Wolfgang Hurst

The Solaris Guidebook

SOLARIS™

Version 0.2.3

Wolfgang Hurst:
The Solaris Guidebook - Version 0.2.3/ Wolfgang Hurst /

Textverarbeitungssystem : *vi*
Satz : LaTeX 2_ε Computer Modern 11 Pkt. (A4 Wide)
Graphische Tools : xfig, xv

INHALTSVERZEICHNIS

I Grundlagen **1**

1 Einleitung **3**
1.1 Dieses Dokument . . . 3
1.2 Aufbau des Dokuments . . . 3
1.3 Die Beispiele . . . 3
1.4 Für wen ist das Buch gedacht ? . . . 4
1.5 Konventionen . . . 4
1.6 History . . . 4
1.7 Noch mehr Infos . . . 4

2 Sun Microsystems Overview / Processorarchitecture **5**
2.1 CISC / RISC . . . 5
2.1.1 CISC . . . 5
2.1.2 RISC . . . 5
2.2 Specialarchitecture . . . 5
2.2.1 Pipelines . . . 5
2.2.2 Branch detection . . . 6
2.3 RISC - The Second Generation . . . 7
2.3.1 Superscalar processors . . . 7
2.3.2 Superpipelining . . . 7
2.3.3 The Post-RISC Architecture . . . 7
2.3.4 Speculative Computation . . . 7
2.4 EPIC . . . 8
2.5 The SPARC Architecture . . . 8
2.5.1 SUN Bussysteme . . . 8
2.5.2 SUN Grafikkarten . . . 9
2.5.3 Die kleinen älteren SPARC Rechner . . . 9
2.5.4 Die aktuellen Workstations . . . 9
2.5.5 Mittelklasse Server und High-End . . . 10
2.6 SunOS / Solaris Releases . . . 10

3 Die Entwicklungsgeschichte von UNIX **11**

4 Begriffe in einer UNIX Umgebung **13**
4.1 Der Kernel . . . 13
4.1.1 Monolithische Kernel . . . 13
4.1.2 Modularer Kernel . . . 13
4.2 Systemcalls . . . 13
4.3 Devicefiles . . . 14
4.4 Terminals . . . 14

4.4.1 Virtuelle Terminals . . . 14
4.5 Signale . . . 14
4.5.1 Standardsignale . . . 14
4.6 Shared Memory . . . 15
4.7 Semaphoren . . . 15
4.8 Filedescriptor . . . 15
4.8.1 Standard Filedescriptoren . . . 15
4.9 Benutzer und Gruppen . . . 16
4.10 Ein Prozeß . . . 16
4.11 Shell . . . 17

5 Der Login und der Prompt 19
5.1 Das Login . . . 19
5.2 Der Prompt . . . 19
5.2.1 Telnet . . . 19

6 Einfache Befehle 21
6.1 Hilfen eines UNIX Systems . . . 21
6.1.1 man . . . 21
6.1.2 Answerbook™ . . . 21
6.2 Systemermittlungsbefehle . . . 21
6.3 Benutzerorientiertebefehle . . . 22
6.4 Prozessorientiertebefehle . . . 22
6.5 Praktikum . . . 23

7 UNIX Shells 25
7.1 Verfügbare Shells . . . 25
7.2 Die Kommandozeile . . . 25
7.3 Sonderzeichen der Bourne Shell . . . 26
7.3.1 Teilen in Argumente . . . 26
7.3.2 Auf der Flucht . . . 27
7.3.3 Metazeichen / Wildcards . . . 28
Der Stern * . . . 28
Das Fragezeichen ? . . . 29
Die eckigen Klammern [] . . . 29
7.3.4 Umlenkungen . . . 30
7.3.5 Übersicht der Sonderzeichen der Bourne Shell . . . 30

8 Grundlagen zum Dateisystem 31
8.1 Verzeichnisstruktur . . . 31
8.1.1 Der Standardbaum . . . 31
8.1.2 Zusammenfassung . . . 33
8.1.3 Absolute und Relative Dateinamen . . . 33
8.2 Dateitypen und Rechte . . . 34
8.2.1 Das Modewort . . . 34
8.2.2 Dateitypen . . . 34
8.2.3 Rechte . . . 36

8.2.4 Numerische Notierung von Rechten . . . 39
8.2.5 Standard Rechte neuer Dateien/Verzeichnisse . . . 39
8.3 Befehle zur Datei/Verzeichnis Verwaltung . . . 39
8.3.1 Befehle zur Verzeichnisverwaltung . . . 40
8.3.2 Befehle zur Dateiverwaltung . . . 40
8.3.3 Befehle zur Rechteverwaltung . . . 41
8.4 Praktikum . . . 43
8.4.1 Umgang mit der Verzeichnisstruktur . . . 43

9 Jobcontrol 45
9.1 Hintergrundprozesse . . . 45
9.2 Stoppen von Vordergrundprozessen . . . 46
9.3 Jobcontrol . . . 47
9.4 Zusammenfassung . . . 49

10 Filterprogramme 51
10.1 Pipes . . . 51
10.1.1 Einfaches Beispiel . . . 51
10.2 Einfache Filterprogramme . . . 53
10.3 Datenstrom veränderen Filterprogramme . . . 53
10.4 Datenstrom vernichtene Filterprogramme . . . 54

11 vi 55
11.1 Gewöhnungsbedürftig . . . 55
11.2 Die 3 Modies . . . 56
11.2.1 Der Kommandomodus . . . 56
Cursorbewegungen innerhalb des Textes . . . 56
Kopieren und Löschen von Texten . . . 56
11.2.2 Der ex-Modus . . . 56
Dateien Operationen . . . 56
Suchen von Textstellen . . . 56
11.2.3 Der Eingabemodus . . . 56
11.3 Einstellungsmöglichkeiten mit `:set` . . . 56
11.4 Macros mit `:map` . . . 56
11.5 *vi* Klones . . . 56
11.5.1 gvim . . . 56
11.6 Advanced Setup . . . 56

II Shellprogrammierung 57

12 Environment einer Shell 59
12.1 Aufbau einer Environmentvariablen . . . 59
12.1.1 Zuweisungen von Werten . . . 59
12.1.2 Auslesen von Variablen . . . 59
12.1.3 Gültigkeitsbereiche von Environmentvariablen . . . 61
12.2 Standard und System Variablen . . . 62

12.2.1 Anpassung der eigenen Umgebung 63

13 Funktionen 65
13.1 Struktur einer Funktion 65
13.2 Funktionsaufruf mit Argumenten 65
13.3 Funktionen die Programme verdecken 66
13.4 Verwaltung von Funktionen 67

14 Das erste Shellscript 69
14.1 Besonderheiten an einem Shellscript 69
14.2 Verwenden von Shellscripts 69
14.3 Hello World 69
14.4 Praktikum - Einfache Scripts 70

15 Kontrollstrukturen 73

16 Pipe Konstruktionen mit Controllstrukturen 75

III Lokale Administration 77

17 Installation 79
17.1 Hardware Configuration Assistant (intel) 79

18 Gerätedateien 87
18.1 Terminologie 87
18.1.1 Hardware Path 87
18.1.2 Logische Geräte 87
18.1.3 Vendor-ID 87
18.1.4 Major Nummer 88
18.1.5 Minor Nummer 88
18.1.6 Instanz 88
18.1.7 Gerätedatei 88
18.2 Gerätedateien Rekonfiguration 88
18.2.1 Das OBP 88
18.2.2 Gerätemanagment im OBP 89
18.2.3 Einstellungen zum Bootdevice 91
18.2.4 Automatischer Bootvorgang 92

19 UFS Dateiensystem Aufbau 93
19.1 Gerätenamen 93
19.1.1 Logische und physische Geräte 93
19.2 Partitionierung 94
19.3 Slices 94
19.4 Aufbau eines UFS Dateiensystems 95
19.4.1 Aufbau einer INode 96
19.4.2 Aufbau einer Verzeichnissdatei 100
19.4.3 Hardlinks 101

19.5 Erstellen von Dateiensystemen . . . 102
19.6 Prüfen von Dateiensystemen . . . 102
19.7 Filesystem Tuning und Überwachung . . . 102
19.8 Mounten von Dateiensystemen . . . 102
19.9 Zusammenfassung . . . 102
19.9.1 Tools und Programme . . . 102
19.10 Praktikum . . . 102

20 Systemstart 103
20.1 Terminologie . . . 103
20.1.1 Startmodule . . . 103
20.1.2 Runlevel . . . 104
20.1.3 Runlevel Verzeichnis . . . 106
20.1.4 Start-Stop-Scripts . . . 106
20.1.5 Runlevel Script . . . 107
Dokumentiertes Beispiel des `/sbin/rc2` Scripts . . . 108
20.2 Bootvorgang auf SPARC Rechnern . . . 111
20.3 Bootvorgang auf Intel Rechnern . . . 112
20.4 Kernel . . . 112
20.5 /sbin/init . . . 112
20.5.1 Environment von `init` . . . 112
20.5.2 Konfigurationsdatei von `init` . . . 112
20.5.3 Der Lauf in den *sysinit* Runlevel . . . 113
20.5.4 Festlegung des Default Runlevels . . . 113
20.6 Zusammenfassung . . . 113
20.6.1 Tools und Programme . . . 113
20.6.2 Konfigurationsdateien und Scripts . . . 114
20.6.3 Liste aller Standard-Startmodule . . . 114
20.7 Praktikumsaufgaben . . . 115

21 Terminalkonfiguration 119
21.1 SAF . . . 119
21.1.1 Grundlegender Aufbau von SAF . . . 119
21.2 Der SAC . . . 120
21.3 Konfiguration des `SAC` . . . 120
21.4 Die Verbindung vom Portmonitor zum Service . . . 121
21.5 Das Programm `ttymon` . . . 121
21.6 Konfiguration über `admintool` . . . 122
21.7 Terminalprogramm `tip` . . . 122
21.8 Serielle Verbindungen über das Modem . . . 123
21.8.1 Konfiguration des Dial-In Servers . . . 123
21.8.2 Konfiguration von `tip` . . . 123
21.9 Zusammenfassung . . . 125
21.9.1 Tools und Programme . . . 125
21.9.2 Konfigurationsdateien, Scripts und Devices . . . 125
21.10 Praktikumsaufgaben . . . 127

22 Benutzermanagement 129

23 Rechtemanagement 131
23.1 UNIX Standardrechte . . . 131
23.1.1 Eigentümer . . . 131
23.1.2 Gruppe . . . 131
23.1.3 Rest der Welt . . . 131
23.2 ACL . . . 131
23.2.1 Aufbau einer ACL . . . 131
23.2.2 Abfrage Reihenfolge . . . 131
23.2.3 Wichtiges für ACL . . . 133
23.2.4 Setzen von ACL's . . . 133
23.2.5 Anzeigen und Übertragung von ACL's . . . 133
23.2.6 Haupteinsatzgebiet . . . 134

24 Quota 135
24.1 Einsatz von Quotas . . . 135
24.2 Planung der Quotas . . . 135
24.3 Berechnung der Quotas . . . 135
24.4 Soft und Hard Limit . . . 136
24.5 Vorbereitung zur Benutzung von Quotas . . . 136
24.6 Editieren von Quotas . . . 136
24.7 Kopieren von Quotas . . . 137
24.8 Zeitzähler Einstellungen . . . 137
24.9 Überwachung von Quotas . . . 138
24.10 Andere Administrative Kommandos . . . 139
24.11 Zusammenfassung . . . 139
24.11.1 Befehle für Quotas . . . 139
24.11.2 Vorgehensweise . . . 139

25 Druckerkonfiguration 141
25.1 Allgemeine Funktionsweise . . . 141
25.1.1 Druckerarten . . . 142
25.1.2 Drucker Befehlsformate . . . 143
25.1.3 Kontrolle behalten . . . 143
25.1.4 Druckerklassen . . . 144
25.2 Konfiguration eines lokal/logischen Druckers . . . 144
25.2.1 Erste Vorbereitungen . . . 144
25.2.2 Einrichten eines Druckers . . . 145
25.2.3 Freischalten des Druckers . . . 145
25.2.4 Erster Druckversuch . . . 146
25.3 Konfiguration eines nicht Postscriptdruckers . . . 146

