16 Bits) – Das *Sourceport* kennzeichnet den Prozess des Senders

Ziel Port (16 Bits) – Das *Destinationport* bestimmt den Prozess im Empfangssystem.

Sequenznummer (32 Bits) – Die Sequenznummer gibt die Stellung innerhalb des Datenstromes an, damit der Empfänger die Teilpakete wieder in der richtigen Reihenfolge zusammensetzen kann.

[25] Mit dem Kommando ’netstat’ ist diese Verbindung nun mit dem Zustand ’ESTABLISHED’ sichtbar.

Quittungsnummer (32 Bits) – Die 'Acknowledge-Number' dient der Bestätigung des Empfanges eines Paketes.

Länge (4 Bits) – Da eine unterschiedliche Anzahl von Optionen möglich ist, ist die Header-Größe nicht fix. Die Größe des aktuellen TCP-Headers wird in diesem Feld angegeben.

reserviert (6 Bits) – für mögliche zukünftige Nutzung reservierte Bits

Kontroll Flags Die folgenden 6 Flags (je 1 Bit) sind definiert:

- **URG** *Urgent Pointer* – zeigt, dass Daten vorhanden sind, die sofort weitergeleitet werden müssen. Die Position der Daten steht im Feld '*Urgent Pointer*'.
- **SYN** *Synchronization* – Ist dieses Bit auf 1 gesetzt, wird damit der Wunsch eines Verbindungsaufbaues gekennzeichnet.
- **ACK** *Acknowldege* – zeigt, dass der Wert im Feld Acknowledge-Number gültig ist
- **RST** *Reset* – Ist dieses Flag gesetzt, ist dies eine Aufforderung, die Verbindung zurückzusetzen.
- **PSH** *Push* – Das Push-Flag kennzeichnet ein Datenpaket, das sofort an die Applikation weitergeleitet werden.
- **FIN** *Final* – das gesetzte Bit kennzeichnet die Aufforderung zum Verbindungsabbau

Fenster Größe (16 Bits) – Das Feld wird für die Datenflusskontrolle verwendet. Der Wert gibt die Datenmenge an, die der Empfänger noch aufnehmen kann.

Prüfsumme (16 Bits) – Die Prüfsumme dient der Kontrolle, ob Daten korrekt übertragen wurden.

Urgent Pointer (16 Bits) – zeigt auf das lezte Byte von wichtigen Daten, die sofort an die nächste Instanz weitergeleitet werden müssen.

Optionen (variabel) – Hier können Optionen, wie z. Bsp. die maximale Segmentgröße hinterlegt werden.

16.5.4 Gegenüberstellung von UDP und TCP

Eigenschaften der beiden wichtigsten Transport Layer Protokolle UDP und TCP. Zusätzlich wurde auch das Internet Protokoll IP in die Tabelle aufgenommen, obwohl IP kein Protokoll der Transportschicht ist, sondern der darunterliegenden Internetschicht zugehört.

Eigenschaft	IP	UDP	TCP
Verbindungsorientiert?	nein	nein	ja
Nachrichtengrenzen?	ja	ja	nein
Daten Prüfsumme?	nein	optional	nein
Positive Rückmeldung?	nein	nein	ja
Timeout und Sendewiederholung?	nein	nein	ja
Erkennen von doppelter Übertragung?	nein	nein	ja
Überwachung der Reihenfolge?	nein	nein	ja
Überwachung des Datenflusses?	nein	nein	ja
Voll-duplex?	nein	nein	ja

16.6 Wichtige Netzwerkdateien

16.6.0.1 /etc/inet/hosts Datei

In dieser Datei erfolgt die Zuordnung von sprechenden Rechnernamen zu IP-Adressen. Damit können Rechner mit ihren Namen anstatt der Adressen angesprochen werden. Die Datei */etc/hosts* ist ein symbolischer Link auf die Datei */etc/inet/hosts*.

Beispiel einer hosts-Datei:

```
1  #
2  # Internet host table
3  #
4  127.0.0.1       localhost
5  #193.16.214.28  sunlicht        sunlicht.nodom   loghost
6  193.16.214.28   sunlicht        loghost
7  193.16.214.23   sunpc
8  193.16.214.204  whl
9  193.16.214.48   Distl
10 195.2.57.91     v223mx29
11 193.16.214.30   hotline hot
12 193.16.214.129  pcchd
13 149.202.162.77  pisch
14 10.1.1.2        suntest
```

Der erste Eintrag[26] (*127.0.0.1 localhost*) hat eine Sonderbedeutung. Die Netzadresse '127' ist reserviert für den sogenannten *localhost*. Dieser Eintrag existiert, damit Software, welche auf Netzwerkschnittstellen aufsetzt, auch dann mit dem lokalen Rechner läuft, wenn keine Netzwerk-Hardware vorhanden ist. Über den Eintrag *localhost* kann der eigene Rechner angesprochen werden.

Die nächste Zeile beschreibt den eigenen Rechner (*193.16.214.28 sunlicht loghost*). *193.16.214.28* ist die IP-Adresse, welche die Netzwerkkarte haben soll. *sunlicht* ist der Rechnername. *loghost* wird von *syslogd* verwendet (siehe Kapitel 6.3, Seite 51). Das ist jener Rechnername, an den *syslogd* Log-Informationen schickt (im Standardfall der eigene Rechner).

Alle weiteren Einträge beschreiben Rechner, welche durch einen Hostnamen ansprechbar sein sollen. Zeilen, in denen 2 oder mehr Namen zu finden sind, definieren zusätzlich zum eigentlichen Rechnernamen noch einen Aliasnamen. Der Zielrechner kann somit auch mit dessen Aliasnamen angesprochen werden. Zu beachten ist, dass bei der Umwandlung einer IP-Adresse zu einem Hostnamen immer der erste Name des Eintrages verwendet wird.

16.6.1 /etc/hostname.hme0

Für jede installierte Netzwerkkarte muss eine Datei */etc/hostname.xxx* existieren (*xxx* steht für den *Instance Name* der Karte - siehe Kapitel 8.1, Seite 67). Beim Hochfahren des Systems wird nach Dateien gesucht, deren Name mit */etc/hostname.* beginnt. Anhand des Rechnernamens, der in diesen Dateien steht, wird der Netzwerkkarte aus */etc/inet/hosts* die zugeordnete IP-Adresse vergeben.

[26] Es ist nicht von Bedeutung an welcher Stelle die Einträge vorgenommen werden. Üblicherweise stehen aber *localhost* und der eigene Rechnername mit dessen IP-Adresse am Beginn der hosts-Datei.

```
root@sunlicht:/etc > more hostname.*
::::::::::::::
hostname.hme0
::::::::::::::
sunlicht
::::::::::::::
hostname.hme0:1
::::::::::::::
suntest
root@sunlicht:/etc >
```

In diesem Beispiel finden wir noch eine Besonderheit: Der Netzwerkkarte *hme0* werden bei unserem Rechner 2 IP-Adressen vergeben. Neben der Datei */etc/hostname.hme0* existiert noch eine Datei */etc/hostname.hme0:1*. Der darin enthaltene Rechner *suntest* trägt in der Datei */etc/inet/hosts* die IP-Adresse *10.1.1.2*. Es ist möglich einer Netzwerkkarte mehrere IP-Adressen zu vergeben. Dies geschieht dadurch, dass zum eigentlichen *Instance Name* der Netzkarte noch ein Doppelpunkt und eine Zahl (aufsteigend bei 1 beginnend) angehängt wird.

16.6.2 /etc/nodename

Darin enthalten ist der wirkliche Name des Hosts. Ein Host kann mehrere Aliasnamen haben, aber nur einen *Canonical Name*. Der Name wird während des Systemstarts vom Skript */etc/rcS.d/S30rootusr.sh* ausgelesen und weiterhin als Hostname verwendet.[27]

Beispiel:

```
root@sunlicht:/etc > more nodename
sunlicht
root@sunlicht:/etc >
```

16.6.3 /etc/passwd

Wird für eine Verbindung eine Benutzerkennung benötigt, so wird zuerst die Datei */etc/passwd* untersucht, ob diese Kennung existiert. Anschließend wird das Passwort überprüft. Existiert die Benutzerkennung nicht, so wird der Zugriff abgewiesen.

16.6.4 /etc/hosts.equiv

Diese Datei ist nach der Installation des Betriebssystemes nicht vorhanden und muss bei Bedarf angelegt werden. Sie enthält Namen von *trusted hosts* und *trusted users* (vertrauenswürdige Rechner und Benutzer) . Man kann dadurch festlegen, wer eine Verbindung ohne Passwortabfrage herstellen darf. Das bezieht sich auf die '*r-Kommandos*' wie *rsh, rlogin* und *rcp*. Die Datei hat keinen Einfluss auf die Kommandos *telnet* oder *ftp*.

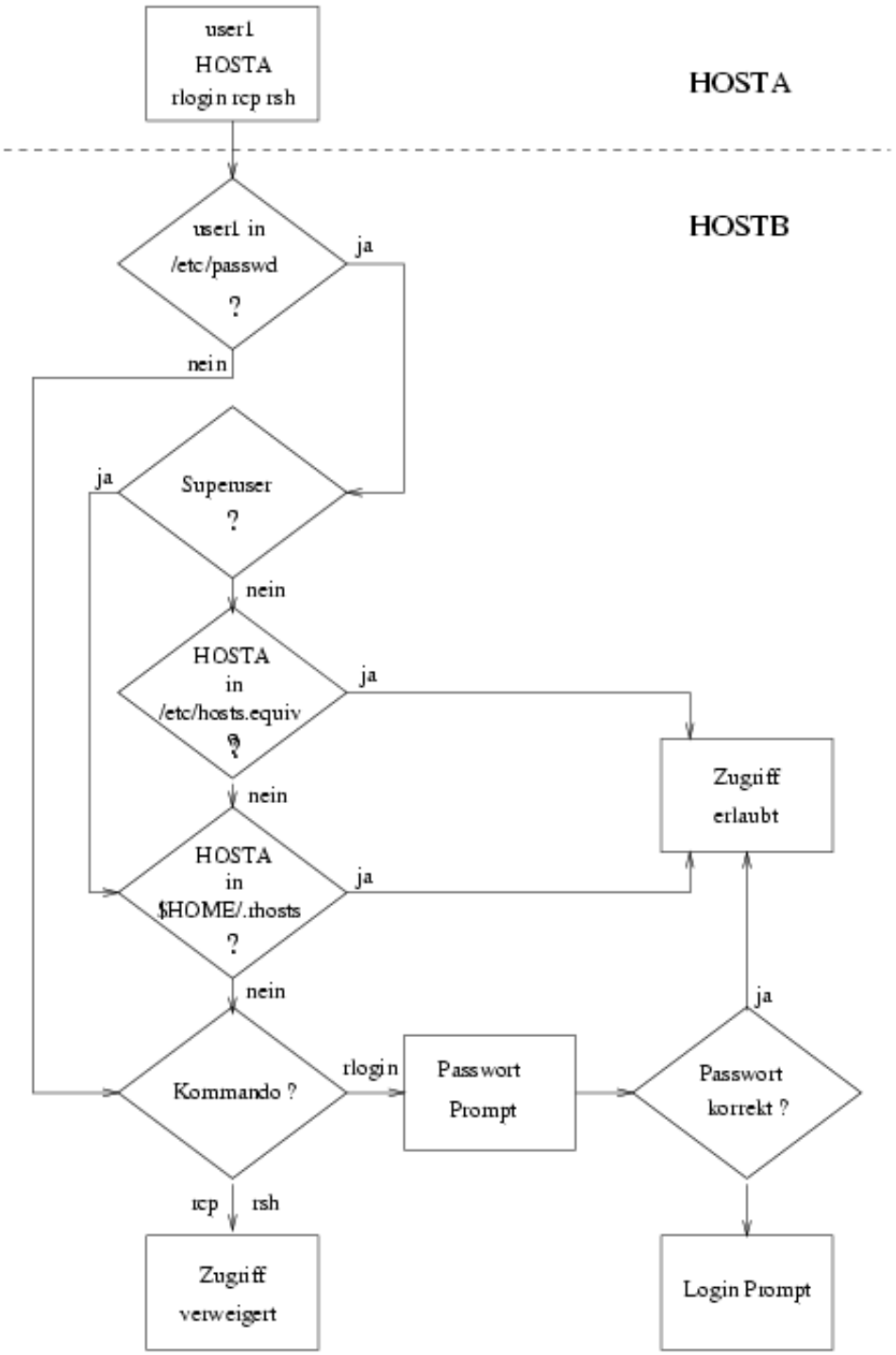
Die Datei verwendet folgendes Format:

[+|-] [HOSTNAME] [USERNAME]

HOSTNAME ist ein *trusted host*, d.h. alle Benutzer dieses Hosts können eine Verbindung ohne Passwortabfrage herstellen, wenn deren Username auch lokal gleichlautend existiert.

[27] Siehe auch den Tipp „Rechnernamen ändern“ auf Seite 284.

Abbildung 16.11: Remote Zugriff Authentifizierung



Steht das Zeichen '+' für sich alleine an Stelle des *HOSTNAME*, so werden alle Hosts zu *trusted hosts*!! VORSICHT!! Das ist eine gefährliche Sicherheitslücke. Ein Schreibfehler in dieser Datei kann leicht das System für jeden öffnen!

Bei Angabe von *USERNAME* wir diesem User des fernen Systems gewährt, sich unter jeder beliebigen lokalen Benutzerkennung (außer Root) ohne Passwort anzumelden.[28] Wird das Zeichen '-' vor einen Usernamen gestellt, so wird diesem User eine trusted Beziehung verweigert. Das bedeutet, dieser User kann sich niemals ohne Passwortabfrage anmelden.

Beispiele für Einträge in der Datei */etc/hosts.equiv*:

Alle Rechner sind *trusted hosts:*

```
+
```

Die beiden Rechner 'neptune' und 'pluto' sind *trusted hosts*:

```
neptune
pluto
```

Der Benutzer 'rimmer' vom Rechner 'pluto' hat Zugriff auf jede Benutzerkennung (außer root) ohne Passwortabfrage:

```
pluto rimmer
```

16.6.5 $HOME/.rhosts

Beim Zugriff von einem fremden Host mittels *r-Kommando* (rlogin, rsh, rcp) wird überprüft, ob im Home-Verzeichnis des lokalen Users eine Datei *.rhosts* existiert. Existiert diese Datei und enthält sie einen Eintrag mit dem Namen des fernen Hosts, so ist dieser ein *trusted host*. Standardmäßig existiert keine Datei *.rhosts*. Die Datei muss bei Bedarf erzeugt werden. Es ist darauf zu achten, dass die Zugriffsrechte auf *600* – also Nurleserecht für den Eigentümer – gesetzt werden.

Format der Dateieinträge:

[+|-] [HOSTNAME] [USERNAME]

Existiert ein Eintrag *HOSTNAME*, so könne alle Benutzer dieses Rechners auf den lokalen Rechner ohne Passwortabfrage zugreifen, sofern lokal auch eine gleichnamige Benutzerkennung existiert.

Steht anstelle von *HOSTNAME* nur das Zeichen '+', so können sich Benutzer von jedem Rechner aus ohne Passwortabfrage lokal anmelden. (Das verursacht eine schwerwiegende Sicherheitslücke!)

Steht neben *HOSTNAME* noch ein Eintrag *USERNAME*, so ist es nur diesem User erlaubt, sich lokal ohne Passwort anzumelden. Derselbe Eintrag in die Datei */etc/hosts.equiv* hätte eine andere Bedeutung!

Jeder Benutzer kann selbst eine Datei *.rhosts* erstellen und somit den Zugriff von fernen Rechnern erlauben. Das ist ein mögliches Sicherheitsrisiko und kann vom Systemverwalter durch eine Änderung in der Datei */etc/pam.conf* verhindert werden. Wird die Zeile `'rlogin auth sufficient /usr/lib/security/pam_rhosts_auth.so.1'` durch das Zeichen '#' am Beginn der Zeile auskommentiert, so werden allfällig vorhandene Dateien *$HOME/.rhosts* nicht bewertet.

[28] ACHTUNG! Dieselbe Syntax hat in der Datei $HOME/.rhosts eine andere Bedeutung!

16.7 Netzzugriff Kommandos (Remote Access Commands)

Damit die im Folgenden beschriebenen Kommandos funktionieren, müssen die entsprechenden Serverprogramme in der Datei */etc/inetd.conf* aktiviert sein. Standardmäßig ist das der Fall. Möchte man diverse Server aus sicherheitsgründen deaktivieren, so muss lediglich das Zeichen '#' am Beginn der entsprechenden Zeile gesetzt werden und der Prozess *inetd* neu gestartet werden oder per Signal *HUP* (kill -HUP PID) zum erneuten Einlesen der Konfigurationsdatei gezwungen werden.

Eine Auflistung der Datei */etc/inetd.conf* ist im Kapitel 16.11.2.1 auf Seite 234 zu finden.

16.7.1 rlogin

rlogin dient dazu, eine Anmeldung auf einem fernen Rechner durchzuführen.

Syntax:

rlogin *hostname|IP-Adresse* [-l *username*]

hostname Rechnername des fernen Rechners. Der Name muss in der Datei */etc/inet/hosts* eingetragen sein.

IP-Adresse Anstatt eines Rechnernamens kann auch direkt die IP-Adresse angegeben werden.

-l *username* Benutzerkennung mit der die Anmeldung erfolgen soll.

16.7.2 rsh

rsh wird verwendet, um ein Programm auf einem fernen Rechner zu starten.

Syntax:

rsh [-l *username*] *hostname|IP-Addresse command*

-l *username* Benutzerkennung unter dem das Kommando am fernen Rechner ausgeführt werden soll.

hostname Rechnername des fernen Rechners

IP-Adresse Anstatt des Rechnernamens kann auch die IP-Adresse des fernen Rechners angegeben werden.

command Das am fernen Rechner zu startende Programm.

ACHTUNG! Das gleichnamige Kommando *rsh* im Verzeichnis */usr/lib* hat eine völlig andere Funktion. */usr/lib/rsh* ist eine *Restricted Shell* (eingeschränkte Shell). Sie wird verwendet, um die Sicherheit in einem System zu erhöhen. Diese Shell erlaubt nicht:

- Wechsel in ein anderes Verzeichnis mit dem Kommando *cd*
- Änderung der Umgebungsvariable $PATH
- Verwendung eines Kommandos, welches das Zeichen '/' enthält. –> Nur die Kommandos, welche durch die Variable $PATH 'erreichbar' sind, können aufgerufen werden.
- Umlenkung der Ausgabe ('>' oder '>>')

16.7.3 rcp

Mit dem Kommando *rcp* können Daten von bzw. zu einem fernen Rechner kopiert werden.

Syntax:

rcp [-r] *source_file hostname*:*destination_file*

rcp [-r] *hostname*:*source_file destination_file*

-r rekursives Kopieren bei Verzeichnissen

16.7.4 telnet

telnet ermöglicht ebenso wie *rlogin* eine Anmeldung an einem fernen Rechner. Auch für *telnet* muss am fernen Rechner ein Dämonprozess auf einen Verbindungswunsch warten. Im Gegensatz zu *rlogin* wird bei *telnet* immer nach einem Passwort gefragt, welches somit auch übertragen wird und u.U. mitgelesen werden kann. Die Dateien */etc/hosts.equiv* und *$HOME/.rhosts* werden vom Telnet-Serverprozess nicht ausgewertet.

Syntax:

telnet *hostname|IP-Adresse* [*portnummer*]

hostname Name des fernen Rechners

IP-Adresse Anstatt des Namens kann auch die IP-Adresse des fernen Rechners angegeben werden

portnummer Ohne Angabe wird die für telnet übliche Portnummer 23 verwendet.

Da das Kommando *telnet* die Angabe der Portnummer erlaubt, eignet sich *telnet* auch, um zu testen, ob ein bestimmtes Port eines Rechners 'offen' ist. Z.Bsp. kann gefragt sein, ob ein Datenbankserver auf einem Host läuft. Ist dessen Portnummer bekannt, so kann mit telnet getestet werden, ob eine Verbindung hergestellt werden kann.

Unterbrechen kann man eine telnet-Verbindung mit der Tastenkombination <CTRL>+<]>.

16.7.5 ftp

ftp (*file transfer protocol*) ermöglicht die Datenübertragung von und zu fernen Rechnern.