26 Backup und Archivierung 153
26.1 Backupstrategien . . . 153
26.2 Backupmedien . . . 153
26.3 Archivierungsprogramme . . . 153

26.3.1 cp . . . 153
26.3.2 tar . . . 153
26.3.3 cpio . . . 153
26.3.4 pax . . . 153
26.3.5 ufsdump . . . 153
26.3.6 dd . . . 153
26.4 Kompression . . . 153
26.4.1 pack/unpack . . . 153
26.4.2 compress/uncompress . . . 153
26.4.3 GNU Zip . . . 153
26.4.4 BZip2 . . . 153

27 Zeitgesteuert Aktionen ausführen 155
27.1 Der Cron Mechanismus . . . 155
27.1.1 Einstellen von Aktionen . . . 155
27.1.2 Format der crontab . . . 156
27.1.3 Die Ausgabe der Prozesse . . . 156
27.1.4 Berechtigungen . . . 156
27.2 Der At Mechanismus . . . 157
27.3 Zusammenfassung . . . 157

28 Softwaremanagement 159
28.1 Allgemeine Informationen . . . 159
28.2 Softwarepaketnamen . . . 159
28.3 Informationen über installierte Software . . . 159
28.4 Prüfen von Paketen . . . 161
28.5 Deinstallation von Software . . . 161
28.6 Software hinzufügen . . . 163
28.7 Patches hinzufügen . . . 163
28.8 Patches entfernen . . . 163
28.9 Praktikum . . . 165
28.9.1 Paketmanagement . . . 165
28.10 Lösungsansätze . . . 167

IV Netzwerkdienste 169

29 Konfiguration der Netzwerkumgebung 171
29.1 Manuelle Konfiguration der NIC . . . 171
29.1.1 Konfiguration der IP Addresse . . . 171
29.1.2 Konfiguration der Netzmaske . . . 172
29.1.3 Konfiguration der Broadcastaddresse . . . 172
29.1.4 Aktivierung der Konfiguration . . . 173
29.2 Manuelle Routingeinträge . . . 173
29.2.1 Temporäres setzen von Routen . . . 173
29.2.2 Anzeigen von Routingeinträgen . . . 174
29.2.3 Löschen von Routen . . . 174

30 Netzwerkstandard Dienste 177
30.1 Funktionsweise . . . 177
30.2 Praktikum . . . 179

31 Druckumgebung im Netzwerk 181
31.1 Funktionsweise . . . 181
31.2 Konfiguration des Druckservers . . . 181
31.3 Konfiguration des Druckclients . . . 181
31.3.1 Konfiguration mit dem `admintool` . . . 181

32 R-Tools 183

33 Remote Procedure Call 185

34 Network File System 187
34.1 Allgemein . . . 187
34.2 Terminologie . . . 187
34.2.1 Versionen . . . 187
34.2.2 NFS-Server . . . 187
34.2.3 Share . . . 188
34.2.4 NFS-Client . . . 188
34.2.5 Operationen . . . 188
34.2.6 Operation Cache Management . . . 190
34.2.7 Fileaccesshandle . . . 190
34.3 Konfiguration des NFS-Servers . . . 191
34.3.1 Einrichten von Shares . . . 191
34.3.2 Hinzufügen von Shares . . . 191
34.3.3 Anzeige von Shares . . . 192
34.3.4 Entfernen eines Shares . . . 192
34.3.5 Übersicht der gemounteten Shares . . . 192
34.3.6 Und nun alles auf einmal . . . 193
34.4 Konfiguration des NFS-Client . . . 193
34.4.1 Mounten von Shares . . . 193
34.5 Rootzugriffe über NFS . . . 197
34.6 User-ID Puzzle . . . 197
34.7 Zusammenfassung . . . 199

35 Network Information Service 205
35.1 Funktionsweise . . . 205
35.2 Konfiguration des Servers . . . 205
35.3 Konfiguration des Clients . . . 205

36 Network Information Service Plus 207
36.1 Konfiguration des Servers . . . 207

37 Automounter **217**

38 Cache File System **223**
38.1 Allgemein . . . 223
38.1.1 Back und Front . . . 223
38.1.2 Funktionsweise im Detail . . . 223
38.2 Einrichten eines CacheFS . . . 224
38.3 Benutzen eines CacheFS . . . 224

39 Network Time Protocol **231**

40 Autoinstallationssystem **237**
40.1 Funktionsweise . . . 237
40.1.1 Regelwerke . . . 237
40.2 Konfiguration des Boot-Servers . . . 237
40.3 Konfiguration des Install-Servers . . . 237
40.4 Konfiguration des Regelwerk-Servers . . . 237
40.5 Konfiguration des cfg-Servers . . . 237

V Anhänge und Referenzen **243**

41 Kommandoreferenz **245**

42 Konfigurationsdateien Referenz **413**

VI Advanced Topics **437**

43 Advanced Printer Interfaces **439**
43.1 Vorwort . . . 439
43.1.1 Benötigte Software . . . 439
43.1.2 Planung im groben . . . 439
43.2 Konfiguration des Druckers . . . 439
43.2.1 Portkonfiguration . . . 439
43.2.2 Dienstkonfiguration . . . 440
43.2.3 Einrichtung mit `lpadmin` . . . 440
43.2.4 Vorläufige Interfacekonfiguration . . . 440
43.3 Die Argumente an ein Interfaceprogramm . . . 441
43.4 Das Erstellen des Druckjobs Tempverzeichnisses . . . 442
43.5 Fehlerkennung . . . 442
43.6 Dateischleife . . . 443
43.6.1 Dateitypanalyse . . . 443
ASCII Texte Konvertieren . . . 443
Konvertierung von PDF Dateien . . . 444
Konvertierung von Postscript . . . 445
Unbekannte Dateitypen . . . 445
43.6.2 Ende der Dateianalyse . . . 446

43.6.3 Postscript konvertieren fuer den Drucker 446
43.6.4 N-mal drucken 447
43.6.5 Der Rest 447
43.7 Das komplette Interface 448

TEIL I

GRUNDLAGEN

- Einleitung
- Die Entwicklungsgeschichte von UNIX
- Begriffe in einer UNIX Umgebung
- Der Login und der Prompt
- Einfache Befehle
- UNIX Shells
- Grundlagen zum Dateisystem
- Jobcontrol
- Filterprogramme
- vi

Kapitel 1

EINLEITUNG

1.1 Dieses Dokument

Dieses Dokument behandelt die Grundlagen eines UNIX Systems. Wobei hier, sofern es mir möglich ist, Solaris, SuSE Linux und RedHat Linux behandelt werden soll.

1.2 Aufbau des Dokuments

Das Dokument ist so aufgebaut das die Kommandos die verwedetet werden hinten im Anhang komplett aufgelistet werden. Es hat sich im alten Dokument gezeigt das eine Vermischung zwischen Information und Programmerklärung nicht gut ist. Daher ist das Dokument jetzt so, das einem nur noch die wirklich wichtigen Sachen um die Ohren gehauen werden, und wer es Anwenden will muß ebend in die Kommandoreferenz schauen, sofern er das Kommando noch nicht kennt.

Ich halte das fuer Sinnvoller.

1.3 Die Beispiele

Sämtliche Beispiele in diesem Dokument beziehen sich auf mein Netzwerk zu Hause. Die meisten Beispiel werden jedoch mit dem CROW und TIGER gezeigt. Beide Rechner sind UNIX Rechner. Wobei TIGER eine Intelmaschine unter Linux ist und CROW eine UltraSPARC unter Solaris. Dazu muß gesagt werden das einen neuen in der RUnde gibt. RAVEN ist neu !

Damit Sie ein Überblick bekommen ist folgende Tabelle eventuell hilfreich :

Tabelle 1.1: Übersicht der verwendeten Beispielrechner

Hostname	Betriebssystem / Funktion
RAVEN	Ultra 10 / Solaris 8
CROW	Ultra 5 / Solaris 8
LX	SPARCstation LX / Solaris 8
EAGLE	Intel / Solaris 8
TIGER	Intel / Linux 2.2.10
PUMA	Intel / Linux 2.0.36
CHEETAH	Intel / Linux 2.2.10
FALCON	Intel / Linux 2.2.5
BUGGY	Intel / Windows 98
PANTHER	Intel / Novell 4.11

1.4 Für wen ist das Buch gedacht ?

Theoretisch sollte es jeder Lesen der mit UNIX in irgendeiner Form etwas zu tun bekommt. Administratoren und Systempfleger. Für Systementwicker steht zur Zeit noch nicht so viel drinnen. Der blutige Anfänger bekommt hier auch jede Menge Stoff mit vielen Beispielen.

1.5 Konventionen

In diesen Unterlagen herrscht eine strenge Typographie.

Dateinamen werden immer mit einem / (Slash) geschrieben. Dabei vollkommen egal ob es sich um ein DOS System handelt oder nicht. Dateinamen werden immer in diesem Font geschrieben

Kommandos werden im laufenden Text immer im `Courier Schriftsatz` geschrieben. Kommandos in Bildschrirmausschnitten die der Benutzer eingeben muß werden zusätzlich unterstrichen. Als Prompt kommt der Dollar (sh, ksh, bash) oder das Prozentzeichen (csh) zum Einsatz. Ist es ein Hash, dann ist der User root.

Hostnamen werden zur Zeit MIT DIESEM SCHRIFTSATZ notiert

1.6 History

Dieser kleine Abschnitt hält die Entwicklung des Dokumentes fest. Die Zeichen haben folgende Bedeutungen :

n (new)	Thema hinzugefügt
d (deleted)	Thema entfernt
u (update)	Thema vervollständigt bzw. aktuallisiert
r (reorg)	Thema neu strukturiert
pn (part new)	Ein kompletten Teil hinzugefügt

Version 0.2.1 Released sometime in 2000

Erste offizielle Release. Auch der Name des Dokuments wurde geändert ! Es heißt jetzt *Solaris Guide* und nicht mehr UNIX Guide. Das ist aber nicht weiter schlimm.

Version 0.1.x Diese steinalte Version ist nicht mehr verfügbar. Das Layout und andere Sachen waren nicht mehr nach meinem Geschmack. Das Dokument stammt aus dem Jahre 1997 oder so.

1.7 Noch mehr Infos

Dieses Dokument wurde komplett unter einer UNIX Plattform entwickelt. Als Textsatzsystem wurde LaTeX verwendet, in einer speziellen UltraSPARC Version. Für die meisten technischen Grafiken wurde `xfig` verwendet. ScreenShoots wurden mit dem Programm `xv` erstellt und eventuell mit `gimp` weiter verarbeitet. Geschrieben wurde der Text mit dem *vi*, und zwar mit dem `gvim` Version 5.5.

Kapitel 2

SUN MICROSYSTEMS OVERVIEW / PROCESSORARCHITECTURE

2.1 CISC / RISC

Es gibt zwei grundlegende Typen von Prozessoren. Es gibt zum einen die CISC[1] und zum zweiten die RISC[2] Prozessoren.

2.1.1 CISC

CISC Prozessoren haben ein grossen Set von Assembleranweisungen die teilweise sehr komplexe Berechnungen ausführen können. Da durch haben die Assembleranweisungen auch unterschiedliche Längen im Speicher. Das macht das ausführbare Programm sehr klein. Wird zum Beispiel ein Register um eins erhöht hat diese Anweisung nur ein Byte, die Berechnung von Sinus oder Cosinus verbrauch 6 Bytes.