Syntax:

ftp *hostname|IP-Adresse*

Nach einem erfolgreichen Verbindungsaufbau mit *ftp* zu einem fernen Rechner, erscheint ein Prompt. Durch die Eingabe des Fragezeichens '?' kann eine Liste aller verfügbaren Kommandos aufgelistet werden.

Die wichtigsten ftp-Kommandos sind:

cd Verzeichniswechsel am fernen Rechner

lcd Verzeichniswechsel am lokalen Rechner

ls Auflistung der Dateien am fernen Rechner

!*command* Kommando am lokalen Rechner ausführen. z.Bsp. !ls

binary Dateien werden binär übertragen (es erfolgt keine Transformation von Daten bei der Übertragung). Textdateien werden ansonsten in das Betriebssystem-übliche Format umgewandelt.

get *filename* Datei vom fernen System auf lokales System kopieren (holen).

put *filename* Datei vom lokalen System zum fernen System senden

mget filename,.. (multiple get) mehrere Dateien 'auf einmal' holen

mput filename,.. (multiple put) mehrere Dateien 'auf einmal' senden

prompt *prompt* wechselt den Eingabemodus für 'multiple commands' wie mget und mput. Der Interaktivmodus wird je nach aktuellem Zustand aktiviert bzw. deaktiviert.

quit Beenden einer ftp-Session

Beispielsitzung mit ftp:

```
1  pische@pisch:~/SNI/solaris/kommandos > ftp sunlicht
2  Connected to sunlicht.
3  220 sunlicht FTP server (SunOS 5.6) ready.
4  Name (sunlicht:pische): pische
5  331 Password required for pische.
6  Password:
7  230 User pische logged in.
8  Remote system type is UNIX.
9  Using binary mode to transfer files.
10 ftp > ?
11 Commands may be abbreviated.  Commands are:
12
13 !               debug           mdir            sendport        site
14 $               dir             mget            put             size
15 account         disconnect      mkdir           pwd             status
16 append          exit            mls             quit            struct
17 ascii           form            mode            quote           system
18 bell            get             modtime         recv            sunique
19 binary          glob            mput            reget           tenex
20 bye             hash            newer           rstatus         tick
21 case            help            nmap            rhelp           trace
22 cd              idle            nlist           rename          type
23 cdup            image           ntrans          reset           user
24 chmod           lcd             open            restart         umask
25 close           ls              prompt          rmdir           verbose
26 cr              macdef          passive         runique         ?
27 delete          mdelete         proxy           send
28 ftp > get SUN-Logo.gif
29 local: SUN-Logo.gif remote: SUN-Logo.gif
```

```
200 PORT command successful.
150 Binary data connection for SUN–Logo.gif (149.202.162.77,1745) (3440
    bytes).
226 Binary Transfer complete.
3440 bytes received in 0.167 secs (20 Kbytes/sec)
ftp> quit
221 Goodbye.
pische@pisch:~/SNI/solaris/kommandos >
```

Ob per *ftp* einer Benutzerkennung der Zugriff erlaubt wird oder nicht, kann in der Datei */etc/ftpusers* festgelegt werden. Jeder Benutzerkennung, die in dieser Datei eingetragen ist, wird der Zugriff verweigert. Standardmäßig sind *root* und eine Reihe anderer Systemkennungen eingetragen.

Verwendet eine Benutzerkennung eine Shell, welche nicht in der Datei */etc/shells* eingetragen ist[29], so kann per ftp keine Verbindung mit dieser Benutzerkennung hergestellt werden! Damit das funktioniert, muss die entsprechende Shell zuerst in */etc/shells* bekanntgemacht werden.

Inhalt von */etc/shells*, welche um die beiden Shells *dtksh* und *bash* erweitert wurde:

```
/usr/bin/sh
/usr/bin/csh
/usr/bin/ksh
/usr/bin/jsh
/bin/sh
/bin/csh
/bin/ksh
/bin/jsh
/sbin/sh
/sbin/jsh
/usr/dt/bin/dtksh
/usr/local/bin/bash
```

16.7.6 rusers

Anzeige von angemeldeten Benutzern auf anderen Rechnern im Netz.

Syntax:

rusers [-a] [-l] [*hostname*]

-a Zeige alle Rechner an, auch dann wenn dort niemand angemeldet ist

-l lange Ausgabe

hostname Nur Daten des angegebenen Rechners werden angezeigt

Beispielausgabe von *rusers*:

```
root@sunlicht:~ > rusers
Sending broadcast for rusersd protocol version 3...
```

[29] Zum Beispiel *dtksh*, welche im System inkludiert ist, oder die freie Shell *bash*, welche sehr beliebt ist.

```
Sending broadcast for rusersd protocol version 2...
hotline winklerf winkler zam walter telesv
mxtele.erd.sni.a teleserv root event
193.16.214.13.4. root em9750 svpst0 cons0 event cons0 svpst0
193.16.214.45.4. em9750 cons0 svpst0 event cons0 svpst0
root@sunlicht:~ >
```

16.7.7 ping

/usr/sbin/ping sendet ein ICMP (Internt Control Message Protocol) ECHO_REQUEST Paket zum gewünschten Host und wartet auf dessen Antwort – ein ICMP ECHO_RESPONSE. Antwortet der Host, so gibt *ping* die Meldung '*host ist alive*' aus. Andernfalls bricht das Kommando nach einer Timeout-Zeit von ca. 20 Sekunden mit einer Fehlermeldung ab.[30] Es ist damit feststellbar, ob ein Host übers Netzwerk erreichbar ist.

ACHTUNG: Das ist keine sichere Methode! Befindet sich dazwischen eine Firewall, so lässt diese u.U. ICMP-Pakete nicht durch. Teilweise kann man auch direkt im Betriebssystem eine Antwort auf ECHO_REQUEST unterbinden (z.Bsp. möglich bei LINUX).

Syntax:

ping [-s] *hostname|IP-Adresse*

-s Es werde endlos Pakete geschickt

hostname Name des Rechners, der überprüft werden soll. Stattdessen kann auch die IP-Adresse angegeben werden.

Beispiel des ping-Kommandos:

```
root@sunlicht:~ > ping pisch
pisch is alive
root@sunlicht:~ > ping -s pisch
PING pisch: 56 data bytes
64 bytes from pisch (149.202.162.77): icmp_seq=0. time=42. ms
64 bytes from pisch (149.202.162.77): icmp_seq=1. time=56. ms
64 bytes from pisch (149.202.162.77): icmp_seq=2. time=34. ms
^C
----pisch PING Statistics---- 3 packets transmitted, 3 packets received, 0% packet loss
round-trip (ms) min/avg/max = 34/44/56
root@sunlicht:~ >
```

16.7.8 spray

Während *ping* auf sehr niedriger Ebene die Erreichbarkeit eines Rechners testet, ermöglicht *spray* einen Test auf höherer Ebene. */usr/sbin/spray* sendet Datenpakete zu einem Host und verwendet dabei

[30] Das Verhalten von *ping* ist Implementationsabhängig. Die Ausgabe und das Zeitverhalten unterscheiden sich zwischen den Betriebssystemen. Jedoch wird bei allen Varianten ein ECHO_REQUEST geschickt und ein ECHO_RESPONSE erwartet.

RPC (remote procedure call). Es werden die abgesendeten Pakete und die beantworteten Pakete gezählt und verglichen. Als Ergebnis erhält man den Datendurchsatz und den Prozentsatz an verlorenen Paketen.

Voraussetzung für das Funktionieren von *spray* ist, dass auf dem Zielrechner ein Serverprozess läuft, der die Datenpakete beantworten kann. Unter Solaris und vielen anderen UNIX-Derivaten wird der Serverprozess durch einen Eintrag in */etc/inetd.conf* aktiviert. Nicht jedes Betriebssystem besitzt einen Serverprozess!

Ist der Sender eine sehr schnelle Maschine und der Empfänger wesentlich langsamer, so ist es normal, dass Pakete verloren gehen.

Syntax:

spray [-c *count*] [-d *delay*] [-l *lengt*] *hostname|IP-Adresse*

-c *count* Anzahl der zu versendenden Pakete. Standard ist 1162 Pakete.

-d *delay* Wartezeit zwischen den gesendeten Paketen in Mikrosekunden. Standard ist 0 Mikrosekunden.

-l *length* Anzahl der Datenbytes in den gesendeten Paketen. Standard ist 86 Bytes.

hostname Rechner, der überprüft werden soll

16.7.9 netstat

netstat zeigt Netzverbindungen, Netzstatistiken, Routingtabellen, etc. an. Es ist ein wertvolles Kommando um Netzwerkprobleme einzugrenzen. Siehe auch 16.3.6 auf Seite 195!

Syntax:

netstat [-a] [-i [*interface*]] [-n] [-r] [-s]

Ohne Angabe einer Option werden alle bestehenden Verbindungen aufgelistet.

-a Zusätzlich zu den bestehenden Netzverbindungen werden 'offene' Ports aufgelistet.

-i [*interface*] Zusammenfassung von Statistikdaten für alle Interfaces wenn kein Interface angegeben wird bzw. für ein bestimmtes Interface. Vor allem ist der Wert der Kollisionen von Bedeutung. Der prozentuelle Anteil von Kollisionen (*Kollisionsrate*) sollte 5 % nicht überschreiten ($Kollisionsrate = 100\frac{Collis}{Opkts}$).

-n IP-Adressen werden nicht in Namen aufgelöst, es werden nur IP-Adressen aufgelistet. Falls DNS verwendet wir und der DNS-Server nicht erreichbar ist, so kann es zu extrem langen Verzögerungen kommen, da der Name nicht aufgelöst werden kann. Auch in diesem Fall ist es hilfreich, die Option '*-n*' anzugeben.

-r Listet Routingtabelle auf.

-s Detaillierte Statistik

Beispiele:

```
root@sunlicht:~ > netstat -i
Name  Mtu  Net/Dest      Address        Ipkts  Ierrs Opkts  Oerrs Collis Queue
lo0   8232 loopback      localhost      235317 0     235317 0     0      0
hme0  1500 sunlicht      sunlicht       271549 0     3929   0     15     0
hme0:1 1500 suntest      suntest        0      0     0      0     0      0
```

```
root@sunlicht:~ > netstat -rn
Routing Table:
Destination           Gateway           Flags  Ref   Use   Interface
-------------------- -------------------- ----- ----- ------ ---------
193.16.214.0         193.16.214.28         U        4    124  hme0
10.1.1.0             10.1.1.2              U        4      0  hme0:1
224.0.0.0            193.16.214.28         U        4      0  hme0
default              193.16.214.250        UG       0   1286
127.0.0.1            127.0.0.1             UH       0      6  lo0
```

16.7.10 snoop

snoop ist ein Kommando, womit Datenpakete am Netzwerk-Interface mitgelesen werden können. Das kann in manchen Fällen ein Mitlesen mit professionellen und teuren Analysewerkzeugen ersparen. *snoop* kann aus Sicherheitsgründen nur von root verwendet werden. Die Ethernet-Schnittstelle wird in den sogenannten *promiscuous mode* geschaltet, welcher bewirkt, dass alle Pakete am LAN an die höheren Schichten weitergeleitet werden. Normalerweise werden nur jene Pakete, welche für den lokalen Rechner bestimmt sind von der LAN-Karte zur darüberliegenden Software weitergereicht.

Bei der Verwendung von *snoop* ist zu beachten, dass bei starker Netzlast u. U. Datenpakete verloren gehen können. Das Tool ist also nicht vergleichbar mit der Zuverlässigkeit von Mitlesegeräten!

Syntax:

snoop [-a] [-v] [-c *maxcount*] [-d *device*] [-i *filename*] [-n *filename*] [-o *filename*]

- **-a** Gibt je empfangenem Paket ein Geräusch nach /dev/audio (meist nur grässliches Krachen) aus.
- **-v** Gibt Paket Header detailliert aus
- **-c *maxcount*** Beendet nach dem Empfang von *maxcount* Paketen.
- **-d *device*** Liest von der Schnittstelle *device*
- **-i *filename*** Ausgabe von Paketen, welche bei einem früheren Mitlesevorgang in die Datei *filename* gespeichert wurden.
- **-n *filename*** Verwende Datei *filename* zur Namensauflösung von IP-Adressen anstatt */etc/inet/hosts*.
- **-o *filename*** Schreibe mitgelesene Daten in die Datei *filename*. Die Daten können anschließend mit der Option '-i *filename*' angesehen werden.

16.7.11 ifconfig

/usr/sbin/ifconfig dient der Konfiguration und Anzeige von Netzwerk-Schnittstellen. Es wird während des Systemstarts vom Skript */etc/rcS.d/S30rootusr* augerufen. Es wird auch später von */etc/rc2.d/S72inetsvc* aufgerufen, um Netzwerkkonfigurationen, welche durch das Nameservice NIS bzw. NIS+ gesetzt wurden, zu aktivieren. Mit *ifconfig* können auch während des laufenden Systemes Eigenschaften wie MTU, IP-Adresse, Netzmaske, Multicast etc. eingestellt werden. Netzwerkschnittstellen können mit den Optionen *plumb* und *unplumb* geöffnet bzw. geschlossen werden.

Syntax:[31]

ifconfig -a | interface [address_family][address[dest_address]] [up] [down] [auto-revarp] [netmask-mask] [broadcastaddress] [metricn] [mtun][trailers|-trailers] [private|-private] [arp|-arp] [plumb] [unplumb]

-a damit werden die Eigenschaften aller konfigurierten Netzwerk-Schnittstellen aufgelistet

interface Name der Netzwerkschnittstelle - z.Bsp. hme0 oder le0 für Ethernet-Adapter oder lo0 für Localloop-Interface

address IP-Adresse

up | down Ist ein Interface '*down*', dann können keine Daten darüber gesandt werden

netmask *nn.nn.nn.nn* Netzmaske

broadcast *address* Broadcastadresse

mtu *nnn* Maximale Transfer Unit – im Ethernet üblicherweise 1500

plumb | unplumb Mit '*plumb*' werd die für den Datentransfer zwischen Software und Netzwerkhardware Streams und Strukturen eingerichtet. Erst dadurch ist ein Netzwerkinterface mit dem Kommando '*ifconfig -a*' sichtbar. '*unplumb*' löscht all diese Strukturen wieder und macht das Netzwerkdevice wieder '*unsichtbar*'

Beispiel:

```
root@sunlicht:~ > ifconfig -a
lo0: flags=849<UP,LOOPBACK,RUNNING,MULTICAST> mtu 8232
inet 127.0.0.1 netmask ff000000
hme0: flags=863<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST> mtu 1500
inet 193.16.214.28 netmask ffffff00 broadcast 193.16.214.255
ether 8:0:20:b1:24:a7
hme0:1: flags=863<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST> mtu 1500
inet 10.1.1.2 netmask ffffff00 broadcast 10.1.1.255
```

Status Flags	
UP	Interface ist aktiv und sendet bzw. empfängt Pakete
NOTRAILERS	Es wird kein 'Trailer' am Ende des Ethernet Rahmens angehängt. Diese Option wird unter Solaris nicht unterstützt, der Parameter existiert aber aus Gründen der Kompatibilität. Die Methode 'Trailers' setzt Header-Informationen in Berkeley UNIX-Systemen ans Ende der Pakete.
RUNNING	Das Interface wurde vom Kernel erkannt.
MULTICAST	Multicast-Adressen werden unterstützt.
BROADCAST	Broadcast-Adressen werden unterstützt.

Für jede Änderung der Schnittstelleneigenschaften muss das Interface zuvor mit dem Parameter *down* deaktiviert werden. Zur Kontrolle der Interface-Eigenschaften müssen mit *'ifconfig -a'* folgende Punkte überprüft werden:

- Sind alle Schnittstellen UP?
- Ist die IP-Adresse korrekt?
- Ist die Netzmaske korrekt?
- Passt die Broadcast-Adresse zu IP-Adresse und Netzmaske?

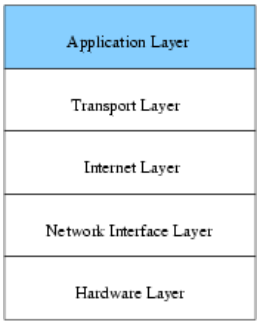
16.8 Netzwerk Management – SNMP

16.9 DHCP (Dynamic Host Configuration Protocol)

16.10 E-Mail, sendmail und SMTP

16.11 Client–Server Modell

Das Client–Server Modell ist ein wesentlicher Teil der Netzwerk-Administration. Ein *Service* ist eine Applikation, auf die über ein Netzwerk zugegriffen wird. Das Client–Server Modell beschreibt das Verhältnis zwischen *Client* (einem Prozess auf einem System, der ein *Service* in Anspruch nimmt) einerseits und *Server* (einem Prozess, der ein *Service* zur Verfügung stellt) andererseits. Dieses Verhältnis spielt sich auf der Applikationsschicht im TCP/IP Modell ab.



16.11.1 ONC+ Technologie

ONC+ (*Open System Network Computing*) ist die von SUN™entwickelte Technik, die den Programmierer bei der Schaffung von Client–Server Applikationen in heterogenen Netzwerken unterstützt. ONC+ Technologien stellen auch Werkzeuge für Administratoren zur Verfügung.

Die folgende Abbildung zeigt den modularen Aufbau einer auf ONC+ basierenden Client–Server Anwendung.

RPC Application Programm	
TI-RPC	XDR
TLI	Sockets
TCP oder UDP Port Nummern	

Die ONC+ Technologie befindet sich im ISO/OSI 7 Schichtenmodell etwa auf der Ebene der Session- und Präsentationsschicht.

TI-RPC Der *Transport-independent Remote Procedure Call* (*TI-RPC*) wurde von SUN und AT&T als Teil von UNIX System V Release 4 (SVR4) entwickelt. RPC Applikationen sind unabhängig von der darunterliegenden Transportschicht. Früher wurde das jeweils verwendete Transportsystem zum Übersetzungszeitpunkt fix eingebunden und war somit nicht mehr auf einem anderen Transportsystem einsetzbar. Wird vom Administrator ein neues Transportsystem hinzugefügt, kann die Applikation nach einen Restart sofort darauf zugreifen. Es sind keine Änderungen im Programm nötig.

XDR *External data representation* (*XDR*) ist das 'Rückgrat' von SunSoft's™*Remote Procedure Call* (*RPC)* Paket. XDR kümmert sich darum, dass Daten unabhängig von der jeweiligen Rechnerarchitektur per RPC übertragen werden können. Es ermöglicht den Datenaustausch zwischen heterogenen Systemen. XDR kümmert sich um die richtige Präsentation der Daten bezüglich 'Byte order'[32], Datentyp etc.

TLI Das *Transport Layer Interface* (*TLI*) wurde von AT&T mit UNIX System V Release 3 im Jahre 1986 eingeführt. Es stellt ein *Application Program Interface* (*API*) für die Transport Schicht zur Verfügung und steht somit zwischen der OSI Transport und Session-Schicht.

Sockets Sockets bilden das an der Universität Berkeley entwickelte Interface für Netzwerkprotokolle unter UNIX. Sie sind allgemein unter dem Namen *Berkeley Sockets* oder *BSD Sockets* bekannt.

Ein Socket ist ein Endpunkt einer Kommunikation, dem ein Name zugeordnet werden kann. Ein Socket hat einen Typ und einen zugeordneten Prozess.

16.11.1.1 Überblick der ONC+ Technologien

NFS Das *Network Filesystem* (*NFS*) ist Sun's verteiltes Dateisystem, das in einem heterogenen Netz den transparenten Zugriff auf entfernte Rechner ermöglicht. Mittels NFS können angefangen von PC's über Workstations, Mainframes bis zu Supercomputern alle Rechner eines Netzes auf dieselben Dateien zugreifen, als ob sie auf der lokalen Festplatte wären. NFS unterstützt Kerberos Authentifizierung, Multithreading, Netzwerk Lock-Manager und Automount. Genaueres ist im Kapitel 12 auf Seite 123.

NIS+ *NIS+* ist das Enterprise Name Service unter Solaris. Es unterstützt eine skalierbare und sichere Informationsdatenbank für Hostnamen, Netzwerkadressen, Benutzerkennungen etc. Es wurde entwickelt für die zentrale Administration von großen Netzwerken. Änderungen von Administrationsdaten werden automatisch an die nötigen Stellen repliziert.

16.11.2 Portnummern

Jeder Service, der im Netzwerk angeboten wird, verwendet ein *Port* worüber Client- und Server-Prozess miteinander kommunizieren. Im Allgemeinen verwendet der Client als ausgehendes Port irgendeine freie Portnummer > 1024 und kommuniziert über ein sogenanntes *well-known port* mit dem Server.