2.1.2 RISC

RISC Prozessoren haben nicht soviele Assembleranweisungen. Um komplexe Berechnungen durchzuführen sind mehrere Anweisungen nötig. Die Anweisungen einer RISC haben immer die gleiche länge. Somit werden auch die Programme sehr groß, welches sich jedoch später als "der"Vorteil rauskristalisiert. RISC Prozessoren haben in der Regel auch ein paar Register mehr, meist 16 - 64 Stück. Da RISC *weniger* können ist auch die Anzahl der Transitoren im Prozessor wesendlich kleiner als bei CISC und verbraucht nicht so viel Strom, folglich werden die Prozessoren auch nicht so warm. Ein 300 MHz RISC kann passiv mit einem Kühlkörper ohne Lüfter versehen werden.

2.2 Specialarchitecture

2.2.1 Pipelines

Damit der Prozessor weis was er machen muß, durchläuft ein Maschinenbefehl mehrere Ebenen.

Instruction fetch Der Maschinenbefehl wird vom Speicher in den Prozessor geladen

Instruction decode Die Instruktion die geladen wurde, wird nun vom Prozessor dekodiert

Operand fetch Die nötigen Operanten werden in die Register des Prozessors geladen

Execute Die Instruktion wird ausgeführt

[1] `CISC`-Complex Instuction Set Computer
[2] `RISC`-Reduced Instruction Set Computer

Writeback Das Ergebnis der Operation wird entweder in ein Register oder wieder in den Speicher gestellt

Diese 5 Schritte sind nötig um ein einzigen Befehl auszuführen. Hat man nur eine solche Pipeline dann kann auch immer nur ein Befehl entsprechend ausgeführt werden.

Aus diesem Grunde haben Prozessoren meist mehr als eine Pipeline parallel am laufen. Während die erste Pipeline bei Stage 2 ist (Instruction decode) fängt die Pipeline 2 mit Stage 1 an (Instruction fetch) usw.

Hat man nun Maschinenbefehle mit unterschiedlichen Längen (CISC), dann ist der Aufbau einer solchen Pipeline durch aus sehr schwierig. Weil in der Stage 1 (Instruction fetch) weiß der Prozessor nicht wieviele Bytes er lesen muß. die Länge der Instruktion bekommt er erst am Ende von Stage 2 (Instruction decode) mit. Bei RISC Prozessoren dagegen existiert dieses Problem nicht.

2.2.2 Branch detection

Ein sehr gemeine Eigenschaft eines Programms besteht darin eine Prüfung auf einen Wert vorzunehmen und daraufhin jenach Wert woanderst hinzuspringen. Beispiel: Das programm prüft den Wert X auf Gleichheit von 10. Ist der Wert 10 dann mache xy, ansonsten überspringe die Aktion xy. Jedesmal wenn ein solche Branch in einem Programm vorkommt muß der Prozessor erst die Bedingung prüfen, danach darf er die nächste Instruktion wieder in seine Pipes laden.

D.h. die Pipes laufen sozusagen leer oder weden umsonst abgearbeitet. Um dieses teilweise zu verhinden gibt es sogenannte *Delay Branch Slots* wo die Anweisungen nach einem Branch aufbewahrt werden. Sollte der Sprung darüberhinaus erfolgen wird der Slot vernichtet.

Eine andere Methode ist die *conditional execution*. Diese Methode kommt ohne Sprung aus. Beispiel man hat folgenden Code :

Quelltext 2.2.1 Conditional Execution Example 1

```
IF ( B < C ) THEN
    A = D
  ELSE
    A = E
  ENDIF
```

Hat man nun einen guten Compiler der formuliert das in etwa so um :

Quelltext 2.2.2 Conditional Execution Example 2

```
COMPARE  B < C
IF TRUE  A = D
IF FALSE A = E
```

Es sind keine Sprünge notwendig. Letztlich ist es jedoch keine Hardwareeigenschaft eines Prozessors sondern mehr die Intiligenz des Compilers den man verwendet.

2.3 RISC - The Second Generation

2.3.1 Superscalar processors

Die Instruktionen sind im Speicher seriell, also eine nach der anderen, notiert. Superscalar bedeutet nun das der serielle Instruktionsfluß auf mehreren Recheneinheiten gleichzeitig bearbeitet wird. Man parallelisiert die Anweisungen. Die mehreren Recheneinheiten benutzen dafür mehrere unabhängige Pipelines. Sind die Befehle unabhängig voneinander können diese Parallel ausgeführt werden. Das Beispiel zeigt ein solchen Verhalt :

Quelltext 2.3.1 Superscalar Example 1

```
A = B + C
D = E + F
G = H + I
```

Werden jedoch Ergebnisse aus vorrangehenden Anweisungen weiterverarbeitet ist eine Recheneinheit auf das Warten des Ergebnisses verurteilt :

Quelltext 2.3.2 Superscalar Example 2

```
A = B + C
D = A + E
F = D + G
```

2.3.2 Superpipelining

Beim Superpipelining werden die einzelenen Stages von Pipelines nochmal aufgeteilt in kleinere Einheiten. Das ermöglicht eine noch höhere Geschwindigkeit und Parallelisierbarkeit. Eine MIPS R4000 z.B. hat 8 Stages.

2.3.3 The Post-RISC Architecture

Das Post-RISC Verfahren ist sein 1994 neu auf dem Markt. Es vereint alle bisherigen Nettigkeiten wei Superscalar und Superpipelining. Der Vorteil einer Post-RISC ligt in der

2.3.4 Speculative Computation

Diese Einheit liest die die Instruktionen und wenn sie merkt das es dort ein Befehl gibt der schon viel früher hätte ausgeführt werden können, dann tut sie es. Dazu werden alle Instruktionen nach dem Dekodieren in den IRB[3] abgelegt. Vom IRB geht es dann zur nächsten Stage der Execution. In dem IRB haben 60 - 120 Befehle Platz. Somit kann der Prozessor sich ein Bild dessen machen was Passiert. Der folgende Quelltext zeigt ein solches Speculative Computation :

[3] IRB-Instruction Record Buffer

Quelltext 2.3.3 Speculative Computation Example 1

```
A = B + C
.........  many other (30 - 50) instructions without using A
Q = A / 2
```

In der IRB wird erkannt daß (A) durch zweigeteilt werden soll und zwischen den beiden Anweisungen mit A nichts passiert. Ist die Fließkommaeinheit des Prozessors gerade in diesen 30-50 Anweisungen untätig dann wird das A geteilt durch 2 berechnet, das kann z.B. direkt nach A = B + C passieren. Kommt der Prozessor irgendwann mal an Q = A / 2 dann bekommt er das Ergebnis in der Zeit von einem Takt. Da das Ergebnis ja schon vorher berechnet wurde.

2.4 EPIC

EPIC[4] ist der letzte Schrei auf dem Markt zur Zeit. Erkommt auch im Intels neuem IA-64 (irgendwann) zum Einsatz. Die Länge eines Befehles ist sehr groß und in der Instruktion werden sehr viele Nebensächlichkeiten mitgeliefert wie

Predicated instruction Eine Instruktion die Conditional Execution ermöglicht

Speculative loads Möglichkeit das Speculative Computing zu steuern

Dazu kommen noch jede Menge parallele Recheneinheiten die alle über sehr viele eigene Pipelines verfügen.

2.5 The SPARC Architecture

Die SPARCs sind RISC Prozessoren.

2.5.1 SUN Bussysteme

SBUS Der SBUS ist der erste Bus von Sun. Er wurde 1982 mit den ersten SUN Rechnern gebaut. Er hat eine Breite von 32 bit und wird mit ca. 25MHz getaktet. Ist also nur etwas langsamer als PCI. (Das war 1982, grins). Mit SBUS werden auch Lasterdrucker angesteuert. SUN NewsPrinter 10 z.B. hat eine SBUS Karte und kann wegen der hohen Geschwindigkeit des 1985 gebauten Druckers nicht über eine Paralelleschnittstelle versorgt werden. (20-50 Seiten pro Minute Postscript 300dpi)

PCI Ja nun, bekannt aus Film und Vernzehn.

UPA DerUPA[5] kommt nur auf UltraSPARCs zum Einsatz. Da nur eine UltraSPARC in der Lage ist 2,6GB/s zu transportieren. Der UPA wird auf von Grafikkarten mit einer Speed von ca. 500MB/s benutzt. Der UPA wird meist zur Serverspiegelung eingesetzt als Crossbar.

[4] EPIC-Explicitly Parallel Instruction Computing
[5] UPA-Ultra Port Architecture

2.5.2 SUN Grafikkarten

PGX Eine PGX ist eine ATI-Mach64 (RageIIc) und kommt einmal als PGX24 und einmal als PGX32 zum Einsatz. Die PGX24 hat nur 2MB Speicher und kann maximal 1280x1024x8 Auflösung. Sie ist meistens on Board montiert. Die PGX32 hat doppelt soviel RAM und kann entsprechend höhere Auflösungen fahren. Manchmal kusiert die Karte auch als Raptor rum. Kostenpunkt ca. 600,- DM

Creator Die Creator-3D übernimmt bereits komplexe 3D Berechnungen und ist OpenGL fähig. Die Karte wird mit dem UPA angesteuert, da ein PCI Bus die Datenmenge nicht mehr verkraften kann. Die Creator-3D ist die kleinste der Serie und kostet ca. 3000,- DM. *SAKASMUS EIN: Jeder der eine Voodoo-18 mit 20-fach AGP unter einer Intel kennt, wird das Problem der 15 Lüfter nie lösen können. Eine Creator-3D ist ca. so groß wie ein komplettes Intel Board. Ihm hat jedoch keine Lüfter. SAKASMUS AUS*

Elite Ist etwas schneller (ca. doppelt so schnell wie eine Creator-3D) und kann neben OpenGL noch einige spezialitäten wie nebelige Reflektionen berechnen. Kostenpunkt ca. 6000,- DM

Expert SUN neustes Grafikboard. Kostet ca. 10000,- DM und ist für High-Performance Rendering gedacht.

2.5.3 Die kleinen älteren SPARC Rechner

2.5.4 Die aktuellen Workstations

SUN Ultra 5 Die SUN Ultra 5 ist in einem Desktopgehäuse aufbewahrt und kann von 270MHz UltraSPARC-IIi bis zum 360MHz UltraSPARC-IIi aufgerüstet werden. Maximal können 512MB Ram das edle Stück schmücken. Die SUN ULTRA 5 wird mit einer PGX-24 Grafikkarte on Board geliefert. Geeignet auch für mittlere Webserver und Mailserver. (Ein spezieller V9 Apache ist bei 270MHz ca. 3 mal schneller als auf einem AMD 300MHz (selftested)). Mit einem deutschen Preis von bis zu 5000,- DM mit 19-inch Monitor, man bekommt die U5 auch im 19-inch Rack schon für 2000 DM. Als Schnittstelle verwendet die U5 leider EIDE.

SUN Ultra 10 Ein süßer kleiner Tower. Kann bis zu 450MHz UltraSPARC-IIi bestückt werden. Insgesammt bekommt man 1GB Ram rein. Leider wird hier auch noch EIDE eingesetzt. Mit einer zusätzlichen Creator-3D wird eine U10 gerne im CAD Bereich eingesetzt, oder auch als Webserver, Mailserver etc. Preis ca. 8000,-. Mit einer Creator-3D kommt man ca. auf 10000,-

SUN Ultra 60 Bis zu 4 UltraSPARC-II Prozessoren mit je 400 MHz und 4MB Cache verkraftet das gute Stück. Als Schnittstelle dienen zwei Ultra-Wide SCSI Schnittstellen. Ideal für zu Hause und im CAD und Rendering Bereich, da mit einer Creator-3D oder mit einer Elite-m6 geliefert. Aber auch gerne als Webserver oder Datenbankserver eingesetzt.

SUN Ultra 80 Der zur Zeit neuste SUN Rechner.

2.5.5 Mittelklasse Server und High-End

Enterprise x00 Die E250 bis E450 werden in einem fast Cubischen nicht gerade leichtem Gehäuse geliefert. Die E450 hat (glaube ich) 20 Einschübe für Platten und wird mit 4 Ultra-Wide SCSI ausgerüstet. Ideal für Abteilungsserver und Applicationserver.