Um eine Client-Server Verbindung herstellen zu können, muss es eine Vereinbarung geben, welcher Service welcher Portnummer zugeordnet ist. Diese Portnummer muss im Netzwerk eindeutig für den jeweiligen Dienst sein.

Die Datei */etc/services* wird dazu verwendet, Portnummern zu registrieren bzw. Ports zu identifizieren. Jene Ports, die in dieser Datei aufgelistet sind, werden *wellknown ports* genannt. Die ersten 1024 Ports sind reservierte Ports, welche nur von Prozessen, deren Eigentümer 'root' ist, bedient werden können.

[32] Je nach CPU-Typ steht das höchst signifikante Bit einer Zahl links oder rechts. In unserem bekannten Dezimalsystem stehen die höherwertigen Zahlen weiter links (Zehner vor Einer). In der binären Darstellung der Prozessoren kann das (z.Bsp. Intel-Prozessoren) anders sein.

Das folgende Listing zeigt einen Auszug aus der Datei /etc/services eines Systems 'Solaris 5.8 für Intel':

```
#ident  "@(#)services   1.27     00/11/06 SMI"   /* SVr4.0 1.8   */
#
#
# Copyright (c) 1999-2000 by Sun Microsystems, Inc.
# All rights reserved.
#
# Network services, Internet style
#
tcpmux          1/tcp
echo            7/tcp
echo            7/udp
discard         9/tcp           sink null
discard         9/udp           sink null
systat          11/tcp          users
daytime         13/tcp
daytime         13/udp
netstat         15/tcp
chargen         19/tcp          ttytst source
chargen         19/udp          ttytst source
ftp-data        20/tcp
ftp             21/tcp
telnet          23/tcp
smtp            25/tcp          mail
time            37/tcp          timserver
time            37/udp          timserver
name            42/udp          nameserver
whois           43/tcp          nicname         # usually to sri-nic
domain          53/udp
domain          53/tcp
bootps          67/udp                          # BOOTP/DHCP server
bootpc          68/udp                          # BOOTP/DHCP client
hostnames       101/tcp         hostname        # usually to sri-nic
pop2            109/tcp         pop-2           # Post Office Protocol -
    V2
pop3            110/tcp                         # Post Office Protocol -
    Version 3
sunrpc          111/udp         rpcbind
sunrpc          111/tcp         rpcbind
imap            143/tcp         imap2           # Internet Mail Access
    Protocol v2
ldap            389/tcp                         # Lightweight Directory
    Access Protocol
ldap            389/udp                         # Lightweight Directory
    Access Protocol
submission      587/tcp                         # Mail Message Submission
submission      587/udp                         #    see RFC 2476
ldaps           636/tcp                         # LDAP protocol over TLS/
    SSL (was sldap)
ldaps           636/udp                         # LDAP protocol over TLS/
    SSL (was sldap)
```

```
44 #
45 # Host specific functions
46 #
47 tftp            69/udp
48 rje             77/tcp
49 finger          79/tcp
50 link            87/tcp          ttylink
51 supdup          95/tcp
52 iso-tsap        102/tcp
53 x400            103/tcp                         # ISO Mail
54 x400-snd        104/tcp
55 csnet-ns        105/tcp
56 pop-2           109/tcp                         # Post Office
57 uucp-path       117/tcp
58 nntp            119/tcp         usenet          # Network News Transfer
59 ntp             123/tcp                         # Network Time Protocol
60 ntp             123/udp                         # Network Time Protocol
61 netbios-ns      137/tcp                         # NETBIOS Name Service
62 netbios-ns      137/udp                         # NETBIOS Name Service
63 netbios-dgm     138/tcp                         # NETBIOS Datagram
       Service
64 netbios-dgm     138/udp                         # NETBIOS Datagram
       Service
65 netbios-ssn     139/tcp                         # NETBIOS Session Service
66 netbios-ssn     139/udp                         # NETBIOS Session Service
67 NeWS            144/tcp         news            # Window System
68 slp             427/tcp         slp             # Service Location
       Protocol, V2
69 slp             427/udp         slp             # Service Location
       Protocol, V2
70 mobile-ip       434/udp         mobile-ip       # Mobile-IP
71 cvc_hostd       442/tcp                         # Network Console
72 #
73 # UNIX specific services
74 #
75 # these are NOT officially assigned
76 #
77 exec            512/tcp
78 login           513/tcp
79 shell           514/tcp         cmd             # no passwords used
80 printer         515/tcp         spooler         # line printer spooler
81 courier         530/tcp         rpc             # experimental
82 uucp            540/tcp         uucpd           # uucp daemon
83 biff            512/udp         comsat
84 who             513/udp         whod
85 syslog          514/udp
86 talk            517/udp
87 route           520/udp         router routed
88 ripng           521/udp
89 klogin          543/tcp                         # Kerberos authenticated
       rlogin
90 kshell          544/tcp         cmd             # Kerberos authenticated
       remote shell
```

```
91  new-rwho        550/udp     new-who         # experimental
92  rmonitor        560/udp     rmonitord       # experimental
93  monitor         561/udp                     # experimental
94  pcserver        600/tcp                     # ECD Integrated PC board
        srvr
95  sun-dr          665/tcp                     # Remote Dynamic
        Reconfiguration
96  kerberos-adm    749/tcp                     # Kerberos V5
        Administration
97  kerberos-adm    749/udp                     # Kerberos V5
        Administration
98  kerberos        750/udp     kdc             # Kerberos key server
99  kerberos        750/tcp     kdc             # Kerberos key server
100 krb5_prop       754/tcp                     # Kerberos V5 KDC
        propogation
101 ufsd            1008/tcp    ufsd            # UFS-aware server
102 ufsd            1008/udp    ufsd
103 cvc             1495/tcp                    # Network Console
104 ingreslock      1524/tcp
105 www-ldap-gw     1760/tcp                    # HTTP to LDAP gateway
106 www-ldap-gw     1760/udp                    # HTTP to LDAP gateway
107 listen          2766/tcp                    # System V listener port
108 nfsd            2049/udp    nfs             # NFS server daemon (clts
        )
109 nfsd            2049/tcp    nfs             # NFS server daemon (cots
        )
110 eklogin         2105/tcp                    # Kerberos encrypted
        rlogin
111 lockd           4045/udp                    # NFS lock daemon/manager
112 lockd           4045/tcp
113 dtspc           6112/tcp                    # CDE subprocess control
114 fs              7100/tcp                    # Font server
```

16.11.2.1 Start eines Server-Prozesses

Jeder Service benötigt einen Prozess, der Client-Anfragen bearbeitet. Viele Server-Prozesse werden im Runlevel 2 gestartet. Zusätzliche Dienste, wie z. Bsp. *in.routed*, *in.rdisc* und *sendmail* können im Runlevel 3 gestartet werden. Diese Prozesse laufen ständig im Hintergrund. Andere Server-Prozesse werden erst bei Bedarf gestartet. Erst wenn ein Client einen Dienst benötigt, wird der entsprechende Prozess gestartet. Nach Beendigung der Client-Server Anwendung wird auch der Serverprozess wieder beendet. Das hat den Vorteil, dass Resourcen gespart werden. Realisiert wird dieses Verhalten mit Hilfe des '*Internet Superservers*', der den Namen *inetd* trägt.

Der inetd Prozess Der Server-Prozess *inetd* wird beim Systemstart im Runlevel 2 vom Runscript */etc/init.d/inetsvc* gestartet. Er läuft anstatt einer Vielzahl von möglichen Serverprozessen um eingehende Anforderungen von Clients zu registrieren. Wird eine Anforderung erkannt, wird der dafür benötigte Serverprozess angestoßen. Woher aber weiß *inetd* welcher Prozess zu starten ist? Die nötigen Informationen entnimmt *inetd* aus der Konfigurationsdatei */etc/inet/inetd.conf*. Bezüglich Logging des inetd-Prozesses siehe auch Seite 54 im Kapitel 2.

Die /etc/inet/inetd.conf Datei Die Konfigurationsdatei *inetd.conf* enthält eine Liste aller Dienste, auf die *inetd* 'hört' und deren zugehörige Prozesse. Folgendes Listing zeigt diese Datei einer SOLARIS Standardinstallation.

```
1  #
2  #ident  "@(#)inetd.conf 1.27     96/09/24 SMI"    /* SVr4.0 1.5    */
3  #
4  #
5  # Configuration file for inetd(1M).  See inetd.conf(4).
6  #
7  # To re-configure the running inetd process, edit this file, then
8  # send the inetd process a SIGHUP.
9  #
10 # Syntax for socket-based Internet services:
11 #  <service_name> <socket_type> <proto> <flags> <user> <server_pathname> <args>
12 #
13 # Syntax for TLI-based Internet services:
14 #
15 #  <service_name> tli <proto> <flags> <user> <server_pathname> <args>
16 #
17 # Ftp and telnet are standard Internet services.
18 #
19 ftp     stream  tcp     nowait  root    /usr/sbin/in.ftpd       in.ftpd
20 telnet  stream  tcp     nowait  root    /usr/sbin/in.telnetd    in.telnetd
21 #
22 # Tnamed serves the obsolete IEN-116 name server protocol.
23 #
24 name    dgram   udp     wait    root    /usr/sbin/in.tnamed     in.tnamed
25 #
26 # Shell, login, exec, comsat and talk are BSD protocols.
27 #
28 shell   stream  tcp     nowait  root    /usr/sbin/in.rshd       in.rshd
29 login   stream  tcp     nowait  root    /usr/sbin/in.rlogind    in.rlogind
30 exec    stream  tcp     nowait  root    /usr/sbin/in.rexecd     in.rexecd
31 comsat  dgram   udp     wait    root    /usr/sbin/in.comsat     in.comsat
32 talk    dgram   udp     wait    root    /usr/sbin/in.talkd      in.talkd
33 #
34 # Must run as root (to read /etc/shadow); "-n" turns off logging in utmp/wtmp.
35 #
36 uucp    stream  tcp     nowait  root    /usr/sbin/in.uucpd      in.uucpd
37 #
38 # Tftp service is provided primarily for booting.  Most sites run this
39 # only on machines acting as "boot servers."
40 #
41 #tftp   dgram   udp     wait    root    /usr/sbin/in.tftpd      in.tftpd -s /
       tftpboot
42 #
43 # Finger, systat and netstat give out user information which may be
44 # valuable to potential "system crackers."  Many sites choose to disable
45 # some or all of these services to improve security.
46 #
47 #finger stream  tcp     nowait  nobody  /usr/sbin/in.fingerd    in.fingerd
48 #systat stream  tcp     nowait  root    /usr/bin/ps             ps -ef
49 #netstat        stream  tcp     nowait  root    /usr/bin/netstat        netstat -
       f inet
50 #
51 # Time service is used for clock synchronization.
```

```
#
time      stream   tcp      nowait   root     internal
time      dgram    udp      wait     root     internal
#
# Echo, discard, daytime, and chargen are used primarily for testing.
#
echo      stream   tcp      nowait   root     internal
echo      dgram    udp      wait     root     internal
discard   stream   tcp      nowait   root     internal
discard   dgram    udp      wait     root     internal
daytime   stream   tcp      nowait   root     internal
daytime   dgram    udp      wait     root     internal
chargen   stream   tcp      nowait   root     internal
chargen   dgram    udp      wait     root     internal
#
#
# RPC services syntax:
#  <rpc_prog>/<vers> <endpoint-type> rpc/<proto> <flags> <user> \
#  <pathname> <args>
#
# <endpoint-type> can be either "tli" or "stream" or "dgram".
# For "stream" and "dgram" assume that the endpoint is a socket descriptor.
# <proto> can be either a nettype or a netid or a "*". The value is
# first treated as a nettype. If it is not a valid nettype then it is
# treated as a netid. The "*" is a short-hand way of saying all the
# transports supported by this system, ie. it equates to the "visible"
# nettype. The syntax for <proto> is:
#       *|<nettype|netid>|<nettype|netid>{[,<nettype|netid>]}
# For example:
# dummy/1       tli      rpc/circuit_v,udp       wait     root     /tmp/test_svc
    test_svc
#
# Solstice system and network administration class agent server
100232/10       tli      rpc/udp wait root /usr/sbin/sadmind     sadmind
#
# Rquotad supports UFS disk quotas for NFS clients
#
rquotad/1       tli      rpc/datagram_v  wait root /usr/lib/nfs/rquotad  rquotad
#
# The rusers service gives out user information.  Sites concerned
# with security may choose to disable it.
#
#rusersd/2-3     tli      rpc/datagram_v,circuit_v        wait root /usr/lib/netsvc
    /rusers/rpc.rusersd     rpc.rusersd
#
# The spray server is used primarily for testing.
#
sprayd/1        tli      rpc/datagram_v  wait root /usr/lib/netsvc/spray/rpc.
    sprayd      rpc.sprayd
#
# The rwall server allows others to post messages to users on this machine.
#
walld/1         tli      rpc/datagram_v  wait root /usr/lib/netsvc/rwall/rpc.
    rwalld      rpc.rwalld
#
# Rstatd is used by programs such as perfmeter.
#
```

```
105 rstatd/2-4       tli    rpc/datagram_v wait root /usr/lib/netsvc/rstat/rpc.rstatd
        rpc.rstatd
106 #
107 # The rexd server provides only minimal authentication and is often not run
108 #
109 #rexd/1           tli  rpc/tcp wait root /usr/sbin/rpc.rexd     rpc.rexd
110 #
111 # rpc.cmsd is a data base daemon which manages calendar data backed
112 # by files in /var/spool/calendar
113 #
114 #
115 # Sun ToolTalk Database Server
116 #
117 #
118 # UFS-aware service daemon
119 #
120 #ufsd/1 tli      rpc/*    wait     root     /usr/lib/fs/ufs/ufsd     ufsd -p
121 #
122 # Sun KCMS Profile Server
123 #
124 100221/1         tli      rpc/tcp wait root /usr/openwin/bin/kcms_server
        kcms_server
125 #
126 # Sun Font Server
127 #
128 fs               stream   tcp      wait nobody /usr/openwin/lib/fs.auto     fs
129 #
130 # CacheFS Daemon
131 #
132 100235/1 tli rpc/tcp wait root /usr/lib/fs/cachefs/cachefsd cachefsd
133 #
134 # Kerbd Daemon
135 #
136 kerbd/4          tli      rpc/ticlts       wait     root     /usr/sbin/kerbd  kerbd
137 #
138 # Print Protocol Adaptor - BSD listener
139 #
140 printer          stream   tcp      nowait   root     /usr/lib/print/in.lpd    in.lpd
141 dtspc stream tcp nowait root /usr/dt/bin/dtspcd /usr/dt/bin/dtspcd
142 xaudio    stream tcp    wait root /usr/openwin/bin/Xaserver Xaserver -noauth -inetd
143 100068/2-5 dgram rpc/udp wait root /usr/dt/bin/rpc.cmsd rpc.cmsd
144 100083/1 tli rpc/tcp wait root /usr/dt/bin/rpc.ttdbserverd /usr/dt/bin/rpc.
        ttdbserverd
145 # rpc.metad
146 100229/1         tli      rpc/tcp          wait     root     /usr/opt/SUNWmd/sbin/rpc.
        metad   rpc.metad
147 # rpc.metamhd
148 100230/1         tli      rpc/tcp          wait     root     /usr/opt/SUNWmd/sbin/rpc.
        metamhd         rpc.metamhd
149 pop3     stream   tcp      nowait   root     /opt/SUNWipop/lib/ipop3d ipop3d
150 imap     stream   tcp      nowait   root     /opt/SUNWimap/lib/imapd imapd
151 bootps        dgram   udp  wait    root /usr/sbin/bootpd    bootpd
```

Kommentarzeilen in der Konfigurationsdatei beginnen mit dem Zeichen '#'. Alle anderen Zeilen bestehen jeweils aus 7 Feldern mit folgender Bedeutung:

SERVICE NAME SOCKET TYPE PROTOCOL WAIT/NOWAIT[.MAX] USER[.GROUP] SERVER PROGRAM SERVER PROGRAM ARGUMENTS

Service Name Das ist der Kurzname eines Services, so wie er auch in der Datei */etc/services* stehen muss. In */etc/services* erfolgt die Zuordnung zu einem bestimmten Port.

Socket Type Das beschreibt die Programmierschnittstelle, auf die der Serverprozess aufsetzt (Stream-Socket, TLI, ...).

Protokoll Dies ist einer der möglichen Werte aus der Datei */etc/inet/protocols*.

wait/nowait Beim TCP-Protokoll wird hier immer 'nowait' eingetragen. Bei Datagrammen steht an dieser Stelle üblicherweise 'wait'. Abhängig ist das davon, ob der Serverprozess *multithreaded* oder *singlethreaded* ist.

user Die angegebene Userkennung muss ein gültiger Name aus der Datei */etc/passwd* sein. Der Eintrag bestimmt, mit welchen Rechten der Serverprozess läuft. Optional kann auch die Gruppe angegeben werden.

Server Programm Dies gibt den absoluten Pfadnamen des Dienstprogrammes an, welches *inetd* starten muss.

Arguments Das sind die Argumente für den Serverprozess, welche eventuell benötigt werden.

Einträge, deren erstes Feld aus 2 durch einen Schrägstrich getrennte Zahlen bestehen, weisen auf die Verwendung von RPC hin. In diesem Fall muss der Eintrag in der Datei */etc/rpc* zu finden sein.[33]

Wenn eine Änderung in dieser Datei vorgenommen wird, so muss *inetd* davon benachrichtigt werden. Das geschieht durch das Signal *HUP*. Signale können mittels Kommando *kill* an Prozesse geschickt werden. In diesem Fall funktioniert das folgendermaßen:

Unter Solaris 2.x oder auf praktisch allen UNIX-Systemen:

```
# ps -ef | grep inetd
# kill -HUP PID
```

PID sei die Prozessnummer von *inetd*, welche vorher ermittelt wurde.

Bei Solaris 7 oder größer:

```
# pkill -HUP inetd
```

16.11.2.2 RPC – Remote Procedure Call

Der Schwachpunkt des oben beschriebenen Client-Server Modells ist, dass jeder Dienst eine für alle Hosts eines Netzes eindeutige Portnummer zugewiesen bekommen muss. Das birgt offensichtliche Probleme in sich – welcher Hersteller kann im Vorhinein alle möglichen Dienste berücksichtigen und Konflikte ausschließen? Sun hat deshalb eine Erweiterung des Client-Server Modells entwickelt: *RPC* oder *Remote Procedure Call*. Unter der Verwendung von RPC verbindet sich der Clientprozess

[33] Der Eintrag in die Datei */etc/inet/service* ist für ein RPC-Service im Normalfall nicht nötig. Im Zusammenhang mit NIS auf den älteren Solaris Versionen 4.x treten aber Probleme auf wenn der Eintrag in */etc/inet/services* fehlt.

zunächst mit dem speziellen Serverprozess *rpcbind* (*portmap* in älteren Versionen), welcher auf dem *well known port* 111 hört. Dieser Prozess teilt dem Client die Portnummer des von ihm gewünschten Servers mit.

Auf diese Weise wird vermieden, dass alle Services im gesamten Netz eindeutige Portnummern haben müssen welche in der Datei */etc/inet/services* verwaltet werden. Es muss nur die eine fixe Portnummer '111', nämlich die des Portmapper-Prozesses *rpcbind* bekannt sein. Services und deren Portnummern auf diesem System stehen in der Datei */etc/rpc*. RPC-Services registrieren sich beim Start bei *rpcbind* und erhalten von ihm die Portnummer. Wendet sich ein RPC-Client über Port 111 an *rpcbind*, erhält er von ihm die benötigte Portnummer des gewünschten Services (sofern sich der Service registriert hat). Hat sich der gewünschte Dienst nicht bei *rpcbind* registriert, antwortet *rpcbind* mit der Fehlermeldung 'RPC: Program not registered'.

Die wohl bekannteste und am weitesten verbreitete Anwendung, welche auf *rpc* aufsetzt, ist *NFS* (mehr dazu ab Seite 123).