Enterprise x000 Nur größer. Ca. 180 cm hoch.

Enterprise 10000 (Starfire) / E10K Sau groß. Kann bis zu 64 Prozesoren des Typs UltraSPARC 400MHz mit 4MB Cache aufnehmen. Dabei sind immer 4 Prozessoren auf einer Einheit die während des Betriebes ausgetauscht werden können. Kostenpunk bei Vollausstattung: 24 Millionen Deutsche Märker. Wird meist im Bankgewerbe oder bei sehr grossen Webservern eingesetzt. Eine E10K verkraftet laut Aussage eines Mitarbeiters einer Bank bis zu 1000 User.

2.6 SunOS / Solaris Releases

Kapitel 3

DIE ENTWICKLUNGSGESCHICHTE VON UNIX

Kapitel 4

BEGRIFFE IN EINER UNIX UMGEBUNG

4.1 Der Kernel

Der Kernel ist das Herzstück eines UNIX Systems. Der Kernel ist ein Programm welches sämtliche Aktivitäten der Hardware steuert. Er verwaltet die Resourcen des Rechners. Weiterhin übernimmt er auch die Aufgabe der Prozessverwaltung und vieles mehr.

Die füheren Kernels älterer UNIX System sind monolithisch, d.h. alle funktionalitäten werden in einem großes Programm vereint. Der Kernel hatte dann meist eine größe von bis zu 5 MB gehabt. Neuere Systeme verwenden einen modularen Kernel der bei bedarf ein Treiber einfach dazu läd. Das macht den Startkernel kleiner und das System flexibler. Unter Linux ist ein modularer Kernel knapp 400kb groß.

4.1.1 Monolithische Kernel

Ein monolithischer Kernel beinhaltet alle Treiber eines System. Man muß vorher eine Liste von allen möglichen Geräten zusammenstellen und ein solchen Kernel erstellen. Benötigt man wärend der Laufzeit eines System einen Treiber der nicht im Kernel drinn ist, so muß ein neuer Kernel erstellt werden und das System muß neu gebootet werden. Das ist sehr negativ wenn es Kernel Patches gibt die z.B. ein Fehler beheben oder die Sicherheitslücke schliessen.

4.1.2 Modularer Kernel

Ein modularer Kernel besitzt nur soviele Treiber wie er benötigt um auf die Systempartition zuzugreifen. Danach kann der Kernel die nachfolgenden benötigten Treiber hinzuladen. Diese Treiber werden Module genannt. Der Vorteil ergibt sich aus der flexiblität des Systems. Wird ein neues Modul fuer die Netzwerkkarte benötigt, dann kann einfach das alte Modul entladen werden und das neue Modul wird anschließend wieder hinzugeladen und neu konfiguriert. Das System jedoch wird nicht neu gebootet und kann ganznormal weiterarbeiten.

Weitere Implementationen von Kernels gehen ein Schritt weiter, die laden ein Modul automatisch wenn es benötigt wird und entladen es automatisch wieder nach einer gewissen inaktiv Phase.

4.2 Systemcalls

Ein Systemcall ist ein Systemaufruf. Der Kernel bietet einem Anwendungsprogramm (Prozeß) die Möglichkeit die Hardware anzusprechen. Da der Kernel jedoch dem Prozeß den direkten Zugriff zur Hardware verweigert, benötigt der Prozeß eine Möglichkeit auf die Hardware zuzugreifen.

Der Kernel besitzt nun eine Reihe von Systemcalls, etwas 200 bis 500 Stück, die eine ganze Reihe an funktionalität bietet. Dazu zählen z.B. Lesen aus einer Datei, Maus abfragen, Daten über das Netzwerk senden, Bildschirmausgaben, etc.pp.

4.3 Devicefiles

Mit einem Devicefile (Gerätedatei) stellt der Kernel verfügbare Hardware im Dateibaum zur Verfügung. Ähnlich wie es DOS mit LPT tut. Der Kernel stellt z.B. für die Serielle Schnittstelle eine Gerätedatei zur Verfügung mit dem Namen `ttyS0`. Eine Anwendung die nun auf dem Seriellenport Daten senden möchte, öffnen nun einfach nur das Devicefile mittels dem Systemcall `open()` und schreibt die Daten mittels Systemcall `write()` auf den Port. Da ein Prozeß kein Zugang zur Hardware hat ist das der einzige Weg.

Leider sind die Namen der Geräte von UNIX Version zu Version unterschiedlich und nicht einheitlich.

4.4 Terminals

Mit einem Terminal wird die Einheit von Tastatur und Bildschirm bezeichnet. Für diese beiden Geräte reserviert der Kernel ein Devicefile. Wird aus dem Devicefile gelesen, bezieht sich das lesen auf die Tastatur, wird jedoch geschrieben bezieht sich das schreiben auf den Monitor. Weil anderstherum macht es noch nicht so den Sinn.

4.4.1 Virtuelle Terminals

Theoretisch gibt es pro Rechner immer nur ein Terminal. Durch ein Trick kann man jedoch mehrere Terminals benutzen. Dabei wird einfach nur der Bildschrim eines anderen Proßesses wieder restauriert. Linux z.B. unterstützt bist zu 12 echte virtuelle Terminals die man mittels STRG-ALT-F1 und STRG-ALT-F2 etc.pp. umschalten kann. In der grafischen Oberfläche bekommt jede geöffnetet Shell im Fenster ein virtuelles Terminal. Wobei es hier keine Begrenzungen gibt,

4.5 Signale

Signale werden vom Kernel verwaltet. Ein Prozeß kann einem anderen Prozeß ein signal senden. Die Wirkung des Signals ist jedoch Prozeß abhängig. Es gibt Prozesse die z.B. die Konfiguration neu einlesen, andere wiederum legen sich schlafen, andere beenden sich einfach. Die Signalfähigkeit eines Prozesses ist im Manual nachzulesen.

Ein Signal einfach nur eine bestimmte Nummer. Es gibt zwei Arten der Schreibweise von Signalen. Zum einen kann man es verbalisieren oder man verwendet dessen Nummer.

4.5.1 Standardsignale

UNIX definiert einige Signale die Standardisiert sind. Die folgende Tabelle schlüsselt einige Signale auf, dessen Wirkung meist überall gleich ist.

Tabelle 4.1: Übersicht von Standardsignalen unter UNIX

Signalname	Nummer	Bedeutung
SIGHUP	1	Die meisten Prozesse werden zur Neukonfiguration aufgerufen. Wird das Signal nicht abgefangen wird der Prozeß verlassen
SIGKILL	9	Der Prozeß wird vom Kernel beendet

Signalname	Nummer	Bedeutung
SIGTERM	15	Der Prozeß wird gebeten sich zu verlassen

Es gibt noch mehr ! (siehe signal(3HEAD))

4.6 Shared Memory

Wenn mehrere Prozesse sich eine einzigen Speicherbereich teilen wollen, muß Shared Memory verwendet werden. Da der Kernel den Adressbereich eines Prozesses gegenüber einen anderen Prozeß unerreichbar macht. Shared Memory wird meist bei Programmen verwendet die durch Performancegründen mehrere Prozesse gleichzeitig erstellen. Meistens Datenbanken oder Webserver, dort werden mehrere Prozesse gestartet die die Anfragen entsprechend auswerten.

4.7 Semaphoren

Eine Semaphore ist im Zusammenhang mit Shared Memory wichtig. Wenn mehrere Prozesse gleichzeitig auf ein Speicherbereich Zugriff haben, kann es vorkommen das zwei oder mehr Prozesse gleichzeitig Daten in diesen Bereich schreiben. Damit es dort nicht zu kollisionen kommt, werden Semaphoren eingesetzt. Wenn ein Prozess schreibend auf die Daten zugreiffen will muß es ein Semaphore setzen. Ist dieser bereits gesetzt muß der Prozeß bis zur Freigabe blockiert werden um es später noch einmal zu versuchen.

4.8 Filedescriptor

Ein Filedescriptor (Dateibeschreiber) ist eine Zugruffsnummer mit dem ein Prozeß auf eine Datei zugreifen kann. Wenn ein Prozeß eine Datei öffnet bekommt er vom Kernel einen fd[1]. Diesen fd muß sich der Prozeß merken. Will der Prozeß nun Daten lesen oder schreiben muß er den fd verwenden. Es bestehen zwei Einschränkungen. Zum einen kann ein Prozeß nicht mehr als 255 fd's belegen. Einige Betriebssysteme mehr oder weniger. Zum zweiten kann der Kernel nicht mehr als 4096 fd's verteilen. Auch hier ist der Wert Betriebssystemabhängig. Die hier geschriebenen Werte gelten für ältere UNIX Systeme. Heute sieht es so aus das man diesen Wert beliebig verändern kann. Besonderst bei sehr großen Server reichen 4096 bei weitem nicht.

4.8.1 Standard Filedescriptoren

Jedes UNIX System belegt standardmäßig die ersten 3 fd's mit Standardkanälen. Die Kanalnummern 0,1 und 2 werder mit dem aktuellen Terminal verbunden. Wobei jeder Kanal eine besondere Bedeutung besitzt. Man kann jeden einzelnen Kanal umleiten in eine Datei z.B. Die folgende Tabelle zeigt die Bedeutung und die Namen der Kanäle.

[1] fd-Filedescriptor

Tabelle 4.2: Standardfiledescriptoren eines Prozesses unter UNIX

fd	Name	Bedeutung
0	stdin	Verbunden mit der Tastatur des Terminals. Der Prozeß liest über diesen Kanal die Eingaben von der Tastatur
1	stdout	Verbunden mit dem Bildschirm des Terminals. Der Prozeß benutzt diesen Kanal um seine Ausgaben zu tätigen
2	stderr	Verbunden mit dem Bildschirm des Terminals. Der Prozeß benutzt diesen Kanal nur dann, wenn der Prozeß einen Fehler melden will. z.B. Datei nicht gefunden, etc.pp.

Wenn ein Prozeß z.B. den stdin Kanal schliesst kann er keine Eingaben mehr von Tastatur entgegen nehmen. Webserver oder andere Server tun dieses.

4.9 Benutzer und Gruppen

Unter UNIX werden Gruppen und Benutzer angelegt. Jede Gruppe und jeder Benutzer bekommen dann eine eindeutige Nummer zugewiesen. Mit dieser Nummer wird die Gruppe bzw. der Benutzer identifiziert. Diese Nummern nennen sich UID (User-ID) und GID (Group-ID). Um Zugriff zu Daten zu bekommen wird die UID und GID mit denen der Datei verglichen. Es gibt im System ein Benutzer der alles darf (ALLES !) dieser Benutzer hat die UID 0 (Null). Der absoluter Administrator.

4.10 Ein Prozeß

Ein Prozeß in einem UNIX System ist ein normales Programm. Jeder Prozeß in einem UNIX System identifiziert sich durch eine eindeutige Nummer, die PID. Jeder Prozeß besitzt auch eine lokale Tabelle mit den geöffneten Dateien (Lokale Filedescriptor Tabelle). Er besitzt die Information an welches Terminal er angeschlossen ist. Ein Prozeß besitzt so viele Informationen, daß wir diese auflisten müssen :

PID Die PID ist die Prozeß-ID. Mit dieser Nummer wird der Prozeß in einem System identifiziert.

PPID Die PPID ist die PID des Eltern-Prozesses. Wenn ein Prozeß gestartet wird, merkt sich dieser von wem er gestartet wurde. Es gibt nur ein Prozeß in einem UNIX System der keine PPID hat, und das ist der Kernel selbst.

PGID Ein Prozeß kann zu einem Prozessgruppenführer gemacht werden. Die PID des Gruppenführers wird dann an alle neuen Prozesse weitergegeben. Ein Prozessgruppenführer ist zum Beispiel die grafische Oberfläche, wird diese Verlassen werden alle Prozesse mit dieser PGID gekillt.

STATE In STATE wird der Status eines Prozesses festgehalten. Ein Prozeß kann Running, Sleeping, Suspend sein.