Folgendes Beispiel zeigt die Datei */etc/rpc* eines Systems Solaris 5.8 für Intel:

```
1  #ident   "@(#)rpc          1.12     99/07/25 SMI"    /* SVr4.0 1.2    */
2  #
3  #        rpc
4  #
5  rpcbind          100000  portmap sunrpc rpcbind
6  rstatd           100001  rstat rup perfmeter
7  rusersd          100002  rusers
8  nfs              100003  nfsprog
9  ypserv           100004  ypprog
10 mountd           100005  mount showmount
11 ypbind           100007
12 walld            100008  rwall shutdown
13 yppasswdd        100009  yppasswd
14 etherstatd       100010  etherstat
15 rquotad          100011  rquotaprog quota rquota
16 sprayd           100012  spray
17 3270_mapper      100013
18 rje_mapper       100014
19 selection_svc    100015  selnsvc
20 database_svc     100016
21 rexd             100017  rex
22 alis             100018
23 sched            100019
24 llockmgr         100020
25 nlockmgr         100021
26 x25.inr          100022
27 statmon          100023
28 status           100024
29 ypupdated        100028  ypupdate
30 keyserv          100029  keyserver
31 bootparam        100026
32 sunlink_mapper   100033
33 tfsd             100037
34 nsed             100038
35 nsemntd          100039
```

```
36 showfhd          100043  showfh
37 ioadmd           100055  rpc.ioadmd
38 NETlicense       100062
39 sunisamd         100065
40 debug_svc        100066  dbsrv
41 bugtraqd         100071
42 kerbd            100078
43 event            100101  na.event        # SunNet Manager
44 logger           100102  na.logger       # SunNet Manager
45 sync             100104  na.sync
46 hostperf         100107  na.hostperf
47 activity         100109  na.activity     # SunNet Manager
48 hostmem          100112  na.hostmem
49 sample           100113  na.sample
50 x25              100114  na.x25
51 ping             100115  na.ping
52 rpcnfs           100116  na.rpcnfs
53 hostif           100117  na.hostif
54 etherif          100118  na.etherif
55 iproutes         100120  na.iproutes
56 layers           100121  na.layers
57 snmp             100122  na.snmp snmp-cmc snmp-synoptics snmp-unisys snmp-
       utk
58 traffic          100123  na.traffic
59 nfs_acl          100227
60 sadmind          100232
61 nisd             100300  rpc.nisd
62 nispasswd        100303  rpc.nispasswdd
63 ufsd             100233  ufsd
64 pcnfsd           150001
65 amiserv          100146
66 amiaux           100147
67 ocfserv          100150
```

Auf RPC basierende Serverprozesse werde gleich wie 'herkömmliche' Serverprozesse gestartet. Einige, wie z. Bsp. *rpc.nisd, mountd* und *nfsd* werden beim Boot des Systems gestartet und laufen ständig. Andere, wie *rwalld, sprayd* und *sadmind* werden über *inetd* bei Bedarf gestartet. In der Auflistung der Datei */etc/inetd.conf* weiter oben befindet sich auf Zeile 84 beispielsweise der Eintrag für *sadmind*.

Das Kommando *rpcinfo* Informationen über RPC Dienste erhält man mit dem Kommando */usr/bin/rpcinfo*.

Syntax[34]

rpcinfo [host] [-p [host]] [-u host prog] [-b prog] [-d prog]

[host] Ohne Option bzw. nur mit Hostnamen zeigt rpcinfo Programmnummer, Version, Protokolltyp, Portnummer, Service und Eigentümer des RPC-Services am lokalen bzw. am angegebenen Rechner an.

-p [host] Listet alle registrierten RPC-Services

-u host prognum Stellt fest, ob ein bestimmter Dienst läuft

-b prognum Stellt per Broadcast fest, auf welchen Rechnern im Netz dieser Dienst läuft

-d prognum Entfernt Registrierung eines RPC-Dienstes (entspricht dem stoppen eines Dienstes)

Die folgenden Beispiele stammen von einem Server mit dem Betriebssystem 'Solaris 5.8 für Intel'.

```
   program version netid     address              service    owner
    100000    4    ticots    sdotele.rpc          rpcbind    superuser
    100000    3    ticots    sdotele.rpc          rpcbind    superuser
    100000    4    ticotsord sdotele.rpc          rpcbind    superuser
    100000    3    ticotsord sdotele.rpc          rpcbind    superuser
    100000    4    ticlts    sdotele.rpc          rpcbind    superuser
    100000    3    ticlts    sdotele.rpc          rpcbind    superuser
    100000    4    tcp       0.0.0.0.0.111        rpcbind    superuser
    100000    3    tcp       0.0.0.0.0.111        rpcbind    superuser
    100000    2    tcp       0.0.0.0.0.111        rpcbind    superuser
    100000    4    udp       0.0.0.0.0.111        rpcbind    superuser
    100000    3    udp       0.0.0.0.0.111        rpcbind    superuser
    100000    2    udp       0.0.0.0.0.111        rpcbind    superuser
    100024    1    udp       0.0.0.0.128.4        status     superuser
    100024    1    tcp       0.0.0.0.128.3        status     superuser
    100024    1    ticlts    \016\020\000\000     status     superuser
    100024    1    ticotsord \021\020\000\000     status     superuser
    100024    1    ticots    \024\020\000\000     status     superuser
    100133    1    udp       0.0.0.0.128.4        -          superuser
    100133    1    tcp       0.0.0.0.128.3        -          superuser
    100133    1    ticlts    \016\020\000\000     -          superuser
    100133    1    ticotsord \021\020\000\000     -          superuser
    100133    1    ticots    \024\020\000\000     -          superuser
    100232   10    udp       0.0.0.0.128.5        sadmind    superuser
    100011    1    udp       0.0.0.0.128.6        rquotad    superuser
    100011    1    ticlts    %\020\000\000        rquotad    superuser
    100002    2    udp       0.0.0.0.128.7        rusersd    superuser
    100002    3    udp       0.0.0.0.128.7        rusersd    superuser
    100002    2    ticlts    +\020\000\000        rusersd    superuser
    100002    3    ticlts    +\020\000\000        rusersd    superuser
    100002    2    tcp       0.0.0.0.128.4        rusersd    superuser
    100002    3    tcp       0.0.0.0.128.4        rusersd    superuser
    100002    2    ticotsord 0\020\000\000        rusersd    superuser
    100002    3    ticotsord 0\020\000\000        rusersd    superuser
    100002    2    ticots    3\020\000\000        rusersd    superuser
    100002    3    ticots    3\020\000\000        rusersd    superuser
    100012    1    udp       0.0.0.0.128.8        sprayd     superuser
    100012    1    ticlts    8\020\000\000        sprayd     superuser
    100008    1    udp       0.0.0.0.128.9        walld      superuser
    100008    1    ticlts    <\020\000\000        walld      superuser
    100001    2    udp       0.0.0.0.128.10       rstatd     superuser
    100001    3    udp       0.0.0.0.128.10       rstatd     superuser
    100001    4    udp       0.0.0.0.128.10       rstatd     superuser
    100001    2    ticlts    D\020\000\000        rstatd     superuser
    100001    3    ticlts    D\020\000\000        rstatd     superuser
    100001    4    ticlts    D\020\000\000        rstatd     superuser
```

```
47      100083    1    tcp         0.0.0.0.128.5       –          superuser
48      100221    1    tcp         0.0.0.0.128.6       –          superuser
49      100235    1    tcp         0.0.0.0.128.7       –          superuser
50      100134    1    ticotsord   O\020\000\000       –          superuser
51      100234    1    ticotsord   R\020\000\000       –          superuser
52      100146    1    ticotsord   U\020\000\000       amiserv    superuser
53      100147    1    ticotsord   X\020\000\000       amiaux     superuser
54      100150    1    ticotsord   [\020\000\000       ocfserv    superuser
55      100021    1    udp         0.0.0.0.15.205      nlockmgr   superuser
56      100021    2    udp         0.0.0.0.15.205      nlockmgr   superuser
57      100021    3    udp         0.0.0.0.15.205      nlockmgr   superuser
58      100021    4    udp         0.0.0.0.15.205      nlockmgr   superuser
59      100068    2    udp         0.0.0.0.128.11      –          superuser
60      100068    3    udp         0.0.0.0.128.11      –          superuser
61      100068    4    udp         0.0.0.0.128.11      –          superuser
62      100068    5    udp         0.0.0.0.128.11      –          superuser
63      100021    1    tcp         0.0.0.0.15.205      nlockmgr   superuser
64      100021    2    tcp         0.0.0.0.15.205      nlockmgr   superuser
65      100021    3    tcp         0.0.0.0.15.205      nlockmgr   superuser
66      100021    4    tcp         0.0.0.0.15.205      nlockmgr   superuser
67      100099    3    ticotsord   sdotele.autofs      –          superuser
68      300598    1    udp         0.0.0.0.128.13      –          superuser
69      300598    1    tcp         0.0.0.0.128.9       –          superuser
70      300598    1    ticlts      {\020\000\000       –          superuser
71      300598    1    ticotsord   ~\020\000\000       –          superuser
72      300598    1    ticots      \201\020\000\000    –          superuser
73   805306368    1    udp         0.0.0.0.128.13      –          superuser
74   805306368    1    tcp         0.0.0.0.128.9       –          superuser
75   805306368    1    ticlts      {\020\000\000       –          superuser
76   805306368    1    ticotsord   ~\020\000\000       –          superuser
77   805306368    1    ticots      \201\020\000\000    –          superuser
78      100249    1    udp         0.0.0.0.128.14      –          superuser
79      100249    1    tcp         0.0.0.0.128.10      –          superuser
80      100249    1    ticlts      \222\020\000\000    –          superuser
81      100249    1    ticotsord   \225\020\000\000    –          superuser
82      100249    1    ticots      \230\020\000\000    –          superuser
83  1289637086    5    tcp         0.0.0.0.128.78      –          superuser
84  1289637086    1    tcp         0.0.0.0.128.78      –          superuser
```

Das Kommando *netstat -a* Mit dem Kommando *netstat -a* können bestehende Verbindungen und dafür reservierte Ports festgestellt werden. In der letzten Spalte wird der Zustand der Verbindung angezeigt.

Die folgende Ausgabe des Kommandos ’netstat -a’ stammt von einem System ’Solaris 5.8 für Intel’.

```
1
2  UDP: IPv4
3      Local Address          Remote Address      State
4  ----------------------  ----------------------  -------
5          *.route                                 Idle
6          *.*                                     Unbound
7          *.sunrpc                                Idle
```

```
8         *.*                                    Unbound
9         *.32771                                Idle
10        *.*                                    Unbound
11        *.32772                                Idle
12        *.name                                 Idle
13        *.biff                                 Idle
14        *.talk                                 Idle
15        *.time                                 Idle
16        *.echo                                 Idle
17        *.discard                              Idle
18        *.daytime                              Idle
19        *.chargen                              Idle
20        *.32773                                Idle
21        *.32774                                Idle
22        *.32775                                Idle
23        *.32776                                Idle
24        *.32777                                Idle
25        *.32778                                Idle
26        *.32779                                Idle
27        *.lockd                                Idle
28        *.syslog                               Idle
29        *.*                                    Unbound
30        *.32781                                Idle
31        *.32782                                Idle
32        *.32783                                Idle
33        *.177                                  Idle
34        *.161                                  Idle
35        *.32787                                Idle
36        *.32788                                Idle
37        *.4736                                 Idle
38        *.*                                    Unbound
39        *.32793                                Idle
40        *.711                                  Idle
41        *.*                                    Unbound
42        *.*                                    Unbound
43        *.*                                    Unbound
44
45  UDP: IPv6
46     Local Address                 Remote Address                State       If
47  ------------------------------ ------------------------------ ----------- -----

48        *.time                                                   Idle
49        *.echo                                                   Idle
50        *.discard                                                Idle
51        *.daytime                                                Idle
52        *.chargen                                                Idle
53
54  TCP: IPv4
55     Local Address          Remote Address     Swind Send-Q Rwind Recv-Q
           State
56  -------------------- -------------------- ------ ------ ------ ------ -------

57        *.*                    *.*                0      0 24576      0 IDLE
```

```
58  *.sunrpc                *.*                0      0 65536        0
        LISTEN
59  *.*                     *.*                0      0 65536        0 IDLE
60  *.32771                 *.*                0      0 65536        0
        LISTEN
61  *.ftp                   *.*                0      0 65536        0
        LISTEN
62  *.telnet                *.*                0      0 65536        0
        LISTEN
63  *.shell                 *.*                0      0 65536        0
        LISTEN
64  *.shell                 *.*                0      0 65536        0
        LISTEN
65  *.login                 *.*                0      0 65536        0
        LISTEN
66  *.exec                  *.*                0      0 65536        0
        LISTEN
67  *.exec                  *.*                0      0 65536        0
        LISTEN
68  *.uucp                  *.*                0      0 65536        0
        LISTEN
69  *.finger                *.*                0      0 65536        0
        LISTEN
70  *.time                  *.*                0      0 65536        0
        LISTEN
71  *.echo                  *.*                0      0 65536        0
        LISTEN
72  *.discard               *.*                0      0 65536        0
        LISTEN
73  *.daytime               *.*                0      0 65536        0
        LISTEN
74  *.chargen               *.*                0      0 65536        0
        LISTEN
75  *.32772                 *.*                0      0 65536        0
        LISTEN
76  *.32773                 *.*                0      0 65536        0
        LISTEN
77  *.32774                 *.*                0      0 65536        0
        LISTEN
78  *.fs                    *.*                0      0 65536        0
        LISTEN
79  *.32775                 *.*                0      0 65536        0
        LISTEN
80  *.printer               *.*                0      0 65536        0
        LISTEN
81  *.dtspc                 *.*                0      0 65536        0
        LISTEN
82  *.lockd                 *.*                0      0 65536        0
        LISTEN
83  *.32777                 *.*                0      0 65536        0
        LISTEN
84  *.32778                 *.*                0      0 65536        0
        LISTEN
```

```
85  *.32779                *.*                    0      0 65536      0
        LISTEN
86  *.6000                 *.*                    0      0 65536      0
        LISTEN
87  *.*                    *.*                    0      0 65536      0 IDLE
88  *.smtp                 *.*                    0      0 65536      0
        LISTEN
89  *.smtp                 *.*                    0      0 65536      0
        LISTEN
90  *.submission           *.*                    0      0 65536      0
        LISTEN
91  *.898                  *.*                    0      0 65536      0
        LISTEN
92  *.5987                 *.*                    0      0 65536      0
        LISTEN
93  *.32789                *.*                    0      0 65536      0
        LISTEN
94  *.32846                *.*                    0      0 65536      0
        LISTEN
95  localhost.32848        localhost.32773    73620      0 73620      0
        ESTABLISHED
96  localhost.32773        localhost.32848    73620      0 73620      0
        ESTABLISHED
97  localhost.32851        localhost.32846    73620      0 73620      0
        ESTABLISHED
98  localhost.32846        localhost.32851    73620      0 73620      0
        ESTABLISHED
99  localhost.32854        localhost.32853    73620      0 73620      0
        ESTABLISHED
100 localhost.32853        localhost.32854    73620      0 73620      0
        ESTABLISHED
101 localhost.32857        localhost.32846    73620      0 73620      0
        ESTABLISHED
102 localhost.32846        localhost.32857    73620      0 73620      0
        ESTABLISHED
103 localhost.32860        localhost.32859    73620      0 73620      0
        ESTABLISHED
104 localhost.32859        localhost.32860    73620      0 73620      0
        ESTABLISHED
105 localhost.32875        localhost.32846    73620      0 73620      0
        ESTABLISHED
106 localhost.32846        localhost.32875    73620      0 73620      0
        ESTABLISHED
107 localhost.32878        localhost.32877    73620      0 73620      0
        ESTABLISHED
108 localhost.32877        localhost.32878    73620      0 73620      0
        ESTABLISHED
109     *.*                    *.*                    0      0 65536      0 IDLE
110 localhost.33307        localhost.32846    73620      0 73620      0
        ESTABLISHED
111 localhost.32846        localhost.33307    73620      0 73620      0
        ESTABLISHED
```

```
112 localhost.33310      localhost.33309      73620      0 73620      0
        ESTABLISHED
113 localhost.33309      localhost.33310      73620      0 73620      0
        ESTABLISHED
114 localhost.33313      localhost.32846      73620      0 73620      0
        ESTABLISHED
115 localhost.32846      localhost.33313      73620      0 73620      0
        ESTABLISHED
116 localhost.33316      localhost.33315      73620      0 73620      0
        ESTABLISHED
117 localhost.33315      localhost.33316      73620      0 73620      0
        ESTABLISHED
118 localhost.33319      localhost.32846      73620      0 73620      0
        ESTABLISHED
119 localhost.32846      localhost.33319      73620      0 73620      0
        ESTABLISHED
120 localhost.33322      localhost.33321      73620      0 73620      0
        ESTABLISHED
121 localhost.33321      localhost.33322      73620      0 73620      0
        ESTABLISHED
122 sdotele.33662        10.39.11.1.6000      31856      0 66608     32
        ESTABLISHED
123 sdotele.33682        10.39.11.1.ftp       32120      0 66608      0
        TIME_WAIT
124 sdotele.telnet       10.39.11.1.4505      32120      0 66608      0
        ESTABLISHED
125 sdotele.33685        sdotele.32789        73620      0 73620      0
        TIME_WAIT
126 sdotele.ftp          10.39.11.1.4506      32120      0 66608      0
        ESTABLISHED
127 sdotele.33686        10.39.11.1.4507      32120      0 66608      0
        TIME_WAIT
128 sdotele.login        10.39.11.1.1023      32120      0 66608      0
        ESTABLISHED
129       *.*                *.*                0      0 65536      0 IDLE
130
131 TCP: IPv6
132   Local Address        Remote Address      Swind Send-Q Rwind Recv-Q   State
          If
133 ------------------------------------------------------------------------

134       *.*                *.*                0      0 24576      0 IDLE
135       *.ftp              *.*                0      0 65536      0 LISTEN
136       *.telnet           *.*                0      0 65536      0 LISTEN
137       *.shell            *.*                0      0 65536      0 LISTEN
138       *.login            *.*                0      0 65536      0 LISTEN
139       *.exec             *.*                0      0 65536      0 LISTEN
140       *.finger           *.*                0      0 65536      0 LISTEN
141       *.time             *.*                0      0 65536      0 LISTEN
142       *.echo             *.*                0      0 65536      0 LISTEN
143       *.discard          *.*                0      0 65536      0 LISTEN
144       *.daytime          *.*                0      0 65536      0 LISTEN
145       *.chargen          *.*                0      0 65536      0 LISTEN
```

```
146     *.printer          *.*                0       0 65536       0 LISTEN
147     *.smtp             *.*                0       0 65536       0 LISTEN
148
149 Active UNIX domain sockets
150 Address  Type          Vnode     Conn     Local Addr       Remote Addr
151 e10b9c68 stream-ord  e0e39c20 00000000     /tmp/smc898/cmdsock
152 e10b9d88 stream-ord  e0e397a0 00000000     /tmp/.X11-unix/X0
153 e10b9ea8 stream-ord  00000000      00000000
```

16.12 Fehlersuche im Netzwerk

Gleich vorweg: Es gibt keine allgemein gültige, sicher zum Ziel führende Fehlersuchstrategie. Aber folgende Vorgangsweise hat sich in meiner persönlichen Praxis als effizient erwiesen und kann zumindest als gute Strategie für eine grobe Eingrenzung des Fehlers befolgt werden.

Annahme: Ein Programm, das über ein Netzwerk mit einem fernen Rechner kommunizieren soll, funktioniert nicht. Es sieht so aus, als ob die Verbindung zum gegenüberliegenden Rechner unterbrochen ist.

- Ist nur meine Station davon betroffen?
- Kann ich den fernen Rechner mittels 'ping'-Kommando erreichen?[35]
  - Wenn ja, ist mir die Portnummer des fernen Programmes bekannt, mit dem ich kommunizieren möchte? In diesem Fall könnte mittels '*telnet IP-Adresse port*' getestet werden, ob dieses Port 'offen' ist. Wenn das Port nicht geöffnet ist, so liegt der Fehler vermutlich auf Applikationsebene im fernen Rechner. Ist das Port offen, ist auch die Applikationsebene in meinem Rechner verdächtig.
  - Klappt *ping* nur mit der Angabe der IP-Adresse, aber nicht mit dem Rechnernamen, so liegt das Problem bei der Namensauflösung. Dann muss, je nachdem welche Art von Namensauflösung im Einsatz ist, der Eintrag in /etc/inet/hosts, NIS, NIS+ oder DNS überprüft werden.
- Kann ich andere Stationen mittels 'ping' erreichen?
  - Wenn ja, aber der ferne Rechner ist nicht erreichbar, so ist zu klären, ob der ferne Rechner im gleichen Netz hängt, oder nicht.
  - Falls der ferne Rechner in einem anderen Netz hängt: Kann 'mein' Gateway erreicht werden?
  - Ist mein Gateway-Eintrag korrekt (Kommandos 'netstat -r' und 'route')?
  - Ist meine Netzmaske korrekt (Kommando 'ifconfig')?
  - Wie weit komme ich auf der Strecke zum fernen Rechner (falls mehrere Router dazwischen liegen). Dazu kann das Kommando 'traceroute' verwendet werden.
- Schlagen alle obigen Tests fehl, so liegt's wohl an meinem Rechner bzw. Netzwerkanschluss
- Funktioniert ein anderer Rechner an meinem Anschluss (je nach Aufwand das zu testen wird man das jetzt schon oder zu einem späteren Zeitpunkt ausprobieren)?
- Ist mein Netzwerk-Interface konfiguriert? – Überprüfung mit 'ifconfig -a'
  Ist die Schnittstelle *UP* und sind alle Einstellungen korrekt (IP-Adresse, Netzmaske, Broadcast,..)

[35] Meist ist 'ping' ein gutes Werkzeug um die Verbindung zu testen. Es kann aber vorkommen, dass dies kein zuverlässiger Test ist. Und zwar kann ein Rechner so konfiguriert sein, dass er nicht auf 'echo requests' antwortet (ist aber wirklich kaum der Fall). Befindet sich der andere Rechner hinter einer Firewall, so werden 'echo requests' häufig nicht durchgelassen.

- Kann ich meine eigene IP-Adresse mit ’ping’ erreichen? – ACHTUNG! Das testet nicht die Hardware! Es muss dafür auch das ’Local-Loop Device’ korrekt eingerichtet sein (das ist aber selten das Problem). Gelingt das, kann ich davon ausgehen, dass meine Grundfunktionen für TCP/IP in Ordnung sind.
- Habe ich die Möglichkeit meinen Rechner mit einem anderen Rechner mittels ’Kreuzkabel’ direkt zu verbinden und zu testen?

Je nach Testergebnis ist dann eigene Kreativität gefragt.

Ohne Reihung nach Wichtigkeit bzw. Abhängigkeit kann die Antwort auf folgende Fragen die Fehlerursache aufdecken:

- Hat meine Netzwerkkarte mehrere Anschlüsse (AUI, BNC oderRJ45) -ist der richtige Ausgang konfiguriert?
- Wurde die richtige Eigenschaft gewählt (10 oder 100 MBit, half- oder full-duplex, auto) - stimmt das mit dem überein, was ’mein’ HUB- oder Switch-Anschluss erwartet?
- Habe ich die Karte soeben getauscht – ist in irgendeiner Netzwerkkomponente noch meine ’alte’ MAC-Adresse im ARP-Cache?
- Vor allem in der Windows-Welt: Habe ich den richtigen Treiber? (Die Treiber von UNIX-Systemen kommen meist viel besser getestet auf den Markt als die Treiber für Windows-Betriebssysteme.)
- Gibt es ein Testprogramm für meine Netzwerkkarte mit dem ich unabhängig vom Betriebssystem testen kann? Das findet man meist für PC’s mit Intel- (oder kompatibler) Architektur.
- Sind sehr viele Kollissionen auf dem Netz? Das ist häufig an eigens dafür angebrachten LED’s von Netzwerkkomponenten ersichtlich.
- Wie sieht die Netzwerk-Statistik aus (Kommando ’netstat -i’ und ’netstat -s’; auf manchen Systemen auch ’etherstat’)? Gibt es viele Fehler? Vor allem die Statistiken von intelligenten Switches oder Routern sind da aufschlussreich.
- Funktionieren andere Applikationen, die auch den Netzwerkzugriff benötigen?
- Besteht die Möglichkeit durch ’Mitlesen’ am LAN den Fehler einzugrenzen?[36]

[36] Unter SOLARIS ist dafür das Kommando ’snoop’ zu finden. Auf UNIX-Systemen findet man sehr häufig ’tcpdump’. Unter Windows kann der ’Network Monitor’ (nicht standardmäßig installiert - ich glaube im Resource-Kit vorhanden!?) verwendet werden. Linux bietet neben ’tcpdump’ und vielen anderen Programmen auch ’ethereal’. ’ethereal’ bereitet die Datenpakete in schön lesbarer Form auf und bietet viele Filtermöglichkeiten und gefällt mir deshalb besonders gut. Nicht zu vergessen sind natürlich speziell für solche Zwecke vorgesehene (und meist recht teure) Mitlesegeräte. Solche Geräte bieten über das bloße Mitlesen hinaus eine Menge ausgezeichneter Diagnosemöglichkeiten.

17 Name Service

Unter *Name Service* versteht man Client Software, welche dazu dient, Namen per Anfrage an einen *Name Server* umzuschlüsseln (z.Bsp. in Nummern). Der Name Server stellt Daten wie Hostnamen, IP-Adressen, Usernamen, Passwörter und Automount Maps zur Verfügung.

Vorteile des Name Service

- Vereinfachung der Administration
  - Administration zentral an einer Stelle
  - Konsistente Information
  - Einheitliche Darstellung der Netzwerkkonfiguration
- Änderungen werden sofort für alle Clients wirksam
- Sicherheit, dass alle Clients die neuen Daten erhalten
- Backup Server umgehen das Problem des 'Single Point of Failure'

Verfügbare Name Services

DNS *Domain Name Service* (DNS) wird in einem TCP/IP Netzwerk verwendet, um Hostnamen in IP-Adressen zu übersetzen. DNS legt auch die Domäne fest, der der Host angehört.
DNS wurde für die Verwaltung der Rechnernamen im Internet entwickelt. Durch das Konzept der Domänen (hierarchische Architektur) ist DNS in der Lage Millionen von Hostnamen zu verwalten, die über den Globus verteilt sind.

NIS *Network Information Service* (NIS) stellt Daten von Ressourcen wie Benutzerkennungen, Hostnamen, Hostadressen, Services, Automount Maps etc. jedem Rechner in einem Netz zur Verfügung. NIS ist leicht zu verwalten und wird von vielen Herstellern unterstützt.

NIS+ *Network Information Service Plus* (NIS+) stellt wie NIS auch Daten unterschiedlichster Art zur Verfügung. NIS+ ist aber keine Erweiterung zu NIS, sondern eine völlige Neuentwicklung. NIS+ bietet mehr Zugriffschutz als NIS und es können mehr Daten effektiv verwaltet werden.

FNS *Federated Name Service* (FNS) unterstützt unterschiedliche Naming Systeme in einer Umgebung. Eine einheitliche Anwenderschnittstelle wird für verschiedene Name Service Systeme angeboten. FNS ersetzt nicht Name Services, sondern setzt darauf auf.

17.1 NIS – Network Information Service

NIS ist ein verteiltes Nameservice, welches Mechanismen zur Identifikation und Lokalisierung von Netzwerk-Objekten und Ressourcen bietet. Es wurde unabhängig von DNS entwickelt und hat auch andere Ziele. Während DNS die Kommunikation im Netzwerk durch die Verwendung von Namen anstelle von IP-Adressen vereinfacht, steht bei NIS die Vereinfachung der Netzwerk-Administration im Vordergrund. NIS speichert Informationen über Worstation-Namen und Adressen, Benutzer, Netzwerke und Netzwerk-Services im *NIS namespace*.

Der erste Benutzerzugriff auf NIS Information erfolgt während des Login. Wenn der Benutzer seine Kennung und das Passwort eingibt, werden die Daten anhand der NIS maps überprüft. Wenn sich das Homeverzeichnis des Users auf einem anderen Rechner befindet und erst gemountet werden muss, so werden die dafür erforderlichen Informationen aus der NIS auto_home map geholt und das NFS-Dateisystem wird automatisch eingehängt.

17.1.1 NIS Softwarepakete

SUNWnisr	SUNWnisu	SUNWypr	SUNWypu	SUNWsprot
NIS Client		-	-	-
NIS Slave Server				-
NIS Master Server				

Nach der Standardinstallation von SOLARIS sind nicht alle Pakete vorhanden, welche für einen NIS Server erforderlich sind! Die fehlenden Packages müssen nachinstalliert werden.

17.1.2 Begriffe

17.1.2.1 NIS Domains

NIS verwendet *Domains*, um den Verwaltungsbereich zu bestimmen. Es wird keine hierarchische Struktur wie bei DNS oder NIS+ verwendet, sondern die Struktur der *NIS namespace* ist flach. Eine NIS Domain kann nicht einfach direkt mit dem Internet verbunden werden nur unter der Verwendung von NIS. Es ist aber kein Problem, NIS und DNS gemeinsam zu verwenden. Die beiden Nameservices schließen sich nicht aus. Namensauflösungen werden über die Datei */etc/nsswitch.conf* gesteuert. Darin kann definiert werden, in welcher Reihenfolge verschiedene Nameservices verwendet werden sollen.

17.1.2.2 Client-Server Konzept

NIS verwendet auch ein Client-Server Konzept. NIS-Server stellen NIS-Clients Dienste zur Verfügung. Die Hauptserver werden *master server* genannt. Um die Verfügbarkeit zu erhöhen, werden

zusätzlich *backup* bzw. *slave server* eingerichtet. Beide – Master- und Slave-server – speichern Informationen in *NIS maps* ab.

Mit Hilfe der NIS Dienste kann der Systemverwalter Verwaltungsdaten automatisch und zuverlässig auf eine Vielzahl von Server verteilen und den Clients rasch und konsistent zur Verfügung stellen.

17.1.2.3 NIS Maps

NIS Information wird in *NIS maps* gespeichert. Diese ersetzen die in UNIX üblichen Konfigurationsdateien im Verzeichnis */etc*, sowie auch andere Konfigurationsdateien. Typischerweise beinhalten diese NIS maps Daten der folgenden Konfigurationsdateien:

- aliases
- auto_home
- auto_master
- bootparams
- ethers
- group
- hosts
- locale
- netgroup
- netmasks
- networks
- protocols
- passwd
- rpc
- services
- timezone

Der NIS Administrator kann selbst zusätzlich individuelle Maps erstellen.

17.1.3 Name Service Switch Configuration File

Das *Name Service Switch Configuration File* (*/etc/nsswitch.conf*) definiert die Name Services, welche vom Betriebssystem bei der Beschaffung von Netzwerk Informationen konsultiert werden sollen.

Unter Solaris werden zusätzlich noch 3 Vorlagen (*Templates*) für diese Datei zur Verfügung gestellt:

/etc/nsswitch.files Durch Kopieren auf die Datei */etc/nsswitch.conf* wird dem Betriebssystem die Suche nach Information nur in den lokalen Konfigurationsdateien erlaubt.

/etc/nsswitch.nis Diese Vorlage enthält eine Konfiguration, welche das Betriebssystem veranlasst, Informationen der Dateien *passwd, group, automount* und *aliases* zunächst in den lokalen Dateien und anschließend in den NIS maps zu suchen. Die restlichen Daten werden direkt aus den NIS maps geholt.

/etc/nsswitch.nisplus Vorlagen, die NIS+ als primäre Informationsquelle (ausgenommen *passwd, group, automount* und *aliases*) definiert.

Standard Vorlage */etc/nsswitch.nis*:

```
1  #
2  # /etc/nsswitch.nis:
3  #
4  # An example file that could be copied over to /etc/nsswitch.conf; it
5  # uses NIS (YP) in conjunction with files.
6  #
7  # "hosts:" and "services:" in this file are used only if the
8  # /etc/netconfig file has a "-" for nametoaddr_libs of "inet" transports.
9
10 # the following two lines obviate the "+" entry in /etc/passwd and /etc/
       group.
11 passwd:     files nis
12 group:      files nis
13
14 # consult /etc "files" only if nis is down.
15 hosts:      xfn nis [NOTFOUND=return] files
16 networks:   nis [NOTFOUND=return] files
17 protocols:  nis [NOTFOUND=return] files
18 rpc:        nis [NOTFOUND=return] files
19 ethers:     nis [NOTFOUND=return] files
20 netmasks:   nis [NOTFOUND=return] files
21 bootparams: nis [NOTFOUND=return] files
22 publickey:  nis [NOTFOUND=return] files
23
24 netgroup:   nis
25
26 automount:  files nis
27 aliases:    files nis
28
29 # for efficient getservbyname() avoid nis
30 services:   files nis
31 sendmailvars:   files
```

In obiger *Name Service Switch Datei* wird NIS als die einzige Datenquelle für *netgroup* festgelegt. *passwd, groups, automount, aliases* und *services* wird zunächst in den lokalen Dateien und dann in NIS gesucht. Falls der gesuchte Eintrag für *hosts, networks, protocols, rpc, ethers, netmasks, bootparams* oder *publickey* in NIS nicht gefunden wird, so wird in den lokalen Dateien gesucht. *sendmailvars* wird nur in den lokalen Dateien gesucht.

17.1.3.1 nsswitch.conf – Mögliche Konfigurationsdaten-Quellen

Quelle	Beschreibung
files	lokale /etc Dateien
nisplus	NIS+ table
nis	NIS map
compat	unterstützt die veraltete '+' Schreibweise für *passwd* und *groups*
dns	DNS - gilt nur für *hosts*

17.1.3.2 nsswitch.conf – Mögliche Status Codes

Status Code	Beschreibung
SUCCESS	Eintrag wurde gefunden
UNAVAIL	Quelle nicht verfügbar
NOTFOUND	Quelle enthält Eintrag nicht
TRYAGAIN	Quelle ist beschäftigt –> später versuchen

17.1.3.3 nsswitch.conf – Mögliche Aktionen

Aktion	Beschreibung
continue	versuche nächste Quelle
return	Suche abbrechen

Das Dateiformat von */etc/nsswitch.conf* hat folgenden Aufbau:

DATENBASIS: QUELLE

oder

DATENBASIS: QUELLE QUELLE

oder

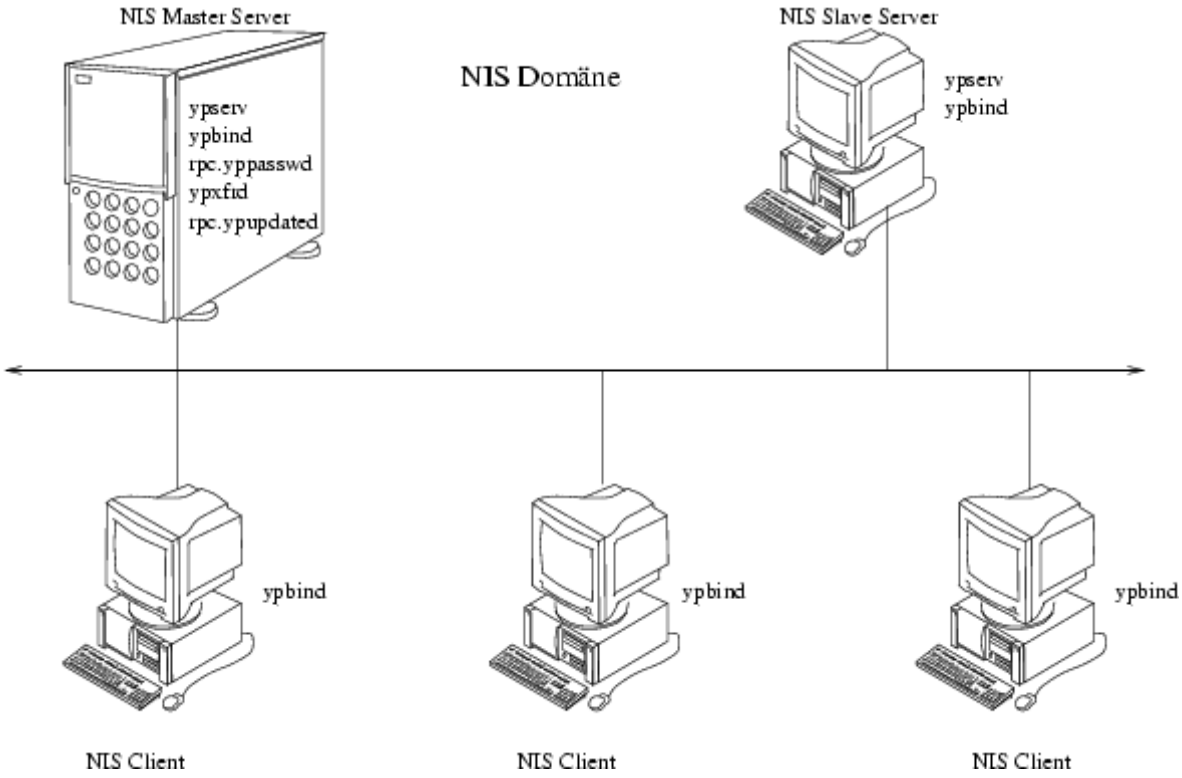
DATENBASIS: QUELLE [REGEL] QUELLE

Am Beginn der Zeile steht der Name der Datenbasis, welcher durch Doppelpunkt begrenzt wird. Danach werden die möglichen Datenquellen in der gewünschten Suchreihenfolge angegeben. Zwischen den Quellen kann noch eine Regel eingefügt werden, welche in eckige Klammern gestellt wird.

Regeln haben folgendes Format:

[STATUSCODE=ACTION]

Abbildung 17.1: NIS Domäne



17.1.4 Architektur von NIS

Ein Administrationsbereich wird in NIS *Domäne* genannt. Innerhalb einer Domäne existiert ein *NIS Master Server*, kein oder mehrere *NIS Slave Server* und ein oder mehrere *NIS Client*. Ob ein Host beim Systemstart zu einem Master Server, einem Slave Server oder zu einem Client wird, hängt einzig und alleine vom Vorhandensein bestimmter NIS-Konfigurationsdateien ab.

17.1.4.1 NIS Master Server

Eigenschaften des Master Servers:

- Besitzt die originalen ASCII Dateien im Verzeichnis */etc*, woraus die *NIS maps* erstellt werden.
- Enthält die *NIS maps*, welche aus den ASCII Dateien generiert wurden.
- Bildet den zentralen Kontrollpunkt für die Administration der gesamten NIS Domäne
- Ist einfach einzurichten.

17.1.4.2 NIS Slave Server

- Enthält keine ASCII Dateien im Verzeichnis */etc* welche für die Generierung von *NIS maps* verwendet werden.
- Besitzt Kopien der *NIS maps* des *NIS Master Server*
- Bietet Redundanz für Serverausfälle
- Ermöglicht Lastverteilung in großen Netzwerken

17.1.4.3 NIS Client

- Enthält keine ASCII Dateien woraus *NIS maps* gebildet werden
- Enthält keine *NIS maps*
- Bindet sich an Master- oder Slave Server um die nötigen Informationen aus den NIS maps zu erhalten
- Bindet sich dynamisch an einen anderen Server im Falle eines Serverfehlers
- Macht alle betroffenen Systemaufrufe per NIS

Alle Hosts in einer NIS Umgebung sind NIS Clients. Auch NIS Master Server und NIS Slave Server sind zugleich NIS Clients.

17.1.4.4 NIS Prozesse

ypserv

- Läuft auf Master Server und Slave Server
- Beantwortet *ypbind* Anfragen von Clients

ypbind

- Läuft auf allen Rechnern der NIS Domäne – Clients und Server
- Führt Bindung zu einem NIS-Server durch
- Speichert *binding* Informationen im Verzeichnis */var/yp/binding/DOMAINNAME*
- Bindet sich dynamisch auf einen anderen NIS-Server bei Verbindungsverlust
- Stellt Anfragen von NIS map Informationen an den Server

rpc.yppasswdd

- Läuft nur am Master Server
- Ermöglicht den Benutzern ihr Passwort zu ändern mittels Kommando *yppasswd* oder *passwd -r nis*.
- Aktualisiert die Dateien */etc/passwd* und */etc/shadow* am Master Server
- Aktualisiert die *NIS password map*
- Schickt die aktualisierte *NIS password map* an alle Slave Server

ypxfrd

- Läuft nur am Master Server
- Sorgt für die schnelle Übertragung der NIS maps

rpc.ypupdated

- Läuft nur am Master Server
- Dient auch der Übertragung der NIS maps

17.1.4.5 Struktur der NIS Maps

NIS Maps befinden sich im Verzeichnis */var/yp/DOMAINNAME*, wobei *DOMAINNAME* der Name der NIS Domäne ist. Für jede Map existieren 2 Dateien mit Suffix *.pag* und *.dir*.

Der Dateiname der Map-Dateien hat folgenden Aufbau:

MAP.KEY.PAG

oder

MAP.KEY.DIR

MAP Name der Map (hosts, passwd, etc.)