PRI Jeder Prozeß besitzt gegenüber allen anderen Prozessen eine Priorität. Je nach Betriebssystem bedeutet eine höhere Zahl eine höhere Priorität. Einige Betriebsysteme erlauben sogar die Einstellung Real-Time. Dort wir dann meist der Prozeß auf ein Prozessor verteilt. Hat man nur ein Prozessor im System hat man verloren.

UID, GID In den beiden Feldern wird die Benutzerkennung des Prozesses festgehalten. Hier wird nur gespeichert wem der Prozeß gehört

EUID, EGID In diesen beiden Feldern wird fest gehalten mit welchem Benutzerrechen der Prozeß sich im System bewegt. Die kann also eine andere sein wie UID und GID. Beispiel : Bei der grafischen Oberfläche ist der direkte weg zur Hardware nötig. Das jedoch wird einem normalen Benutzer verwehrt. Also arbeitet der Prozeß intern mit der root Kennung.

TTY In diesem Feld wird das dazugehörige Terminal gespeichert. Prozesse die kein Terminal bedienen nennt man Daemonen, die stellen Dienste im Hintergrund zur Verfügung. Beispiel : Webserver, Fileserver, Nameserver, etc.pp.

SIGNAL Jeder Prozeß besitzt immer eine Schnittstelle zu Signalen. Jeder Prozeß kann ein Signal abfangen um eine eigene Routine zu implementieren. Das einzige Signal (SIGKILL (9)) kann nicht angefangen werden.

FD Jeder Prozeß besitzt eine Tabelle wo er seine zur Zeit geöffneten Dateien vermerkt. Normalerweise wird dem Prozeß immer stdin, stdout und stderr geöffnet.

CWD Hier wird das aktuelle Verzeichnis eines Prozesses gespeichert. Somit weiß der Prozeß immer wo er ist.

TIMES Jeder Prozeß verfügt über die angaben wieviel Zeit er verbrät. Diese werden hier gespeichert.

Alle diese Informationen lassen sich mit den entsprechenden Tools ermitteln.

4.11 Shell

Eine Shell ist ein Kommandointerpreter. Eine Shell liest ein Kammando von der Tastatur ein und führt dieses aus. Man kann die Kommandos auch in eine Datei schreiben und diese Datei ausführen. Die Shell interpretiert dann den Inhalt der Datei und führt die Befehle dort aus. Eine solche Datei nennt sich Shellscript. Unter UNIX gibt es einige Shells die alle etwas anderst funktionieren. Die Bekannteste und älteste ist die Bornce Shell, der Nachfolger nennt sich Kornshell, unter Linux existiert die Bornce Again Shell und es gibt noch die C-Shell. Alle Shells funktionieren auf allen Betriebssystemen gleich. Somit ein ein Shellscript welches sich an den Standard der Shell hält Plattformunabhängig. Vergleichbar ist eine Shell mit dem command.com unter DOS. Ein Shellscript wäre vergleichbar mit einer Batchdatei unter DOS.

Kapitel 5

DER LOGIN UND DER PROMPT

5.1 Das Login

Nachdem Ihr Client gestartet wurde zeigt sich auf dem Bildschirm der Text `login:`. Das System wartet nun auf die Eingabe Ihres Benutzernamens, den Sie vom Dozent bekommen. Nach der Eingabe des Benutzernamens kommt normalerweise die Passwortabfrage, die unter Umständen nicht kommt. Nach dem erfolgreichen Login zeigt sich der Prompt

5.2 Der Prompt

Unter UNIX gibt es verschiedene Arten eines Prompts. Der folgende Ausschnitt soll mal einige davon zeigen :

```
$
%
tiger: otto $
[otto @ tiger]/home/otto >
```

Bildschirmausschnitt 5.2.1: Beispiele von Prompts

Der erste Prompt signalisiert die Verwendung einer Standardshell. Der zweite Prompt signalisiert das es sich um die C-Shell handelt. Der dritte Prompt zeigt Ihnen den Hostnamen und den Benutzernamen an. Der vierte zeigt Ihnen zudem noch das aktuelle Verzeichnis an, der vierte entspricht dem des MS-DOS zum Teil und wird bei SuSE-Linux verwendet.

5.2.1 Telnet

die Sie gerade benutzen unterstützt Telnet. Sie können sich also nicht nur auf Ihrer Maschine einloggen sondern auf allen anderen auch. Dazu müssen Sie den Namen des Rechners kennen, dann können Sie sich mittels `telnet rechnername` erneut auf einen anderen Rechner einloggen.

Den Rechnernamen bekommen Sie mit `uname` (siehe weiter hinten) raus.

Kapitel 6

EINFACHE BEFEHLE

Bevor man ein UNIX System nutzen wollen, müssen Sie sich erst einmal über das System Informieren. Dazu existieren auch einige Befehle.

6.1 Hilfen eines UNIX Systems

Ein UNIX System ist sehr gross. In einem UNIX System gibt es jenach installation zwischen 200 und 5000 einzelne Befehle. Die kann man sich alle nicht merken. Viele jedoch muß man einfach wissen. Was jedoch noch erschwerend hinzu kommt : Jeder Befehl hat unterumständen 5 bis 20 Aufrufoptionen. Ächz. Ein UNIX System stellt Ihnen glücklicherweise eine sehr perfekte Hilfe zur Verfügung.

6.1.1 man

`man`[1] ist ein Programm, daß ein anderen Programmnamen von Ihnen als Argument erwartet. Es sucht dann den Namen des Programms in seiner Onlinehilfe und zeigt diese Hilfe auf den Bildschirm an. Man kann nun mittels `CTRL-B` und `CTRL-F` in der Hilfe navigieren. Die Hilfe selbst wird durch `q` wieder Verlassen. Man kann auch nach Begriffen darin suchen. Diese Hilfeseiten nennt man *manpages*. In diesem Dokument werden Namen von Manpages wie folgt geschrieben : `ls`.

6.1.2 Answerbook™

Das Answerbook™[2] ist eine Sammlung von Dokumenten unter Solaris die bescheiben, wie etwas funktioniert und wie es konfiguriert wird. Zur Betrachtung ist ein Webbrowser nötig. Fragen Sie mal Ihren Dozenten ob im Haus ein Answerbook installiert ist.

6.2 Systemermittlungsbefehle

Diese Befehle sind meist dann direkt aufzurufen, wenn man nicht weiß mit welchem System man arbeitet. Der Einsatz dieser Befehle macht sich jedoch in Programmen sehr gut. Da jedes UNIX System seine Dateien und Konfigurationsdateien wo anderst speichert, kann ein Programm anhand dieser Befehle feststellen um welches System es sich handelt.

uname Das Programm `uname`[3] gibt Auskunft über das Betriebssystem. Die Ausgabe umfasst die Angabe des Betriebssystems, Hostname, installierte Patches und verwendete Hardware.

arch Das Programm `arch`[4] gibt Auskunft über die Hardware Architektur des Rechners.

[1] `man` - Siehe Kommandoreferenz Seite 323
[2] Eingetragenes Warenzeichen von SUN Microsystems
[3] `uname` - Siehe Kommandoreferenz Seite 405
[4] `arch` - Siehe Kommandoreferenz Seite 247

hostname Das Programm `hostname`[5] gibt Auskunft über den Hostnamen des Rechners. Unter Linux wird der Hostname mit dem Programm auch gesetzt. Man sollte jedoch `uname -n` einsetzen.

6.3 Benutzerorientiertebefehle

Ein Benutzer besitzt einige Kennungen, wie z.B. UserID, etc.pp. Um diese Kennungen werden wir uns später kümmern. Es ist jedoch wichtig zu wissen wie man im Notfall seine BenutzerID's bekommen kann.

logname Gibt Ihnen Ihren loginnamen aus. (siehe 41 auf Seite 305)

id Gibt Auskunft über die User ID und Group ID eines Benutzers (siehe 41 auf Seite 291)

groups Gibt Auskunft über alle zugehörigen Gruppen eines Benutzers (siehe 41 auf Seite 283)

who Das Programm `who`[6] gibt eine Liste von eingeloggten Benutzern auf den Bildschirm aus.

write Mit `write`[7] können Nachrichten auf den Bildschirm eines anderen Benutzers geschrieben werden.

mesg Das Programm `mesg`[8] wird eingesetzt um den Empfang von `write` Nachrichten zu erlauben oder zu verbieten.

mail Mit `mail`[9] können EMails gesendet werden. Mit `mail` können jedoch auch EMails gelesen werden. In Verbindung mit dem Internet kann auch `mail` verwendet werden.

6.4 Prozessorientiertebefehle

Prozesse werden von Ihnen gestartet. Um sich ein überblick der Prozesse zu schaffen gibt es die folgenden Befehle. Auch hier kann es sehr hilfreich sein diese Befehle sich zu merken.

tty Gibt Auskunft über das aktuelle Terminaldevice

ps Gibt Auskunft über Prozesse. (siehe **??** auf Seite ??)

kill Mit `kill` werden Signale an Prozesse gesendet (siehe 41 auf Seite 301)

sleep Mit `sleep` wird der aktuelle Prozeß einige sekunden lang schlafen gelegt. (siehe 41 auf Seite 389)

[5] `hostname` - Siehe Kommandoreferenz Seite 289
[6] `who` - Siehe Kommandoreferenz Seite 409
[7] `write` - Siehe Kommandoreferenz Seite 411
[8] `mesg` - Siehe Kommandoreferenz Seite **??**
[9] `mail` - Siehe Kommandoreferenz Seite 321

6.5 Praktikum

- Installieren Sie zuerst die Clientsoftware (folgen Sie den Anweisungen des Dozenten)
- Loggen Sie sich ein. Benutzen Sie dazu die Kennung die Sie vom Dozenten erhalten haben.
- Benutzen Sie nun `uname` um festzustellen mit welchem Betriebsystem Sie arbeiten.
- Benutzen Sie `uname -n` um Ihren Rechnernamen ausfindig zumachen
- Benutzen Sie das Programm `id` um Ihre UserID ausfindig zumachen
- Mit welchen Programm können Sie sich alle Gruppenzugehörigkeiten anzeigen lassen ? Verwenden Sie dieses Programm um sich zu Informieren
- Stellen Sie fest auf welchem Terminal Sie gerade arbeiten
- Stellen Sie fest welche Prozeß-ID Ihre Shell hat
- Lassen Sie sich alle Prozesse ausgeben die Ihnen gehören.
- Benutzen Sie nun den Rechner Ihres Nachbarn, loggen Sie sich dort lokal ein. Nachdem Sie sich eingeloggt haben führen Sie `telnet rechnername` aus. Ersetzen Sie `rechnernamen` durch den Namen Ihres Rechners. Loggen Sie sich auf Ihren Rechner ein.
- Lassen Sie sich alle Prozesse ausgeben die Ihnen gehören. Woran können Sie nun erkennen welche Prozesse auf welchem Terminal ausgeführt werden ?
- Starten Sie nun auf Ihrem Rechner das Programm `sleep 10000` um ein Prozeß zu simulieren der abgestürtzt ist.
- Gehen Sie wieder zu Ihrem Nachbarn und holen sich von dort aus die Prozeß ID des `sleep` Prozesses und schießen den Prozeß mittels `kill` ab. Auf Ihrem Rechner erscheint der Prompt wieder.
- Um die Telnet Session zu verlassen brauchen Sie nur die Shell mit `exit` zu verlassen. Ihren Nachbar Rechner brauchen Sie dann nicht mehr. Wiederholen Sie die vorgehensweise indem Ihr Nachbar Ihren Rechner benutzt.
- Suchen Sie sich einen Teilnehmer den Sie besonderst mögen. Fragen Sie den nach dem Rechnernamen und loggen sich auf dessen Rechner ein.
- Lassen Sie sich dort anzeigen wer auf dieser Station eingeloggt ist und schreiben Sie irgendjemanden eine Nachricht mit `write`

Kapitel 7

UNIX SHELLS

Die UNIX Shell ist sozusagen der *command.com* unter DOS. Eine UNIX Shell hat die Aufgabe die Eingabe die Sie tam Prompt eingeben entsprechend zu analysieren und das dazugehorige Programm auszuführen.