KEY Sortierschlüssel (byname, byaddr, etc.)

pag es handelt sich um eine Map-Datei

dir Indexdatei für die entsprechende Map-Datei (Diese Datei kann leer sein, falls die zugehörige Map-Datei klein ist).

Der Inhalt einer Mapdatei sind Paare von Schlüssel und Wert. Der Schlüssel ist der gesuchte Eintrag in der NIS-Map, der Wert wird vom NIS-Server an den NIS-Client zurückgeliefert, falls die Sucher erfolgreich war. Dieselben Daten können in unterschiedlichen Maps mehrfach vorkommen, da für jeden Sortierschlüssel eine eigene Map vorhanden sein muss. So ist z. Bsp. der Inhalt der Datei */etc/hosts* in den Map-Dateien *hosts.byname.pag, hosts.byname.dir* als auch in *hosts.byaddr.pag* und *hosts.byaddr.dir* enthalten. Im ersten Dateipaar sind die Daten nach Hostnamen, im zweiten Paar nach IP-Adressen sortiert.

17.1.5 Generierung der NIS Maps

Die Quellen für die NIS Maps befinden sich normalerweise im Verzeichnis */etc*. Aus Sicherheitsgründen wird aber vorgeschlagen, die Quellen in ein anderes Verzeichnis zu verlegen. Um diese Änderung bekanntzumachen, muss die Datei */var/yp/Makefile* editiert werden. Und zwar sind die beiden Umgebungsvariablen *DIR* und *PWDIR* auf den neuen Pfad anzupassen. Speziell die Dateien *passwd* und *shadow* stellen ein Sicherheitsrisiko dar. Diese beiden Dateien sollten deshalb im neuen Verzeichnis nur mehr jene Benutzerkennungen enthalten, die für Clients zugänglich sein sollen.

17.1.5.1 Das Kommando *make*

make ist in Tool für Programmierer um Programme zu erzeugen. Hier wird es dazu verwendet, um NIS Maps zu erstellen.

make wird durch eine *Makefile* Datei gesteuert. Diese Datei enthält üblicherweise Variablen (Makros) und Beschreibungen von Abhängigkeiten. Makros werden wie Shellvariablen verwendet: Sie werden am Beginn des Makefile mit Werten belegt und können weiterhin verwendet werden, indem das Zeichen '$' vor den Variablennamen gestellt wird.

make ist in der Lage anhand von Regeln und Abhängigkeiten, welche in der Datei *Makefile* beschrieben sind, Ziele zu erstellen. *make* übernimmt somit die Überprüfung, was ist aktueller - Quelle oder Ziel - und entscheidet, ob das Ziel neu zu erstellen ist oder nicht. Die Abhängigkeiten und Regeln, wie ein Ziel zu erstellen ist, entnimmt *make* dem *Makefile*.

Im Falle von *NIS Maps* sind die Abhängigkeiten relativ einfach. Im Folgenden sind Ausschnitte der Datei */var/yp/Makefile* beschrieben:

Der erste Teil in *Makefile* enthält Makros (ein kleiner Auszug).

```
DIR=/etc
PWDIR=/etc
DOM=`domainname`
YPDIR=/var/yp
```

Der zweite Teil beschreibt die Ziele, welche mit *make* erstellt werden sollen:

```
all:        passwd group hosts ethers networks rpc services \
            protocols netgroup bootparams aliases publickey \
            netid netmasks c2secure timezone auto.master \
            auto.home auto.direct
```

Wir wollen eine selbst erstellte NIS-Map hinzufügen (siehe **fett** gedruckter Eintrag).

Im vierten Teil (wir ziehen in der Beschreibung den 4. Teil vor, da dies dem zeitlichen Ablauf von *make* entspricht) werden die Abhängigkeiten der Maps, von welchen das Ziel *all:* wiederum abhängt, beschrieben (nur ein Auszug):

```
passwd: passwd.time
group: group.time
hosts: hosts.time
ethers: ethers.time
```

```
auto.master: auto.master.time
auto.home: auto.home.time
auto.direct: auto.direct.time
$(DIR)/netid:
$(DIR)/timezone:
$(DIR)/auto_master:
$(DIR)/auto_home:
$(PWDIR)/shadow:
$(DIR)/auto_dir:
```

Der dritte Teil enthält die Regeln, wie die Ziele erstellt werden können (im Beispiel das Regelwerk für die Erstellung einer neuen *auto_direct map*):

```
auto.direct.time: $(DIR)/auto_direct
-@if [ -f $(DIR)/auto_direct ]; then \
sed -e "/^#/d" -e s/#.*$$// $(DIR)/auto_direct \
| $(MAKEDBM) - $(YPDIR)/$(DOM)/auto.direct; \
touch auto.direct.time;
echo "updated auto.dirct"; \
if [ ! $(NOPUSH) ]; then \
$(YPPUSH) auto.direct; \
echo "pushed auto.direct"; \
else \
: ; \
fi \
else \
echo "couldn't find $(DIR)/auto_direct"; \
fi
```

Links vor dem Doppelpunkt steht das Ziel. *make* nimmt an, dies sei eine Datei und überprüft, ob das Änderungsdatum des Zieles (hier *auto.direct.time*) neuer ist, als das Änderungsdatum der Abhängigkeit (hier *$(DIR)/auto_direct*). Falls *auto.direct.time* neuer ist als *$(DIR)/auto.direct*, so muss nichts getan werden. Andernfalls aber (oder wenn das Ziel gar nicht existiert, wie es beim erstmaligen Erstellen einer NIS-Map zutrifft) müssen die Kommandos in den folgenden Zeilen ausgeführt werden, um das Ziel zu erstellen.

Die Kommandos zur Erstellung des Ziels müssen mittels Tabulator eingerückt werden! Zeilen, welche mit dem Zeichen '@' beginnen, werden nicht auf den Bildschirm ausgegeben. Bei Zeilen, welche mit dem Zeichen '-' beginnen, werden Fehlerausgaben nicht zum Bildschirm geschickt.

17.1.5.2 Erstellen einer neuen NIS-MAP:

1. Schlüssel–Wert Paar aus der gewünschten Datei extrahieren (z. Bsp. per *sed* bzw. *awk*).
2. Umwandeln der Schlüssel–Wert Paare in das bei NIS Maps verwendete Format mit *makedbm*.
3. Mit *touch* eine *Timestamp*-Datei erzeugen, damit diese Map nur bei Änderungen in der Quelle wieder neu erzeugt wird.
4. Meldung auf den Bildschirm, dass die Map neu erzeugt wurde.
5. Verteilen der neuen Map an alle Slave Server mittels *yppush*.
6. Meldung auf den Bildschirm, dass *yppush* erfolgte.

17.1.6 Konfiguration – NIS Master Server

Folgende Schritte sind nötig, um einen Master Server einzurichten:

1. Datei */etc/nsswitch.nis* nach */etc/nsswitch.conf* kopieren und ggf. anpassen.
2. Namen für die NIS Domäne wählen (üblicherweise weniger als 32 Zeichen, maximal 256 Zeichen).
3. Setzen des NIS Domänennamens:
   `# domainname gewünschter_Name`
4. In die Datei */etc/defaultdomain* Domänennamen eintragen (falls diese Datei noch nicht existiert, muss sie erzeugt werden).
5. Festlegen, welche Rechner im Netzwerk NIS Slave Server werden sollen.
6. Kontrolle des Inhaltes aller Konfigurationsdateien, welche per NIS verteilt werden sollen. Eventuell eine Kopie derselben in ein anderes Verzeichnis erstellen und die Kopien als NIS-Quellen[1] verwenden.
7. Falls nicht vorhanden, leere Dateien */etc/ethers, /etc/bootparams, /etc/locale, /etc/timezone, /etc/netgroup* und */etc/netmasks* erzeugen (z. Bsp. mit dem Kommando *touch*).
8. Datei */var/yp/Makefile* auf Korrektheit überprüfen und bei Bedarf anpassen.
9. Erstellen der neuen Master Maps aus den Quelldateien:
   `# ypinit -m`
   Dieses Programm fragt nach einer Liste von Slave Servern. Nachdem alle Namen der gewünschten Slave Server eingegeben wurden, kann die Eingabe durch *Control-d* beendet werden. Anschließend wird noch gefragt, ob *ypinit* bei Fehlern abbrechen oder fortsetzen soll. Bei der Wiederholung von *ypinit* erfolgt eine Abfrage, ob das Verzeichnis mit den Maps gelöscht werden soll (im Allgemeinen mit '*y*' beantworten).

```
1  root@sunlicht:/etc > ypinit –m
2
3  In order for NIS to operate sucessfully, we have to construct a list
       of the
4  NIS servers.  Please continue to add the names for YP servers in
       order of
5  preference, one per line.  When you are done with the list, type a <
       control D>
6  or a return on a line by itself.
7          next host to add:  sunlicht
8          next host to add:  ^D
9  The current list of yp servers looks like this:
10
11 sunlicht
12
```

[1] In diesem Fall nicht vergessen, die Variablen DIR und PWDIR im Makefile anzupassen.

```
13 Is this correct? [y/n: y] y
14
15 Installing the YP database will require that you answer a few
       questions.
16 Questions will all be asked at the beginning of the procedure.
17
18 Do you want this procedure to quit on non-fatal errors? [y/n: n]
19 OK, please remember to go back and redo manually whatever fails.  If
       you
20 don't, some part of the system (perhaps the yp itself) won't work.
21 The yp domain directory is /var/yp/sundom
22 There will be no further questions. The remainder of the procedure
       should take
23 5 to 10 minutes.
24 Building /var/yp/sundom/ypservers...
25 Running /var/yp/Makefile...
26 updated passwd
27 updated group
28 updated hosts
29 updated ethers
30 updated networks
31 updated rpc
32 updated services
33 updated protocols
34 updated netgroup
35 updated bootparams
36 WARNING: local host name (sunlicht) is not qualified; fix $j in
       config file
37 WARNING: writable directory /var
38 WARNING: writable directory /var/spool
39 WARNING: writable directory /var
40 WARNING: writable directory /var
41 WARNING: writable directory /var
42 WARNING: writable directory /var
43 WARNING: writable directory /var
44 /var/yp/sundom/mail.aliases: 3 aliases, longest 10 bytes, 52 bytes
       total
45 /usr/lib/netsvc/yp/mkalias /var/yp/`domainname`/mail.aliases /var/yp
       /`domainname`/mail.byaddr;
46 updated aliases
47 updated publickey
48 updated netid
49 /usr/sbin/makedbm /etc/netmasks /var/yp/`domainname`/netmasks.byaddr;
50 updated netmasks
51 updated timezone
52 updated auto.master
53 updated auto.home
54
55 sunlicht has been set up as a yp master server without any errors.
56
57 If there are running slave yp servers, run yppush now for any data
       bases
```

```
58 which have been changed. If there are no running slaves, run ypinit
       on
59 those hosts which are to be slave servers.
60 root@sunlicht:/etc >
```

10. Starten der NIS-Dämonen:
    `# /usr/lib/netsvc/yp/ypstart`
    Das Stoppen des NIS-Dienstes wird mit dem Kommando */usr/lib/netsvc/yp/ypstop* erreicht.

```
root@sunlicht:/etc > /usr/lib/netsvc/yp/ypstart
starting NIS (YP server) services: ypserv ypbind ypxfrd rpc.yppasswdd rpc.ypupdated done.
root@sunlicht:/etc >
```

17.1.6.1 ypinit

ypinit [-c] [-m] [-s *masterserver*]

-c NIS Clientsystem initialisieren

-m NIS Master Datenbank initialisieren

-s *masterserver* NIS Slave Server initialisieren. Es muss der NIS Masterserver der Domäne angegeben werden.

17.1.7 Zugriff und Test des NIS Dienstes

Der erste Zugriff auf den NIS Dienst erfolgt bei der Anmeldung als Benutzer.

Folgende Kommandos ermöglichen einen Test für die korrekte Funktion des NIS Dienstes:[2]

17.1.7.1 ypcat

ypcat listet alle Datensätze der Map auf.

ypcat [-k] [-d *domain*] *mapname*

ypcat -x

> ***mapname*** Name der NIS Map, deren Inhalt aufgelistet werden soll.
>
> **-k** Der Schlüssel wird zusätzlich aufgelistet
>
> **-d *domain*** Angabe der NIS Domäne, falls nicht die Standarddomäne gefragt ist
>
> **-x** Auflistung der Kurznamen aller Maps

Beispiele:

```
[root@scanner /root]# ypcat ethers
00:60:97:27:52:11 149.202.162.177
[root@scanner /root]# ypcat -k ethers
149.202.162.177 00:60:97:27:52:11 149.202.162.177
```

17.1.7.2 ypmatch

Mit *ypmatch* kann in den NIS Maps der zugehörige Wert zu einem Schlüssel ermittelt werden.

ypmatch [-k] [-d *domain*] *key* . . . *mapname*

> **key** Das gesuchte Schlüsselwort (z. Bsp. ein Hostname)
>
> ***mapname*** Name der NIS Map, deren Inhalt aufgelistet werden soll.
>
> **-k** Der Schlüssel wird zusätzlich aufgelistet
>
> **-d *domain*** Angabe der NIS Domäne, falls nicht die Standarddomäne gefragt ist
>
> **-x** Auflistung der Kurznamen aller Maps

Beispiele:

```
[root@scanner /root]# ypmatch pische passwd
pische:s3ZCRekbRVDTc:500:1001:Ernst Pisch:/home/pische:/bin/bash
[root@scanner /root]#
```

[2] Diese Kommandos stehen jedem Benutzer zur Verfügung, es sind nicht Superuser Rechte erforderlich.

17.1.7.3 ypwhich

Wird keine Option angegeben, so zeigt das Kommando *ypwhich* den NIS Server an, der dem lokalen NIS Client die Information per NIS Dienst liefert.

ypwhich [-d *domain*] [*hostname*]

ypwhich [-d *domain*] -m [*mapname*]

ypwhich -x

hostname Es wird der dem Client *hostname* zugeordnete NIS Server angezeigt

-m Zeige für jede NIS Map, welcher NIS Server Informationen liefert

mapname Anzeige des NIS-Servers, der Informationen der Map *mapname* liefert.

-d *domain* Angabe der NIS Domäne, falls nicht die Standarddomäne gefragt ist

-x Auflistung der Kurznamen aller Maps

```
1  root@sunlicht:/var/yp > ypwhich
2  sunlicht
3  root@sunlicht:/var/yp > ypwhich -x
4  Use "passwd"     for map "passwd.byname"
5  Use "group"      for map "group.byname"
6  Use "networks"   for map "networks.byaddr"
7  Use "hosts"      for map "hosts.byname"
8  Use "protocols"  for map "protocols.bynumber"
9  Use "services"   for map "services.byname"
10 Use "aliases"    for map "mail.aliases"
11 Use "ethers"     for map "ethers.byname"
12 root@sunlicht:/var/yp > ypwhich -m
13 auto.home sunlicht
14 auto.master sunlicht
15 timezone.byname sunlicht
16 netmasks.byaddr sunlicht
17 netid.byname sunlicht
18 publickey.byname sunlicht
19 mail.byaddr sunlicht
20 mail.aliases sunlicht
21 bootparams sunlicht
22 netgroup.byhost sunlicht
23 netgroup.byuser sunlicht
24 protocols.byname sunlicht
25 services.byservicename sunlicht
26 services.byname sunlicht
27 rpc.bynumber sunlicht
28 networks.byaddr sunlicht
29 networks.byname sunlicht
30 ethers.byname sunlicht
31 netgroup sunlicht
32 ethers.byaddr sunlicht
33 hosts.byaddr sunlicht
```

```
34 hosts.byname sunlicht
35 group.bygid sunlicht
36 group.byname sunlicht
37 passwd.byuid sunlicht
38 protocols.bynumber sunlicht
39 ypservers sunlicht
40 passwd.byname sunlicht
41 root@sunlicht:/var/yp >
```

17.1.8 Konfiguration – NIS Client

1. Datei */etc/nsswitch.nis* nach */etc/nsswitch.conf* kopieren und anpassen, falls erforderlich.
2. Datei */etc/hosts* kontrollieren und sicherstellen, dass der NIS Master Server und alle NIS Slave Server korrekt eingetragen sind.
3. Mittels Kommando *domainname* die NIS Domäne eintragen.
4. Den NIS Domänennamen in die Datei */etc/defaultdomain* eintragen.
5. NIS initialisieren mit dem Kommando:
   # **ypinit -c**
6. Bei der Abfrage nach einer Liste von NIS Servern, Namen des NIS Master Server und NIS Slave Server eingeben.
7. Starten des NIS Dienstes mit:
   # **/usr/lib/netsvc/yp/ypstart**
8. Testen der NIS Funktion:
   # **ypwhich -m**

17.1.9 Konfiguration – NIS Slave Server

1. Datei */etc/nsswitch.nis* nach */etc/nsswitch.conf* kopieren und anpassen, falls erforderlich.
2. Datei */etc/hosts* kontrollieren und sicherstellen, dass der NIS Master Server und alle NIS Slave Server korrekt eingetragen sind.
3. Mittels Kommando *domainname* die NIS Domäne eintragen.
4. Den NIS Domänennamen in die Datei */etc/defaultdomain* eintragen.
5. NIS zunächst als Client initialisieren mit dem Kommando:
   # **ypinit -c**
6. Bei der Abfrage nach einer Liste von NIS Servern, Namen des NIS Master Server und NIS Slave Server eingeben.
7. Kontrolle am NIS Master Server, ob der Prozess *ypserv* läuft. Falls nicht, muss zunächst das behoben werden.

8. Starten des NIS Dienstets:
   # **/usr/lib/netsvc/yp/ypstart**

9. NIS Slave Funktion aktivieren:
   # **ypinit -s *masterserver***[3]

10. NIS Dämonen stoppen:
    # **/usr/lib/netsvc/yp/ypstop**

11. NIS Dienst erneut starten (nun als Slave Server):
    # **/usr/lib/netsvc/yp/ypstart**

12. Test, ob NIS korrekt funktioniert:
    # **/ypwhich -m**
    Als Ausgabe sollte der Namen des Master Server erscheinen.

17.1.10 Aktualisierung einer NIS Map

Wenn sich Konfigurationsdaten ändern, so müssen auch die zugehörigen NIS Maps aktualisiert werden.

1. Aktualisieren der ASCII Konfigurationsdateien (falls nicht die Datei */var/yp/Makefile* verändert wurde, befinden sich diese Dateien im Verzeichnis */etc*) mit den neuen Daten.

2. Wechsel in das Verzeichnis */var/yp*.

3. Erneuern und verschicken der NIS Maps durch Aufruf von *make*:
   # **/usr/ccs/bin/make**[4]

Im obigen 'Rezept' werden neue NIS Maps vom Master Server aus an die Slave Server verschickt (*push*). Man kann auch vom Slave Server aus gezielt einzelne (oder alle) Maps holen (*pull*):

Holen der 'hosts' Maps:

```
# /usr/lib/netsvc/yp/ypxfr hosts.byaddr
# /usr/lib/netsvc/yp/ypxfr hosts.byname
```

Holen aller Maps:

```
# ypinit -s masterserver
```

Manchmal kommt es vor, dass Aktualisierungen von Maps fehlschlagen. Das Verteilen muss dann manuell mit dem Kommando *ypxfr* durchgeführt werden. Um diesen Vorgang zu automatisieren, können cron-Jobs erstellt werden. Sun stellt für diesen Zweck Vorlagen im Verzeichnis */usr/lib/netsvc/yp* zur Verfügung.

[3] *masterserver* ist der Name des NIS Master Servers im Netzwerk. Wenn am NIS Master Server beim Initialisieren von NIS dieser Slave Server nicht angegeben wurde, so kann das Kommando *ypinit -m* erneut aufgerufen werden. Im Zuge dessen wird nochmals nach den Namen der Slave Server gefragt. Die Frage, ob die aktuelle Datenbasis zerstört werden soll, mit '*y*' beantworten.

[4] NIS Maps können auch mit dem grafischen Tool *Solstice AdminSuite* aktualisiert werden.

17.1.11 Aktualisierung der Passwort Map

Wenn an einem Client das Passwort für eine Benutzerkennung geändert werden soll, so müssen NIS Master Server und NIS Slave Server davon unterrichtet werden. Damit das funktioniert, muss am Master Server der Prozess *rpc.yppasswdd* laufen:

 # /usr/lib/netsvc/yp/rpc.yppasswdd /etc/passwd -m passwd

rpc.yppasswdd sorgt dafür, dass sowohl die Datei */etc/passwd* als auch die *passwd map* aktualisiert werden. Die Passwortänderung am Client kann entweder mit den Kommandos *passwd* oder *yppasswd* durchgeführt werden.

```
pische@scanner ~> passwd
Changing NIS password for pische on sunlicht.
Old password:
New password:
Retype new password:
NIS entry changed on sunlicht
```

17.2 DNS – Domain Name Service

18 OS Server

Ein Betriebssystem Server (OS Server) hat üblicherweise folgende Ressourcen:

- Die Dateisysteme / (root) und */usr*, sowie *swap*
- */export* und */export/swap* für die Unterstützung von *Diskless Clients*. Falls der Server auch als Homeverzeichnis-Server eingesetzt wird, auch */export/home*.
- Das Verzeichnis */opt*, welches Applikationen enthält.
- OS Services für Diskless- oder Auto-Clients. Die HW-Architektur der Clients kann unterschiedlich zu der des Servers sein. Ebenso kann sich auch die OS-Version unterscheiden.
- CD-ROM Image von Solaris und Boot-Software für Clients, die über das Netzwerk booten bzw. installieren.

18.1 Network Client

18.1.1 Bootvorgang

Client	Server
sendet RARP Request, um eigene IP-Adresse zu erfahren	
	/usr/sbin/in.rarpd antwortet mit IP-Adresse des Clients
fordert mit *tftp* das Bootprogramm an	
	schickt *boot*
sendet *whoami* Request	
	schickt *hostname*
getfile Request für *boot* Parameter	
	rpc.bootparamd schickt Ort von *root* und *swap*
boot mountet per NFS das root Filesystem	
	mountd behandelt Dateianforderungen
boot startet */platform/ `uname -m `/kernel/unix*	

Wird der Client mit dem Kommando
ok **boot net -v**
gebootet, so werden zusätzliche Informationen ausgegeben, welche für die Problemeingrenzung hilfreich sind:

```
Requesting Internet address for 8:0:20:22:a0:b3
Found my IP address: ac140477 (171.20.4.119)
```

Der Name des Bootprogrammes, welches mit *tfpt* angefordert wird, hat den Namen *hex-IP-address.architecture* (z.Bsp.: `ac140477.SUN4M`).

18.1.2 Konfigurationsdateien für Network Boot

Die im Folgenden besprochenen Dateien werden automatisch mit den richtigen Daten versorgt, wenn die Einrichtung des OS Servers mit Hilfe des Host Managers von Solstice AdminSuite durchgeführt wird.

/etc/ethers

Die Datei enthält Ethernet-Adressen und Hostnamen der Clients.

/etc/inet/hosts

Die *hosts* Datei enthält die Zuordnung zwischen IP-Adresse und Hostname.

/tftpboot Verzeichnis

Im Verzeichnis */tftpboot* befinden sich die Boot-Dateien, welche für den Systemstart der Clients nötig sind.

/etc/bootparams

Diese Datei enthält Angaben über die Namen von *swap* und *root* Dateisystem.

```
client1 root=server1:/export/client1/root \
swap=server1:/export/client1/swap \
domain=bldg1.workco.com
```

/etc/timezone

/etc/timezone enthält die für den Client gültige Zeitzone und den Hostnamen des Clients.

```
US/Pacific client1
```

/etc/dfs/dfstab

Enthält für jeden Client einen NFS Share für *root* sowie *swap*.

```
share -F nfs -o rw=client1,root=client1 /export/root/client1
share -F nfs -o rw=client1,root=client1 /export/swap/client1
```

/export/root/*client1*/etc/vfstab

Im Root Verzeichnis des Network Clients wird die Datei *vfstab* angepasst, damit der Client automatisch beim Systemstart die Dateisysteme gemountet werden.

18.1.3 Konfiguration eines OS Servers

Die Konfiguration des OS Servers wird am Besten mit Hilfe von Solstice AdminSuite durchgeführt.[1]

1. Aufruf von Solstice:
   # **solstice &**
2. *Host Manager* starten
3. Auswahl des Name Services, falls ein Nameservice konfiguriert ist.
4. Auswahl des Rechners, welcher zum OS Server konfiguriert werden soll
5. Solaris 7 Software CD (im Falle von Release 5/99 von Solaris 7, Server Configuration CD) einlegen
6. Pulldown Menü: Edit –> Convert –> to OS server...
   Das Fenster mit dem Namen 'Host Manager Convert' erscheint und muss ausgefüllt werden.
7. 'Add' anklicken
8. Im Fenster 'Set Media Path' den Pfad der CD angeben und 'OK' anklicken
9. Im nun erscheinenden Fenster 'Host Manager: Add OS Services' kann die Software für verschiedene Hardwarearchitektur, sowie Ländervariante ausgewählt werden.
10. Nachdem alle benötigten Hardware Plattformen ausgewählt wurden, wird im Fenster 'Host Manager: Convert' der 'OK' Knopf angeklickt.
11. Im Fenster 'Host Manager' steht nun ganz links neben dem ausgwählten Host das Zeichen '%'.
    Auswahl des Pulldown Menüs: File –> Save Changes
    Nach dem Abschluss der Konfiguration steht in der Spalte 'Type' des Hosts 'Solaris OS Server'.

18.1.4 Hinzufügen eines Network Client

Folgende Informationen werden benötigt um einen Client ins Netz hinzuzufügen:

- Hostname des Clients
- Ethernet Adresse
- IP-Adresse
- Zeitzone
- benötigte Kernel Architektur

Die Konfiguration eines zusätzlichen Network Clients am OS Server wird mit Solstice AdminSuite durchgeführt.

[1] Sollte das Produkt nicht installiert sein, so kann es von der CD 'Solaris Easy Access Server 2.0' nachinstalliert werden.

18.1.4.1 Diskless Client

1. Aufruf von Solstice:
   `# solstice &`
2. Vor Beginn der Konfiguration muss überprüft werden, ob eine Datei */etc/ethers* bzw. per Name Service eine *ethers map* (NIS) oder *ethers table* (NIS+) verfügbar ist.
3. *Host Manager* starten.
4. Pulldown Menü 'Edit –> Add...' auswählen
5. Das nun erscheinende Fenster '*Host Manager: Add*' ausfüllen. Im Feld '*System Type*' '*Solaris Diskless*' auswählen.[2]
6. Im Host Manager Menü '*Save Changes*' auswählen. Anschließend müssen die Eintragungen für den neuen Client in den Dateien */etc/ethers*, */etc/hosts* und */etc/bootparams* zu finden sein.
7. OS Server durchstarten:
   `# init 6`
8. Network Client booten:
   `ok boot net`

18.1.4.2 Auto Client

1. Sofern verfügbar, *licensing* für Solstice AutoClient aktivieren. Details dazu in der Dokumentation '*Solstice AutoClient 2.1 Installation and Product Notes*'.[3]
2. Vor Beginn der Konfiguration muss überprüft werden, ob eine Datei */etc/ethers* bzw. per Name Service eine *ethers map* (NIS) oder *ethers table* (NIS+) verfügbar ist.
3. *Host Manager* starten.
4. Pulldown Menü 'Edit –> Add...' auswählen
5. Das nun erscheinende Fenster '*Host Manager: Add*' ausfüllen. Im Feld '*System Type*' '*Solstice Auto Client*' auswählen.
6. Die Eigenschaft '*Enable Disconnectability*' ermöglicht es dem AutoClient weiterzuarbeiten, falls der OS Server ausgefallen ist.
   Der Eintrag bei '*Disk Config*' muss überprüft werden - nicht einfach den Standardwert übernehmen!
   Durch Einträge bei '*Enable Scripts*' können zusätzliche Skripte bei den folgenden Situationen ausgeführt werden:

Pre-Add	bevor Client hinzugefügt wird
Post-Add	nachdem Client hinzugefügt wurde

[2] Die beiden Felder '*Root Path*' und '*Swap Path*' nicht verändern. Die Ethernet-Adresse kann beim Einschalten des Clients aus der *banner* Information ermittelt werden.
[3] Das ist nicht zwingende Voraussetzung!

Pre-Boot bevor der neue Client zum ersten Mal gebootet wird

Post-Boot nachdem der neue Client zum ersten Mal gebootet wurde

Pre-Modify vor dem Ändern des Host Eintrages

Post-Modify Nachdem Änderungen beim Host Eintrag durchgeführt werden

7. Wird das Root-Passwort nicht jetzt gesetzt, so wird es beim ersten Boot des Clients abgefragt.
8. Am Ende der Eintragungen den Knopf '*OK*' anklicken.[4] Der neu hinzugefügte Client erhält im Host Manager Fenster in der ersten Spalte das Zeichen '+'.
9. Pulldown Menü '*File –> Save Changes*' auswählen. Nun kann es einige Minuten dauern. Nachdem der Client erfolgreich hinzugefügt wurde, verschwindet das Zeichen '+' in der ersten Spalte.
10. Kontrolle, ob die Eintragungen in den Dateien */etc/ethers*, */etc/hosts* und */etc/bootparams* bzw. in den entsprechenden NIS Maps durchgeführt wurden.
11. AutoClient booten:
    `ok` **`boot net`**

[4] Falls *licensing* nicht aktiviert ist, erscheint eine Fehlermeldung, welche aber ignoriert werden kann.

19 JumpStart – Automatische Installation

JumpStart ist eine automatische Installation, welche unter Solaris verfügbar ist. Der Systemverwalter kann Maschinen in unterschiedliche Klassen einteilen. Die Stationen können dann automatisch mit der für die jeweilige Klasse passenden Konfiguration installiert werden. Diese Installationsart stellt eine Alternative zu *SunInstall* dar, welche im Dialog erfolgt.

19.1 Arbeitsweise von JumpStart

19.1.1 Installations-Server Typen

Die einzelnen Installationsschritte können von unterschiedlichen Servern mit Daten versorgt werden.

Configuration Server Ein Server, der die individuellen Konfigurationsdateien bereithält, welche während des Installationsvorganges benötigt werden.

Install Server Der Server, der das Solaris 7 Installationsmedium (meist CD-ROM) bereitstellt. Install-Server und Configuration-Server sind meist dieselbe Maschine.

Boot Server Ein Server, der die nötigen Dienste und Boot-Dateien bereitstellt, um einen Netzwerk-Client über das Netzwerk booten zu können bzw. eine Installation zu ermöglichen.

19.1.1.1 JumpStart Ablauf:

Client	Server
Einschalten der Maschine bzw. '`ok` **`boot net - install`**'	
Client schickt *RARP* Anforderung	
	BootServer: Prozess *in.rarpd* antwortet mit IP-Adresse
Client eröffnet *tftp*-Verbindung	
	BootServer: *in.tftp* sendet *JumpStart Bootimage*
JumpStart Bootimage wird gestartet und fordert *hostconfig* an	
	BootServer: ermittelt *hostconfig* Daten aus *bootparams* und sendet diese

Mountet root Dateisystem per NFS und startet Unix-Kernel '*/platform*/ `uname -m `/kernel/unix'	
Mountet Konfigurationsverzeichnis vom Configuration-Server und startet *sysidtool*	
Mountet Installationsverzeichnis vom Install-Server und startet *SunInstall*	

19.2 Einrichten eines JumpStart Servers

19.2.1 Konfiguration, wenn NIS vorhanden ist

Die automatische Systeminstallation erfordert folgende Informationen:

- Locale (Sprachvariante)
- Ethernet Adresse
- Host Name
- IP Adresse
- Zeitzone
- Netzmaske
- Bootparameter (optional)

Die *Locale* (Sprachvariante) wird standardmäßig nicht von NIS unterstützt und muss zuvor manuell hinzugefügt werden.

19.2.1.1 Hinzufügen einer NIS Map '*Locale*'

1. */var/yp/Makefile* modifizieren:
   a) Folgenden Eintrag nach den *.time Einträgen (ca. Zeile 305) hinzufügen
   ```
   locale.time: $(DIR)/locale
   -@if [ -f $(DIR)/locale ]; then \
   sed -e "/ ^#/d" -e s/#.*$$// $(DIR)/locale \
   | awk '{for (i = 2; i <= NF; i++) print $$I, $$0}' \
   | $(MAKEDBM) - $(YPBDIR)/$(DOM)/locale.byname; \
   touch locale.time; \
   echo "updated locale"; \
   if [ ! $(NOPUSH) then \
   $(YPPUSH) locale.byname; \
   echo "pushed locale"; \
   else \
   : ; \
   fi \
   else \
   echo "couldn't find $(DIR)/locale"; \
   fi
   ```
   b) Das Wort *locale* beim Ziel *all:* hinzufügen (ca. Zeile 47)
   c) Am Ende der Datei hinzufügen:
   ```
   locale: locale.time
   ```
2. Datei */etc/locale* erzeugen und einen Eintrag von folgendem Format machen:
   domainname locale
3. NIS Map erzeugen:
   ```
   # cd /var/yp
   # make
   ```

19.2.1.2 Dateien */etc/hosts* und */etc/ethers* anpassen:

1. MAC-Adressen und Hostnamen der Systeme, welche per JumpStart installiert werden sollen in die Datei */etc/ethers* eintragen.

2. IP-Adressen und Hostnamen der Systeme in die Datei */etc/hosts* eintragen.

3. Zu einem der Rechner in der Datei */etc/hosts* einen Aliasnamen *timehost* hinzufügen.

4. NIS Maps neu erstellen:
   ```
   # cd /var/yp
   # make
   ```

19.2.1.3 Zeitzonen (etc/timezone) und Netzmasken (/etc/netmasks) festlegen :

1. Falls nicht vorhanden, erzeugen der Datei */etc/timezone*. Der NIS-Domäne wird eine Zeitzone zugeordnet. Bsp:
   ```
   US/Mountain    Central.Sun.COM
   ```

2. Ergänzen der Datei */etc/netmasks*. Es wird die Netzadresse und die dazugehörige Netzmaske eingetragen. Bsp:
   ```
   192.168.0.0    255.255.0.0
   ```

3. NIS Maps neu erzeugen:
   ```
   # cd /var/yp
   # make
   ```

19.2.2 Kein Name Service vorhanden:

Falls im Netzwerk kein Nameservice wie NIS oder NIS+ vorhanden ist, müssen in der Datei */etc/hosts* am JumpStart Bootserver alle IP-Adressen und Hostnamen jener Rechner eingetragen werden, welche per JumpStart installiert werden sollen.

Außerdem können durch eine *sysidcfg* Datei zusätzliche Parameter übergeben werden. Existiert kein Nameservice und keine *sysidcfg* Datei, so werden die benötigten Daten während der Installation im Dialog abgefragt.

19.2.2.1 sysidcfg Datei

Diese Datei enthält Informationen, welche während der Installation von Solairs benötigt werden. Die Datei hat folgenden Aufbau:

KEYWORD ARGUMENT

Jedes Keyword-Argument Paar muss in einer einzelnen Zeile stehen! Folgende Schlüsselwörter können verwendet werden:

KEYWORD	ARGUMENT
name_service	NIS | NIS+ | OTHER | NONE {domain_name=*domainname*}
network_interface	*interface_name* {hostname=*hostname* ip_address=*ipaddress* netmask=*netmask*}
root_password	verschlüsseltes Passwort (*/etc/shadow*)
system_locale	Eintrag von */usr/lib/locale*
terminal	Eintrag von */usr/share/lib/terminfo* Datenbasis
timezone	Eintrag von */usr/share/lib/zoneinfo*
timeserver	*localhost*, IP-Adresse oder Hostname aus */etc/hosts*

Für die Datei *sysidcfg* gelten folgende Regeln:

- Schlüsselwörter können in beliebiger Reihenfolge eingetragen werden.
- Schlüsselwörter sind nicht *case sensitive* (es wird nicht zwischen Groß- und Kleinschreibung unterschieden).
- Schlüsselwörter können mit den Zeichen (’) oder (") eingeklammert werden.
- Sind dieselben Schlüsselwörter mehrmals eingetragen, so ist der erste Eintrag gültig.

Beispiel:

```
# Sample sysidcfg file for SPARC systems
system_locale=en_US
timezone=US/Central
timeserver=93.15.6.83
name_service=NONE
root_password=HDzyeüRoGQo7b
network_interface=hme0 {netmask=255.255.255.0}
```

Der Ort der Datei *sysidcfg* (Rechner und absoluter Pfadname) wird im *add_install_client* Skript festgelegt (siehe Seite 282).

19.2.3 Erstellen eines individuellen Konfigurationsverzeichnisses

Ein Konfigurationsverzeichnis enthält mindestens eine *rules* und eine *class* Datei. Nachdem diese beiden Dateien erstellt wurden, müssen sie mit dem Kommando *check* überprüft werden. Ist die Syntax OK, so erzeugt das Skript *check* eine Datei *rules.ok*. Die eigentliche Installation arbeitet mit der Datei *rules.ok* und nicht mit der Datei *rules*. Optional können noch die beiden Skripte *begin* und *finish* erstellt werden

19.2.3.1 Datei rules

Die Datei *rules* klassifiziert die Maschinen im Netz. Nach der Installation des *jumpstart_sample* Verzeichnisses von der Solaris 7 CD befindet sich eine Beispieldatei von *rules* am Rechner. Diese Datei wird sequentiell abgearbeitet. Nach jedem Einlesen einer Regel wird diese vom JumpStart Prozess abgearbeitet bevor die Datei weiter eingelesen wird.

Die Einträge haben folgendes Format:

```
[!] match_key match_value [&& [!] match_key match_value]* \
begin class finish
```

Die Felder bedeuten:

match_key vordefiniertes Schlüsselwort. Damit werden Kriterien festgelegt, womit Maschinen klassifiziert werden können.
Folgende Schlüsselwörter stehen zur Verfügung:

any	hostname	model
arch	installed	network
domainname	karch	totaldisk
disksize	memsize	

match_value Wert oder Wertebereich, der vom Systemverwalter für das jeweilige Schlüsselwort festgelegt wird.

begin Name des *begin* Skriptes. Wird kein *begin* Skript verwendet, so ist stattdessen das Zeichen '-' einzutragen.

class Name der *class* Datei.

finish Name des finish Skriptes, oder das Zeichen '-', falls nicht verwendet.

&& Mit Hilfe von '*&&*' können logische Verknüpfungen von Regeln definiert werden. *&&* steht für ein logisches *AND*

! Das Zeichen '*!*' steht für ein logisches *NOT*.

Kommentarzeilen können durch das Zeichen '*#*' am Zeilenbeginn gekennzeichnet werden. Es sind auch Leerzeilen erlaubt.

Beispiel:

```
network 192.43.34.0 && ! model 'SUNW,Sun 4_50' - class_net3 -
model 'SUNW,Sun 4_50' - class_ipx complete_ipx

hostname adm1 - class_basic_user -
memsize 16-32 && arch sparc - class_prog_user -

# This last rule matches any system that has not matched a rule above.
any - - class_generic -
```

Bedeutung der Beispieleinträge:

- Die erste Regel passt für eine Maschine im Netzwerk 192.43.34, welche keine Sun 4/50 ist. Die *class* Datei heißt *class_net3*, es existiert kein *begin* oder *finish* Skript.
- Die zweite Regel gilt für eine Sun 4/50. Deren *class* Datei heißt *class_ipx* und das *finish* Skript nennt sich *complete_ipx*.
- Die dritte Regel wird für den Rechner mit dem Hostnamen *adm1* angewandt. Die *class* Datei ist *class_basic_user*.
- Die vierte Regel passt für Rechner mit einem Hauptspeicher der Größe 16 bis 32 Megabytes und SPARC Architektur. Die *class* Datei ist *class_prog_user*.
- Die letzte Regel deckt alle jene Rechner ab, die bisher in keiner Regel berücksichtigt wurden. Die zugehörige *class* Datei heißt *class_generic*.