In diesem Kapitel werden wir nur auf die einfachen Shellmechanismen eingehen, weiter hinten befinden sich die Kapitel zur Shellscriptprogrammierung wo dann auch die Unterschiede der Shells dargestellt werden.

7.1 Verfügbare Shells

Ein UNIX System ist groß, es gibt viele UNIX Systeme. Um zwischen den einzelnen UNIX Systemen auch nur annähernd eine gewisse kompatiblität zu gewährleisten werden meistens Shellscripts zur Programmierung benutzt. Ähnlich wie Batchdateien unter DOS. Um dieses jedoch zugewährleisten gab es mal einen Mr. Bourne der sich genau über dieses Problem den Kopf gemacht hat. Er definierte eine Sprache und definierte die Funktionsweise einer Shell. Ab diesem Zeitpunkt mußte ein UNIX System immer eine Bourne-Shell implementation mitbringen. Diese Implementation musste 100%-ig kompatibel sein. Das jedoch war nur der Anfang

Bourne Shell Ist bei allen System V Betriebssystemen vorhanden. Ist auf fast allen UNIX Versionen vorhanden. Das Programm dazu ist `/bin/sh`. Um diese Shell werden wir uns auch erst einmal kümmern.

C Shell Die C-Shell wurde bei BSD Systemen eingeführt, da die Bourneshell Lizenzsoftware ist, aber BSD mehr oder weniger frei war. Die C-Shell funktioniert auf einer ganz anderen Basis. Bourneshell und C-Shellscripts sind untereinander nicht austauschbar. Die C-Shell ist jedoch für C-Programmierer ideal. Als Programm wird meist `/bin/csh` verwendet

Korn-Shell Mr. Korn hat die Bourne Shell erweitert aber die kompatiblität gewahrt. Wird heute meist als Standardshell für Benutzer konfiguriert. Als Programm wird `/bin/ksh` verwendet. Über die speziallitäten dieser Shell sollten Sie ein Blick in die Shellscriptprogrammierung werfen. Alles war die Bourne Shell kann, kann die Korn-Shell auch.

Bourne Again Shell Diese Shell ist auf freien UNIX Derivaten (like Linux) verfügbar. Sie hält sich einigermassen an den Korn-Shell Standard mit sehr vielen Erweiterungen. Normalerweise erreicht man die Shell via `/bin/bash`. Alles war die Korn-Shell kann, kann Bourne Again Shell auch.

7.2 Die Kommandozeile

Die Kommandozeile einer Shell ist ein wirklich mächtiges Instrument. Die Anzahl der Möglichkeiten der Shell sind sehr groß. Richtig kombiniert bekommt man ein sehr gutes Tool. Wendet man die Shell jedoch falsch an kann es zu empfindlichen Chrashs kommen.

Wenn man die dolgende Auflistung jedoch beachtet dann kann fast nix schiefgehen. Wenn die Shell eine Eingabezeile von Ihnen bekommt macht sie folgendes :

- Die Shell nimmt Ihre Eingabezeile auseinander und sucht dort Leerzeichen und Tabulatoren. Teilt an diesen Stellen die Eingabezeile in Argumente
- Untersucht nun alle Argumente der Eingabezeile nach Shell-Besonderen Zeichen (wzB Metazeichen Umlenkungen etc.pp.). Die Shell interpretiert die Besonderen Zeichen und ersetzt wenn nötig das Argument mit dem Ergebnis der Interpretation
- Durchsucht wieder alle Argumente ..... siehe 2. Das macht Sie solange bis nichts mehr da ist
- Die Shell sucht im Suchpath den Programnamen des ersten übergebliebenen Arguments. Findet die Shell das Programm, dann führt sie es aus. Ansonsten gibt es eine Fehlermeldung: *command not found*

Diese grobe Übersicht sollte man sich immer bewußt sein beim Arbeiten mit der Shell.

7.3 Sonderzeichen der Bourne Shell

7.3.1 Teilen in Argumente

Die Teilung der Argumente ist am besten am folgenden Beispiel erklärt :

Das Programm `echo` gibt das aus was man dem Programm als Argument übergibt. Das folgende Beispiel zeigt den `echo` Befehl in Aktion :

```
$ echo Hallo Welt
Hallo Welt
$
```

Bildschirmausschnitt 7.3.1: Teilung in Argumente Beispiel 1

Soweit so Gut. Die Shell hat in diesem Befehl 3 Argumente gesichtet. Zum einen `echo`, `Hallo` und als drittes `Welt`. Das konnte sie weil dort Leerzeichen zwischen sind. Geht man nun hin und macht mehrere Leerzeichen dazwischen ignoriert die Shell die nachfolgenden Leerzeichen und man bekommt das gleiche in grün :

```
$ echo Hallo       Welt
Hallo Welt
$
```

Bildschirmausschnitt 7.3.2: Teilung in Argumente Beispiel 2

Auch hier erkennt die Shell nur 3 Argumente. Das heisst die Shell betrachtet das Leerzeichen bereits als ein besonderes Zeichen. Will man jedoch ein besonderes Zeichen einem Programm übergeben dann muß man es flüchten, oder vor der Shell verstecken oder Entwerten. Alle Begrifflichkeiten meinen das gleiche.

7.3.2 Auf der Flucht

Um ein besonderes Zeichen vor der Shell zu verstecken können mehrere Zeichen verwendet werden, die dann wieder zu besonderen Zeichen der Shell werden, ein Teufelskreis.

Um ein einzelnes Zeichen als Fluchtzeichen zu markieren kann der Backslash verwendet werden. Der Backslash entwertet alle besonderen Zeichen. Um jedoch ein Backslash einem Programm zu übergeben muß man ein Backslash entsprechend entwerten. Also man benötigt zwei Backslashe.

```
$ echo Hallo \ \ \ Welt
Hallo    Welt
$ echo Ein Backslash \\
Ein Backslash \
$
```

Bildschirmausschnitt 7.3.3: Auf der Flucht mit einem Backslash

Hat man jedoch wie in diesem Beispiel mehrere Zeichen hintereinander zu flüchten wird es sehr schnell unübersichtlich.

Dann können die einfachen Anführungsstriche etwas abhilfe schaffen. Eine einfache Anführungszeichen Konstruktion hat immer zwei Anführungsstriche. Da wo es anfängt und dort wo es aufhört. Fehlt ein Zeichen gibt es Fehler und unvorhersehbare Ergebnisse. Um ein einfaches Anführungszeichen einem Programm zu übergeben muß es geflüchtet werden :

```
$ echo 'Hallo    Welt'
Hallo    Welt
$ echo Mit \' kann man Zeichen fluechten
Mit ' kann man Zeichen fluechten
$ echo Das \' fluechtet den '\' und das '$'
echo Das ' fluechtet den \ und das $
$
```

Bildschirmausschnitt 7.3.4: Auf der Flucht mit einfachen Anführungsstrichen

Zwischen den anführungsstrichen findet die Shell nun keine Leerzeichen und stellt dem Programm das Konstrukt `Hallo Welt` als Argument ein. Weiterhin sieht man das ein einzelnes Anführungszeichen auch den Backslash flüchten kann.

Und zu guter letzt noch die dritte Möglichkeit. Mit doppelten Anführungszeichen können auch Zeichen geflüchtet werden, jedoch keine Backslashe und keine einfachen Anführungszeichen, und auch kein Dollar. In der Programmierung werden doppelte Anführungszeichen des öffteren eingesetzt.

```
$ echo "Hallo     Welt"
Hallo     Welt
$ echo Mit \" kann man KEIN "\\" fluechten
Mit " kann man KEIN \ fluechten
$
```

Bildschirmausschnitt 7.3.5: Auf der Flucht mit doppelten Anführungszeichen

7.3.3 Metazeichen / Wildcards

UNIX Programme wie `cp` oder `ls` oder ... können mit dem Stern * nix anfangen! Fast alle UNIX Programme interpretieren den * als ein ganz normales Zeichen in einem Dateinamen. Der Mechanismus wurde von den Programmen auf die Shell gelegt. Das hat einen Vorteil : Die Programme sind kleiner. Aber auch ein Nachteil : Man muß etwas mehr ausfassen.

Man kennt vom DOS her eventuell folgenden Befehl : `COPY *.* C:`. Der Befehl `COPY` bekommt hier als Argument das `*.*` und es muß sich selbst um alle Dateinamen kümmern.

Unter UNIX erledigt das die Shell und übergibt dem Programm die gefundenen Dateinamen. Das Programm selbst kann keine Windcards auflösen. Das heisst : Wildcards können überall eingesetzt werden. Egal um welches Programm es sich handelt.

Die folgenden Beispiele Beziehen sich auf ein Verzeichnis wo folgende Dateien sich drinn befinden :

```
$ ls
abc.gif    abcdef     c.text     Hallo  hallo
ABCDEF     bcdef      GIF.datei  Hello  hello
$
```

Bildschirmausschnitt 7.3.6: Beispielverzeichnis für Wildcards

Der Stern *

Der Stern steht für eine beliebige Anzahl von beliebigen (auch 0 (NULL)) Zeichen.

Die folgenden Beispiele zeigen die verwendung des Sterns

```
$ echo *
ABCDEF GIF.datei Hallo Hello abc.gif abcdef bcdef c.text hallo hello
$ echo a*
abc.gif abcdef
$ echo A*
ABCDEF
$ echo *.text
c.text
$ echo *.*
GIF.datei abc.gif c.text
$ echo H*ll*
Hallo Hello
$ echo *cd*
abcdef bcdef
$ echo * Toll *
ABCDEF GIF.datei Hallo Hello abc.gif abcdef bcdef c.text
hallo hello Toll ABCDEF GIF.datei Hallo Hello abc.gif abcdef
bcdef c.text hallo hello
$ echo "* Toll *"
* Toll *
$ echo \* Toll \*
* Toll *
$
```

Bildschirmausschnitt 7.3.7: Beispiele zum Metazeichen Stern

Geht man jedoch in ein Verzeichnis ohne Dateien hinein und/oder die Shell findet keine Übereinstimmungen dann übergibt sie dem Programm das Metazeichen :

```
$ echo Z*
Z*
$ cd emptydir
$ echo *
*
$ cd ..
$
```

Bildschirmausschnitt 7.3.8: Metazeichen Stern ohne Auflösung

Das Fragezeichen ?

Das Fragezeichen steht für genau ein beliebiges Zeichen.

Die eckigen Klammern [ ]

Die eckigen Klammern stehen genau für ein Zeichen aus der Menge in den eckigenen Klammern.

7.3.4 Umlenkungen

7.3.5 Übersicht der Sonderzeichen der Bourne Shell

Nocheinmal als Übersicht :

Fluchtzeichen	flüchtbare Zeichen	nicht flüchtbare Zeichen
\	Flüchtet alle Sonderzeichen der Shell. Um jedoch auch ein \zu erhalten muß man zwei \'e machen	keine
'	Flüchtet alle Zeichen bis auf sich selbst.	sich selbst, also ', geht nicht
"	Flüchtet Leerzeichen	fast alle

Tabelle 7.1: Übersicht der Fluchtzeichen der Bourne Shell

Kapitel 8

GRUNDLAGEN ZUM DATEISYSTEM

In einem UNIX Dateiensystem gibt es für einen Menschen Dateien, Verzeichnisse, Gerätedateien und jede Menge mehr. Ein UNIX System jedoch sieht alles als Dateien an. Wenn im folgenden von Datei gesprochen wird kann also auch ein Verzeichnis gemeint sein.

Jede Datei in einem UNIX System kann einen Namen mit bis zu 14 Zeichen besitzen. neuere UNIX Systeme jedoch können bis zu 255 Zeichen lange Dateinamen haben. Ein Beschränkung der zu verwendenen Zeichen gibt es nicht. Unter MS-DOS gibt es zur Dateiidentifikation ein Suffix, wzB. .gif für Gifdateien, unter UNIX Systemen ist so etwas optional.

Jede Datei hat immer ein Eigentümer und eine Gruppenangehörigkeit. Der Eigentümer darf die Rechte einer Datei verändern. Ansonsten zählen die Rechte die er hat.

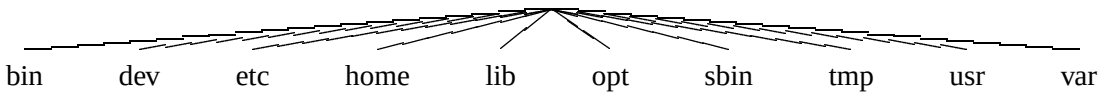
Jede Datei in einem UNIX System besitzt Rechte die über den Zugriff eines Benutzers entscheiden. Die Rechte kann der Eigentümer entsprechend ändern.

8.1 Verzeichnisstruktur

UNIX Systeme ordnen Dateien ähnlich wie DOS in einem Dateienbaum. Jedoch wird hier eine Ebene mit einem Slash getrennt und nicht mit einem Backslash.

8.1.1 Der Standardbaum

Die folgende Grafik zeigt ein Standardbaum der meist so vorhanden ist.



Er ist nicht vollständig bietet jedoch ein kleinen Überblick.

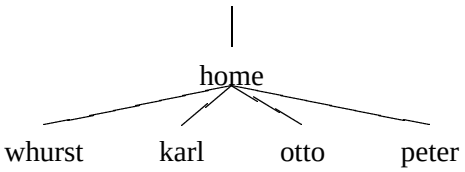
bin Das bin Verzeichnis existert mehrfach im Dateibaum. Dort befinden sich ausführbare Programme. Unter Linux stehen im /bin Verzeichnis die Programme die das System benötigt um soweit hochzufahren bis es die anderen Massenspeicher erreichen kann. Unter Solaris ist dies ein Link auf /usr/bin

dev Hier befinden sich immer alle Gerätedateien

etc Im etc Verzeichnis befinden sich fast alle Konfigurationsdateien zum System. Im Normalfall kann man das /etc Verzeichnis als Systembackup betrachten. Unter Solaris werden hier auch alle Scripts abgelegt die für den Systemboot verantwortlich sind.

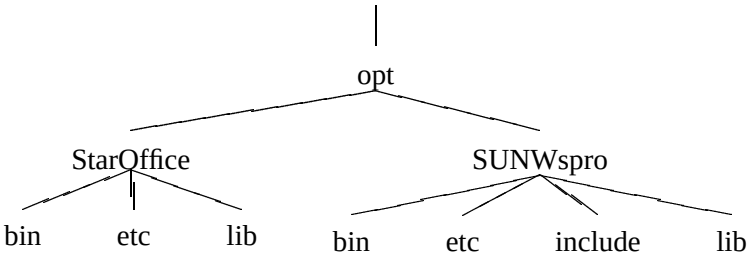
export Unter Solaris wird export für Freigaben und Heimatverzeichnisse lokaler Benutzer verwendet. Unter Linux ist es nicht vorhanden.

home Unter home befinden sich meist alle Heimatverzeichnisse



lib Systembilioteken findet man hier. Unter Solaris ist es wieder ein Link nach /usr/lib unter Linux befinden sich im /lib/modules die Kernelmodule

opt Software die normalerweise nicht im Lieferumfang des Betriebssystemes stehen werden hier installiert. Wenn eine Software sich in diesen Baum einkopiert erstellt diese meist ein Unterverzeichnis mit einem Produktnahen Namens. Darunter erstellt es dann sein eigenes bin, etc usr. Verzeichnis. Somit kann jeder Benutzer der Schreibrechte auf opt hat Pakete installieren. Bietet somit schutz vor dem bereits installiertem System. Das folgende Beispiel zeigt den Baum von einem Office und vom WorkShop C++



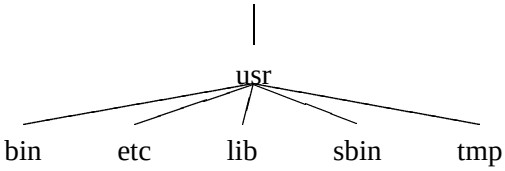
sbin In sbin Verzeichnissen befinden sich Programme für den Administrator. Im sbin Verzeichnis befinden sich meist Serverprogramme, Hardwarekonfigurationstools. Im /sbin befinden sich nur die Programme die das System benötigt um die anderen Dateiensysteme zu erreichen. Serverdienste liegen meist in /usr/sbin. Unter SuSE-Linux und ULTRIX liegen die Start und Boot Scripts hier. Vollkommen unüblich in der UNIX Welt und wird von mir als Dozent nicht unterstützt.

tmp Dieses Verzeichnis ist die Globale ablage von temporären Dateien. Normalerweise wird das /tmp Verzeichnis beim booten immer gelöscht. Unter Linux passiert das leider nicht. Gehen Sie mal davon aus das das System mit dem Sie im Unterricht entfernt arbeiten das /tmp Verzeichnis nach jedem Booten löscht.

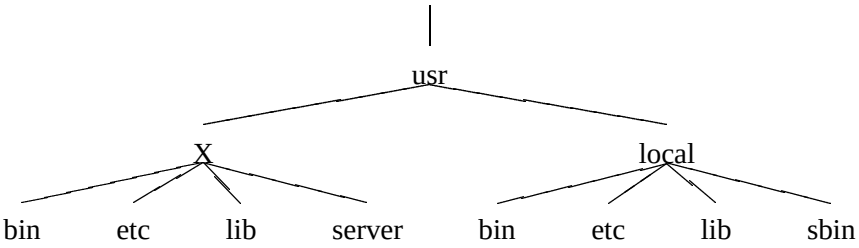
usr Das usr ist so umfangreich das wir es etwas aufteilen müssen. Unter dieser Verzeichnisstruktur befinden sich alle Programme und Dateien die es gibt. In fast 100% aller Fälle kann nur der Administrator auf /usr schreibend Zugreifen, d.h. für alle User gilt : Nur Lesen. Das ermöglicht das entfernte Einbinden des gesamten /usr Verzeichnisses von einem Netzwerkserver aus. Der Netzwerkserver stellt das /usr nur zum Lesezugriff zur Verfügung. Das wird meist bei Clients so gemacht um den Administrationsaufwand pro Maschine zu sparen und somit die TCO[1] niedrig zu halten im Vergleich mit MS-Windows Produkten.

Zum einen gibt es dort wieder die bekannten Verzeichnisse :

[1] TCO-Total Cost of Ownership



Aber es gibt noch sehr viele spezialfälle die wir nochchten müssen :



Im Unterverzeichnis /usr/X befinden sich alle Programme die mit der grafischen Oberfläche zu tun haben. Unter SuSE-Linux wird das meist nicht sehr ernst genommen und dort landen Programme für die grafische Oberfläche auch schonmal nach /usr/bin. Unter Solaris wird das noch stärker differenziert in dem CDE Anwendungen unter /usr/dt sich befinden und ältere OpenWin Anwendungen sich im /usr/openwin befinden. Unter Solaris ist das /usr/X ein Link auf /usr/openwin.

Das /usr/local ist ein ganz besonderes. In heterogenen UNIX Umfeldern kommt es zum Einsatz von mehreren Architekturen gleichzeitig. Alle Architekturen haben jedoch ein Nachteil : Die Binärprogramme sind nicht beliebig austauschbar, Shellscript dagegen schon. Weiterhin werden größere Resourcen (Programme) vom Fileserver im Netz geholt. Das /usr/local wird nun für die Architekturäbhängigen Programmdateien benutzt. Wie man das genau administriert siehe NFS-Guidebook von mir.

var Hier legt das System variable Daten ab. Dazu zählen Spooldateien zum Drucken, Mails, Logdateien, Systempaketinformationen. Und jede Menge mehr. Wichtig in diesem Zusammenhang ist das /var/run Verzeichnis. Dort hinterlassen meist die aktuellen laufenden Serverdienste ihre Spuren, damit diese wissen das sie laufen :-).

8.1.2 Zusammenfassung

Allgemein lässt sich sagen das der UNIX Dateibaum riesig groß ist. Das hängt auch damit zusammen, daß es unter UNIX keine *Laufwerksbuchstaben* gibt. Partitionen, Disketten, Netzwerklaufwerke, etc.pp. werden einfach in den UNIX Dateibaum mit eingehängt. Der Administrator oder der Benutzer merkt davon nichts. Wer weiss vielleicht befindet sich /usr/bin garnicht auf der Festplatte sonder kommt von einem Netzwerkserver. Keiner weiß es genau, dem Benutzer kann es schließlich egal sein, oder ?

Das ist auch der Grund einer gewissen Ordnung. Und wer Ordnung halten will muß ein Systemhaben um Sachen wiederzufinden bzw. zuzuorden. Daher ist es auch so groß. Den ganzen Verzeichnisbaum aufzuzeichnen währe Wahnsinn.

8.1.3 Absolute und Relative Dateinamen

Ein Absoluterdateiname wird immer von der Wurzel an beschrieben. Angenommen es existiert eine Datei mit dem Namen rwho im Verzeichnis /var/spool. Dann gibt es nun zwei Möglichkeiten auf diese

Datei zuzugreifen. Entweder Absolut von Oben wzB /var/spool/rwho. Die so gemachte Angabe ist immer gültig und es ist egal wo man sich im Verzeichnisbaum befindet.

Ein Relativerdateiname ist immer abhängig vom aktuellen Verzeichnis (CWD). Ist das aktuelle Verzeichnis z.B. /var/spool dann kann man auf rwho einfach mit rwho zugreifen. Der absolute Dateiname ist nicht notwendig. Angenommen rwho ist auch ein Verzeichnis und Sie befinden sich in diesem Verzeichnis. Ihr aktuelle Verzeichnis ist also /var/spool/rwho, um nun auf das Verzeichnis zuzugreifen kann man ein Punkt . verwenden. Jedes Verzeichnis hat als Dateinamen ein Punkt . der immer auf sich selbst zeigt. In diesem Fall also auf /var/spool/rwho.

Mal angenommen im rwho Verzeichnis befindet sich ein weiteres Verzeichnis mit dem Namen localnet und Sie wären in diesem Verzeichnis drin. Ihr CWD wäre also /var/spool/rwho/localnet. Um nun auf rwho zugreifen zu können, müssen Sie eine Verzeichnisebene höher. Dort kann dann der Punkt-Punkt .. verwendet werden. Hier muß also der Name .. angegeben werden.

8.2 Dateitypen und Rechte

8.2.1 Das Modewort

Jedes UNIX System speichert die Rechte und den Dateityp im Modewort. Das Modewort ist ein 16bit grosser Bereich in der INode zu dieser Datei. Eine INode werden wir später behandeln. Soviel sei aber gesagt : In einer INode werden die Eigenschaften einer Datei gespeichert, wie Grösse, Rechte, Zeitstempel, etc.pp. (siehe 19.4.1 auf Seite 96)

Die 16 bit sind wie folgt aufgeteilt. 9 Bits werden für Rechte reserviert. 3 Bits bekommen einen speziellen Touch in Verbindung mit den Rechten. Die übrigen 4 Bits sind für den Dateityp reserviert.

Die folgende Grafik zeigt dessen Aufbau.

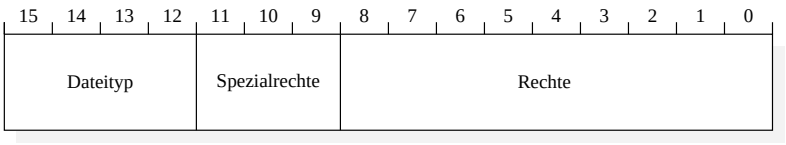


Abbildung 8.1: Grundaufbau des Modewortes

8.2.2 Dateitypen

Ein UNIX System entscheidet durch einige Bits um welchen Typ es sich bei einer Datei handelt. Und je nach Type sind Operationen auf die Datei erlaubt oder nicht.