19.2.3.2 Class Datei

Die *class* Datei, deren Name in der Regeldatei *rules* hinterlegt wird, bestimmt wie die Installation durchgeführt wird und welche Softwarepakete installiert werden. Der Name der *class* Dateien ist frei wählbar. Es existieren auch hierfür vordefinierte Schlüsselwörter:

Keyword	Parameter
install_type	initial_install | upgrade
system_type	standalone | dataless | server
partitioning	default | existing | explicit
cluster *cluster_name*	add | delete
package *package_name*	add | delete
usedisk	*disk_name*
dontuse	*disk_name*
locale	*locale_name*
num_clients	*number*
client_swap	*size*
client_arch	*kernel_architecture*
filesys	devide *size filesystem optional_parameters*

Beispiel 1:

```
# Select software for programmers
install_type initial_install
system_type standalone
partitioning default
filesys any 60 swap # specify size of swap
filesys s-ref:/usr/share/man - /usr/share/man ro,soft
cluster SUNWCprog
package SUNWman delete
package SUNWypr add
package SUNWypu add
```

Im Beispiel wird ein System für Programmierer installiert. Es wird die Standardpartitionierung verwendet, aber die Swap-Größe wird mit 60MB festgelegt. Der Software-Cluster *SUNWCprog* enthält Entwicklersoftware. Die Online Manuale werden gelöscht, da sie vom Server *s-ref* per NFS gemountet werden. Die für NIS benötigten Pakete werden hinzugefügt.

Beispiel 2:

```
install_type initial_install
system_type standalone
partitioning explicit
filesys c0t3d0s0 150 /
filesys c0t3d0s1 128 swap
filesys c0t3d0s6 800 /usr
filesys c0t3d0s7 free /var
filesys c0t1d0s7 all /opt
cluster SUNWCall
package SUNWman delete
```

Im zweiten Beispiel werden die Partitiongrößen explizit angegeben.

Zur Erstellung eines Konfigurationsverzeichnisses sind folgende Schritte erforderlich:

1. Auswahl des Rechners und es Verzeichnisses wo die Konfigurations Informationen abgelegt werden sollen.
2. Mounten der Solaris 7 CD und Kopie des *jumpstart_sample* Verzeichnisses in das neue Konfigurationsverzeichnis.
3. Freigabe des Konfigurationsverzeichnisses per NFS:
   a) Datei */etc/dfs/dfstab* editieren.
   b) *shareall* Kommando ausführen
4. Erheben der unterschiedlichen Klassen von Maschinen im Netz und Erstellen der Datei *rules*.
5. Die in der Datei *rules* erwähnten *class* Dateien erzeugen und editieren.
6. Falls erforderlich auch die *begin* und *finish* Skripte erstellen. Das *begin* Skript läuft vor Abarbeitung der *class* Datei. Das *finish* Skript läuft am Ender der Abarbeitung der *class* Datei vor dem Reboot.
7. *check* Skript starten. Falls Fehler auftreten, diese beheben. Nach erfolgreichem Ablauf des *check* Skriptes muss eine Datei *rules.ok* vorhanden sein.
   # **cd */configuration_directory***
   # **./check**[1]
8. Überprüfen der *class* Dateien mit dem Kommando *pfinstall*. Das Kommando läuft nur dann korrekt, wenn Configuration- und Install-Server auf derselben Maschine laufen bzw. auf beiden Servern dieselbe Solaris-Version läuft.

/usr/sbin/install.d/pfinstall -D | -d *disk_file* [-c *path_to_distr*] *class_file_name*

-D überprüft die Datei *class_file_name*, und gibt Plattenaufteilung und Softwareauswahl aus. Es wird aber keine Information auf Platte geschrieben.

-d disk_file Die Plattenaufteilung wird auf Plausibilität überprüft. Dies geschieht mit Hilfe der Datei *disk_file*, welche die Ausgabe von *prtvtoc* von den Platten enthält.

-c path_to_distr Angabe des Pfades, wo sich Solaris 7 befindet.

Beispiele:

```
# /usr/sbin/install.d/pfinstall -D -c /cdrom/cdrom0/s0 prog_class
# /usr/sbin/install.d/pfinstall -d disk_file -c /cdrom/cdrom0/s0 prog_class
# /usr/sbin/install.d/pfinstall -D -c /export/install prog_class
```

[1] Falls nicht in Solaris 7 Umgebung, ist mittels Option *-p /path_to_Solaris_distribution* der Pfad anzugeben.

19.2.4 Einrichten des Install/Boot-Servers

1. Auswahl der Maschine im Netzwerk für den Install Server
2. Solaris 7 CD mounten und freigeben oder den Inhalt auf Platte kopieren (benötigt ca. 510 MB freien Platz)
   ```
   # cd /cdrom/cdrom0/s0/Solaris_2.7/Tools
   # ./setup_install_server /export/install
   ```
3. Install Clients hinzufügen mittels Kommando *add_install_client* nachdem das Konfigurationsverzeichnis fertig konfiguriert wurde.

19.2.5 Einrichten des Boot-Servers

1. Auswahl des Boot-Servers. Es werden je HW-Architektur der Clients etwa 156MB freier Plattenplatz benötigt.
2. Solaris 7 CD mounten
3. Wechsel in das Mountverzeichnis und Aufruf des *setup_install_server* Skripts
   ```
   # cd /cdrom/cdrom0/s0/Solaris_2.7/Tools
   # ./setup_install_server -b exported_dir_for_client_support
   ```
   exported_dir_for_client_support bezieht sich auf das Verzeichnis, das die Boot-Images der Clients enthält.
4. Wechsel nach *exported_dir_for_client_support*
5. Hinzufügen der Install Clients mittels Kommando *add_install_client* nachdem das Konfigurationsverzeichnis fertig konfiguriert wurde.

add_install_client -e *ethernet_address* -i *ip_address* -s *install_svr*:/*distr* -c *config_svr*:/*config_dir* -p *config_svr*:/*config_dir client_name client_arch*

- **-e** Ethernet-Adresse des Clients (kann beim Einschalten des Clients der *banner*-Information entnommen werden)
- **-i** IP-Adresse des Clients
- **-s** Name und Pfad des Install-Servers *(host:/cdrom/cdrom0/s0*)
- **-c** Angabe des Namen und Pfades des Configuration Servers
- **-p** Name Configuration Servers und Pfad zur Datei *sysidcfg*.

add_install_client übernimmt nötige Änderungen in der Datei */etc/bootparams*. Wird NIS verwendet, muss anschließend die neue NIS Map erzeugt und verteil werden mit *make* im Verzeichnis */var/yp*.

20 Tipps

20.1 Was man niemals tun sollte!

Verwenden Sie **nie** das Kommando '**rm .***' um auch versteckte Dateien in einem Verzeichnis zu löschen!!

Es passiert nicht selten, dass im Zuge einer überfüllten Partition alte Verzeichnisse gelöscht werden sollen. *rmdir* kann ein Verzeichnis nur dann löschen, wenn sich keine Dateien mehr darin befinden. Wenn zuvor in diesem Verzeichnis mit '*rm **' gelöscht wurde, bleiben aber alle 'versteckten' Dateien (also jene, die mit '.' beginnen) bestehen. An dieser Stelle wird 'gerne' ein fataler Fehler begangen: der Systemadministrator möchte alle 'Punkt-Dateien' löschen indem er das Kommando '*rm .**' aufruft.

Was passiert dann? Es werden nicht nur die Dateien im aktuellen Verzeichnis gelöscht, die mit '.' beginnen, sondern auch Dateien im übergeordneten Verzeichnis, weil auch '..' davon betroffen ist.

Wenn Sie '*' zusammen mit löschenden Kommandos verwenden, empfehle ich anstatt 'rm' zuvor **immer** 'ls' mit denselben Dateiangaben dahinter zu verwenden. Damit kann ich überprüfen, ob nicht doch noch eine Datei betroffen sein könnte, die nicht gelöscht werden soll.

Um ein gesamtes Verzeichnis zu löschen, ohne anschließend noch 'Punkt-Dateien' darin vorzufinden, kann '*rm*' mit den Optionen '*-rf*' verwendet werden. Das löscht rekursiv und ohne zu fragen das gesamte Verzeichnis. Z.Bsp.:

 # rm -rf *verzeichnisname*

20.2 Booten von CD-ROM

CD einlegen, Maschine einschalten, <STOP> drücken und im Boot PROM Kommando eingeben:

```
# boot cdrom -sw
...
# mount /dev/dsk/c0t0d0s0 /mnt
# TERM=sun; export TERM
```

20.3 Restaurieren der Systemplatte

Ist die Platte mit */root* und */usr* kaputt, so muss von CD gebootet werden, die Datensicherung zurückgespielt werden und anschließend der Bootblock restauriert werden:

1. Booten von CD
   ok **boot cdrom -s**

2. Erzeugen eines ufs-Dateisystems auf der Root-Slice
   # **newfs /dev/rdsk/c0t3d0s0**

3. Filesystemcheck starten
   # **fsck /dev/rdsk/c0t3d0s0**

4. Root Dateisystem mounten
   # **mount /dev/dsk/c0t3d0s0 /a**
   # **cd /a**

5. Sicherung von Band zurückspielen
   # **ufsrestore rvf /dev/rmt/0** # Falls eine Differenzsicherung gemacht wurde, so ist dies mit allen Sicherungen zu machen

6. Datei *restoresymtable* entfernen
   # **rm restoresymtable**

7. Root-Dateisystem abmounten
   # **cd /**
   # **umount /a**

8. Filesystemcheck starten
   # **fsck /dev/rdsk/c0t3d0s0**

9. Falls /usr auch kaputt ist, auch dafür die Punkte 2 bis 8 durchführen.

10. Bootblock wiederherstellen
    # **cd /usr/platform/‘uname -i‘/lib/fs/ufs**
    # **installboot bootblk /dev/rdsk/c0t3d0s0**

11. System rebooten
    # **init 6**

20.4 Rechnernamen ändern

1. /etc/hosts anpassen

2. /etc/hostname.hme0 - Rechnernamen eingeben (hme0 ist LAN-Karte lt. ’netstat -i’)

3. uname -S hostname —> /etc/nodename wird geändert

4. Dateien korrigieren: /etc/net/ticlts/hosts, /etc/net/ticots/hosts, /etc/net/ticotsords/hosts

Nun sollte ’uname -n’ den richtigen Namen anzeigen.

20.5 Platten-Spiegelung einrichten

1. Solstice Disk Suite installieren (siehe Seite 87)
2. Platte, welche den Spiegel bilden soll gleich partitionieren wie das Original:
   # **prtvtoc -s /dev/rdsk/c0t0d0s2 | fmthard -s - /dev/rdsk/c0t1d0s2**
3. Meta-DB erzeugen:
   # **/usr/opt/SUNWmd/sbin/metadb -a -c 3 -f /dev/dsk/c0t0d0s4**
   -a . . . add
   -c . . . copies (muss so sein!)
   -f . . . force (beim 1. Mal)
   /dev/. . . Systemplatte mit root

   Kontrolle mit Kommando '**metadb -i**'
4. Reboot
5. Metadb auf der anderen Platten auch einrichten:
   # **metadb -a -c3 /dev/dsk/c0t1d0s4** (Root-Spiegelung)
6. Spiegelung der Datenplatte einrichten:
   Datei '/etc/opt/SUNWmd/md.tab' editieren

```
# Systemplatte
d10 1 1 /dev/dsk/c0t0d0s0
d20 1 1 /dev/dsk/c0t0d0s1
d30 1 1 /dev/dsk/c0t0d0s3
d40 1 1 /dev/dsk/c0t0d0s5
d50 1 1 /dev/dsk/c0t0d0s6
d60 1 1 /dev/dsk/c0t0d0s7
# 1. Spiegelhälfte der Systemplatte („Einwege Spiegel")
d1 -m d10
d2 -m d20
d3 -m d30
d4 -m d40
d5 -m d50
d6 -m d60
# 2. Spiegelhälfte der Systemplatte
d11 1 1 /dev/dsk/c0t1d0s0
d21 1 1 /dev/dsk/c0t1d0s1
d31 1 1 /dev/dsk/c0t1d0s3
d41 1 1 /dev/dsk/c0t1d0s5
d51 1 1 /dev/dsk/c0t1d0s6
d61 1 1 /dev/dsk/c0t1d0s7
```

7. Initialisieren der 2. Spiegelhälfte
   # **cd /usr/opt/SUNWmd/sbin**
   # **for i in d11 d21 d31 d41 d51 d61**
   > **do**
   > **./metainit $i**

```
> done
```

Kontrolle des Status mit ’metastat’

8. Initialisieren der 1. Spiegelhälfte
```
# for i in d10 d20 d30 d40 d50 d60
> do
> ./metainit -f $i                          # ’-f’ ...force, da ’in use’
> done
```

9. Einwege-Spiegel initialisieren
```
# for i in d1 d2 d3 d4 d5 d6
> do
> ./metainit $i
> done
```

Kontrolle des Status mit ’metastat’

10. Anpassen von den Dateien ’/etc/system’ und ’/etc/vfstab’ für Root-Spiegelung mit Kommando
```
# ./metaroot /dev/md/dsk/d1
```

11. Restliche Partitionen müssen in der Datei ’/etc/vfstab’ händisch angepasst werden:
/dev/**md**/dsk...

12. Reboot durchführen

13. Spiegelung starten:
```
# ./metaattach d1 d11
# ./metaattach d2 d21
```
...

14. Kontrolle der Spiegelung mit Kommando
```
# ./metastat
```

15. In OBP in der Variablen **boot-device** auch die Spiegelplatte als Boot-Device angeben:
```
# init 0 (Wechsel in OBP)
ok probe-scsi
```
Namen der richtigen Platte, welche Spiegel enthält notieren/merken
```
ok show-disks
```
gewünschte Platte auswählen (Buchstaben der jeweiligen Zeile drücken)
```
ok nvalias mirror /pci@1f,4000/scsi@3/disk@1,0
```
(Der lange Name kann durch die Tastenkombination <CTRL>+<Y> eingefügt werden. Lediglich die Ziffern hinter dem letzten ’@’ müssen angefügt werden)
```
ok nvstore
ok reset
```

16. Dump-Device richtigstellen (damit bei Absturz dump sichergestellt werden kann):
Bei PANIC des Systems wird ein Speicherabbild im Swapbereich der Systemplatte zwischengespeichert. Beim nächsten Boot wird dieser ’dump’ in eine Datei kopiert was für Diagnosezwecke sehr hilfreich sein kann. Da sich der Name des Geräteknotens für den Swapbereich durch die Spiegelung geändert hat, muss auch das Dumpdevice angepasst werden. Es muss derselbe Name verwendet werden, der in der Datei ’/etc/vfstab’ als Swapdevice eingetragen ist.

 # dumpadm
Gibt aktuelles Dumpdevice aus.
 # dumpadm -d /dev/md/dsk/d*xx*
Setzt den Namen des Dumpdevices.

20.6 Vorgangsweise, wenn 1.Spiegelhälfte von Rootplatte defekt ist

Annahme (Einrichtung der Spiegel wie zuvor beschrieben):

Der Root-Spiegel ’d1’ besteht aus den beiden Hälften ’d10’ und ’d11’. Es sei ’d10’ defekt!

1. Meta-DB löschen:
   # **`metadb -d -c3 /dev/dsk/c0t0d0s4`**
2. Platte tauschen
3. Platte partitionieren mit ’**format**’
4. Meta-DB wieder einrichten
   # **`metadb -a -c3 /dev/dsk/c0t0d0s4`**

20.7 Plattentausch bei Photon (Veritas)

1. Anhand Fehlermeldung das Gerätedevice feststellen (ls -l /dev/dsk/cxtydz).
2. Mit ’**luxadm probe**’ können alle Photons aufgelistet werden.
   Steckplatz der defekten Platte feststellen mit ’**luxadm -v display *photonname***’.
3. Platte logisch entfernen mit
   # **`vxdiskadm`** —> Remove a disk for replacement
   . . . removal of disk ... completed succsessfully
4. Platte per Kommando abschalten:
   # **`luxadm -v remove_device`*`photonname`*`,r0`** # r . . . ’rear’ (Rückseite), 0 . . . Platte 0
   # f . . . ’front’ (Vorderseite)
5. Platte physikalisch entfernen und anschließend mit <CR> bestätigen
6. bevor neue Platte gesteckt wird
   # **`luxadm -v insert_device`*`photonname`*`,r0`** # r . . . ’rear’ (Rückseite), 0 . . . Platte 0
   # f . . . ’front’ (Vorderseite)
7. bei Aufforderung, Platte physikalisch stecken und mit <CR> bestätigen
8. Platte logisch ersetzen mit
   # vxdiskadm —> Replace a failed or removed disk

20.8 Modemkommandos

at Wenn Modem mit 'OK' antwortet, ist die Verbindung in Ordnung und das Modem reagiert richtig.

atz Modem rücksetzen

atx1 Das Modem wartet beim Wählen nicht auf Freizeichen.

atdt### Wählen mit Tonwahl

atdp### Wählen mit Pulswahl

ats1=0 Modem hebt bei ankommendem Ruf nicht ab

ats1=2 Modem hebt bei ankommendem Ruf nach dem 2. Läuten ab

at&v Modemkonfiguration anzeigen

at&w Modemkonfiguration abspeichern

at&f Fabrikseinstellung laden

atv1 Modem gibt Textmeldungen aus

20.9 Tools

20.9.1 Netzmasken-Kalkulator

Der 'Netzmasken-Rechner' soll den Umgang mit Subnetmasken erleichtern. Der Kalkulator ist in Perl geschrieben und setzt voraus, dass Perl und das optionale Modul Net::Netmask installiert sind. Perl gibt es für eine Vielzahl von Betriebssystemen, u.a. auch für Solaris, Linux oder Windows. Perl ist frei und kann aus dem Internet bezogen werden. Fertig übersetzte Binärversionen sind z.Bsp. zu finden auf http://www.cpan.org/ports/.

Das im Standardumfang nicht vorhandene Modul 'Net::Netmask' kann auch aus dem Internet (http://www.cpan.org/) geladen werden. Mit funktionierendem Perl geht das einfach mit folgenden Kommandos:

```
# perl -MCPAN -e shell
....
cpan> install Net::Netmask
....
cpan> q
```

Das benötigte Perl-Modul wird automatisch in den richtigen Pfad kopiert und kann dann gleich verwendet werden. Wie im folgenden Listing zu sehen ist, steckt die eigentliche Programmierarbeit schon im oben genannten Modul. Der von mir programmierte Teil ist eigentlich nur mehr die Benutzerschnittstelle.[1]

[1] Das Programm ist nicht schön geschrieben. Es geht, wie ich inzwischen gelernt habe, auch eleganter, aber es funktioniert. Zu berücksichtigen ist, dass das Programm nur wie in RFC950 'empfohlene' Netzmasken unterstützt. Nicht zusammenhängende Netzmasken führen zu Fehlern, aber diese sollten ohnehin niemals verwendet werden!

```
#!/usr/bin/perl

use Net::Netmask;

#$FS = '\t';
#$, = ' ';
$\ = "\n";


use Getopt::Long;
&GetOptions ("ip=s", "mask=s", "checkip=s", "help");

$ERRFLAG = 0;

if ( $opt_help == 1 ) {
  &usage;
   exit 0;
}

if ( !$opt_ip ) {
   print "option '-ip=aaa.bbb.ccc.ddd' is obligatory";
   $ERRFLAG = 1;
}

if ( !$opt_mask ) {
   print "option '-mask=aaa.bbb.ccc.ddd | -mask=0xaabbccdd' is obligatory"
      ;
   $ERRFLAG = 1;
}

if ( $ERRFLAG ) {
  &usage;
   exit 1;
}

$block = new Net::Netmask ($opt_ip, $opt_mask);
print "a.b.c.d/bits .... ", $block->desc();
print "netaddress ..... ", $block->base();
print "netmask ........ ", $block->mask();
print "broadcast ...... ", $block->broadcast();
print "hostmask ....... ", $block->hostmask();
#print "subnet-bits ..... ", $block->bits();
print "blocksize ...... ", $block->size();
print "hostaddresses ... ", $block->nth(1), " - ", $block->nth(-2);
print "next net ....... ", $block->next();
if ( $opt_checkip ) {
    print "\nchecked host .... ", $block->match($opt_checkip), "\nif '
       checked host' = 0 then hostip doesn't match this network";
}

sub usage {
```

```
50      print "\nusage: $0 [-help] -ip=ip-address -mask=netmask [-checkip=ip-
            address]\n";
51      print "        ip-address     aaa.bbb.ccc.ddd (decimal)";
52      print "        netmask        aaa.bbb.ccc.ddd | 0xaabbccdd (hexadecimal
            )\n";
53      print "        checkip        aaa.bbb.ccc.ddd (decimal)";
54      print "examples: $0 -ip=192.168.210.196 -mask=255.255.255.192";
55      print "           $0 -ip=192.168.210.196 -mask=0xffffffc0";
56      print "           $0 -ip=192.168.210.192 -mask=255.255.255.192 -
            checkip=192.168.210.196";
57  }
```

Abbildungsverzeichnis