Es gibt 4 Bits für den Dateityp. Damit können bis zu 16 verschiedene Dateitypen erzeugt werden (0-15). Einige Kombinationen sind jedoch nicht belegt und noch frei. Wir wollen uns mal die verschiedenen Typen anschauen

Es gibt :

Reguläre Dateien sind Dateien die einfach nur als Datendatei existieren. Sie können entweder Daten oder können auch ausführbare Programme sein. Das ist in einem UNIX System unwichtig.

Verzeichnis Datei sind spezielle Eintragungen die andeuten das es sich hiebei um Verzeichnisse handelt.

Zeichenorientierte Dateien sind Gerätedateien. Die Geräte arbeiten Zeichenorientiert. Dazu zählen Serielle und Parallele Schnittstellen, Streamer, Terminals

Blockorientierte Dateien sind Gerätedateien. Die geräte jedoch werden blockorientiert angesprochen und unterstützen keine zeichenorientierte Übertragung. Dazu zählen Festplatten, Wechselplatten, Disketten

Socket Dateien sind Netzwerkorientierte Dateien. Prozesse können mit diesen speziellen Dateien bidirektional (in beiden Richtungen) komunizieren. Der Ablauf ist ähnlich wie die Komunikation über das Netzwerk.

Fifo Dateien (Pipe) sind lokale Messagequeres die nach dem First-In-First-Out Prinziparbeiten. Auch hier können mehrere Prozesse eine solche Datei als Komunikationsweg wählen.

Softlink Dateien sind Verweise auf eine andere Datei. Somit wird ein Aliasname für eine Datei erstellt.

Diese ganzen Eigenschaften werden wie folgt in den Bits eingeordnet :

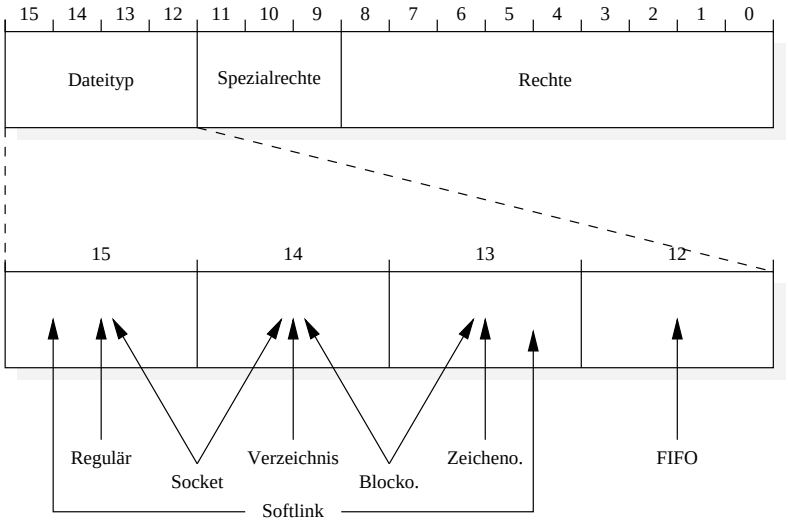


Abbildung 8.2: Binärkodierung der Dateitypen im Modewort

Die folgende Tabelle zeigt nochmal die Zusammenfassung :

Tabelle 8.1: Übersicht der Dateitypen Anzeigesymbole

Dateityp	Gesetzte Bits	`ls -l`	`ls -F`
Reguläre Datei	Bit 15	`-`	keine
Verzeichnis Datei	Bit 14	`d`	`/`
Zeichenorientierte Datei	Bit 13	`c`	keine
Blockorientierte Datei	Bit 14 und Bit 13	`b`	keine
Socket Datei	Bit 15 und Bit 14	`s`	`=`

Dateityp	Gesetzte Bits	ls -l	ls -F
Fifo Datei (Pipe)	Bit 12	p	|
Softlink	Bit 15 und Bit 13	l	@

8.2.3 Rechte

In einem UNIX System gibt es 3 Rechte. Die jenach kombination und auf den verwendeten Dateityp eine gewisse Bedeutung hat. Das Rechte System von UNIX System scheint gegenüber anderen Betriebsystemen einfacher zu sein. Das scheint jedoch nur so. In Wirklichkeit ist das Rechtesystem ohne die Hilfe von NIS oder NIS+ sehr schwierig zu Planen. (NIS und NIS+ kommt im Netzwerkkurs)

Es gibt 3 Rechtegruppierungen (im weiteren Sparten genannt um Verwechselungen zu vermeiden). Eine Sparte gilt für den Eigentümer eine weitere für die Gruppe eine weitere deckt den Rest der Welt ab. Jede dieser Sparten definiert genau 3 Rechte : Lesen (read), Schreiben (write) und Ausführen (execute). Das ergibt genau 9 mögliche Rechte. Plus die drei Sonderrechte.

Die folgende Grafik zeigt den Aufbau der Rechte im Modewort

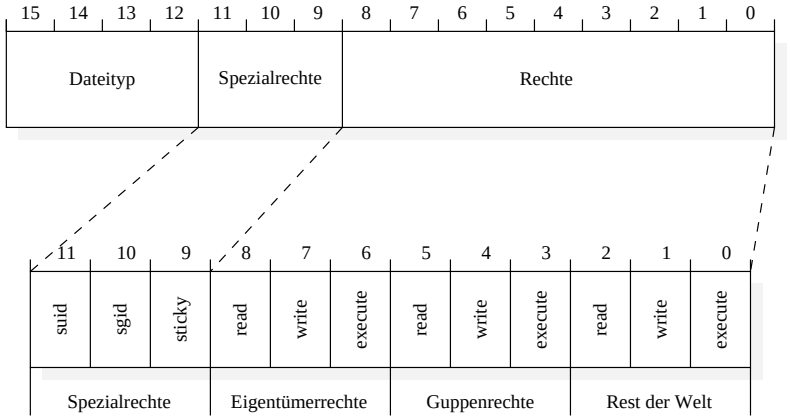


Abbildung 8.3: Binärkodierung der Rechte im Modewort

Allgemein kann man die Rechte wie folgt Beschreiben. Bei Kombinationen der Rechte jedoch können sehr interessante Effekte erzielt werden :

Tabelle 8.2: Übersicht der Auswirkungen der Lese-, Schreib- und Ausführrechte

Auswirkung auf Dateien	Auswirkungen auf Verzeichnisse
Keine Rechte - - -	
Kein Zugriff	Kein Zugriff
Nur Ausführen - - x	

▲

Auswirkung auf Dateien	Auswirkungen auf Verzeichnisse
Ist dieses Recht auf einer regulären Datei gesetzt, dann kann diese ausgeführt werden. Ähnlich wie .exe Dateien unter DOS. Bei Shellscripts muß das Leserecht gesetzt sein, ist es eine Binärdatei ist das Leserecht nicht notwendig. Bietet ein Kopierschutz für Binärdateien, diese können Ausgeführt werden aber nicht gelesen oder kopiert werden.	Dieses Recht erlaubt das Wechseln in ein Verzeichnis und den Aufruf der Dateien im inneren. Ist meist mit dem Leserecht zusammen gesetzt. Ist das Leserecht nicht gesetzt kann man den Inhalt des Verzeichnisses zwar nicht lesen, kennt man jedoch den Dateinamen kann man auf die Datei zugreiffen, sofern die Rechte der Datei Ihnen dieses erlauben.
Nur Schreiben `-w-`	
Dieses Recht erlaubt das Schreiben in eine Datei hinein. Jedoch wird damit nicht ausgesagt ob man eine Datei löschen bzw. erstellen kann. Das Screibrecht allein ist blödsinnig.	Dieses Recht auf ein Verzeichnis wird benötigt wenn man innerhalb des Verzeichnisses eine Datei löschen oder erstellen will. Man kann dort Dateien löschen die einem nicht gehören. Es reicht das Recht write auf dem Verzeichnis. Auch hier gilt : Steht es allein ist das blödsinnig
Nur Schreiben `-wx`	
Blödsinn	Blödsinn
Nur Lesen `r--`	
Dieses Recht erlaubt das Lesen des Dateiinhaltes. Ein solches Recht erlaubt auch die Kopie einer Datei. Shellscripts müssen dieses Recht gesetzt haben. Dieses Recht haben meist Systemkonfigurationsdateien und Datendateien die der Benutzer nicht ändern sollte, darf.	Dieses Recht erlaubt das Lesen der Verzeichnis Liste. Ohne Ausführrecht jedoch kann man nicht in das Verzeichnis wechseln, und auch kein Dateiinhalt dort lesen. Das Leserecht allein auf ein Verzeichnis ist ehr selten und ziemlich unüblich
Lesen und Ausführen `r-x`	
Dieses Recht erlaubt das Lesen des Dateiinhaltes. Ein solches Recht erlaubt auch die Kopie einer Datei. System Shellscripts haben meist dieses Recht gesetzt.	Dieses Recht erlaubt das Lesen der Verzeichnis Liste. Das Ausführrecht sorgt dafür das man auch hinein wechseln kann. Systemverzeichnisse haben meist dieses Recht damit der Benutzer dort nichts löschen kann.
Lesen und Schreiben `rw-`	
Dieses Recht erlaubt das Lesen des Dateiinhaltes. Ein solches Recht erlaubt auch die Kopie einer Datei. Shellscripts müssen dieses Recht gesetzt haben. Dieses Recht haben meist Systemkonfigurationsdateien und Datendateien die der Benutzer nicht ändern sollte, darf.	Dieses Recht erlaubt das Lesen der Verzeichnis Liste. Ohne Ausführrecht jedoch kann man nicht in das Verzeichnis wechseln, und auch kein Dateiinhalt dort lesen. Das Leserecht allein auf ein Verzeichnis ist ehr selten und ziemlich unüblich
Lesen, Schreiben und Ausführen `rwx`	

▼

▲

Auswirkung auf Dateien	Auswirkungen auf Verzeichnisse
Dieses Recht erlaubt das Lesen und Schreiben des Dateiinhaltes. Weiterhin kann man die Datei noch ausführen. Meistens haben Ihre Schellscripts dieses Recht	Erlaubt die Administration des Verzeichnisses, weil Lesen, Schreiben und das wechseln erlaubt sind. Meist hat Ihr Heimatverzeichnis dieses Recht.
Sonderechte	
Sticky Bit `--t`	
Ist das Ausführrecht für den Rest der Welt nicht gesetzt hat es keine Auswirkungen und ist auch falsch ! Bei älteren UNIX Systemen konnte man dem Kernel durch das Stickybit sagen, daß er den Hauptteil des Programms nicht aus dem Speicher entfernen sollte, auch dann wenn es beendet wurde. Führer waren die Festplatten sehr langsam und diese Methode war üblich. Heute ist dem nicht mehr so und deswegen hat das Stickybit heute keine Bedeutung mehr.	Ist das Ausführrecht für den Rest der Welt nicht gesetzt hat es keine Auswirkungen und ist auch falsch ! In diesem Verzeichnis dürfen die Benutzer ihre eigenen Dateien löschen, jedoch NUR ihre eigenen, selbst dann wenn der Benutzer das Schreibrecht auf das Verzeichnis besitzt. Dieses Recht ist grundsätzlich im temporären Verzeichnissen wzb `/tmp` gesetzt.
Set Group-ID `-s-`	
Ist das Ausführrecht für die Gruppe nicht gesetzt hat es keine Auswirkungen und ist auch falsch ! Wenn das Programm aufgerufen wird ändert sich die effektive Zugriffsgruppenangehörigkeit des Prozesses. Der Prozeß arbeitet danach mit der Gruppenkennung dessen die Datei gehört und nicht mehr mit Ihrer. Wird meist bei dem gesamten Druckbefehlen gemacht. Dort haben die Programme die Gruppe `lp` und das SGID Bit gesetzt. Somit wird der Befehl effektiv mit der Gruppe `lp` ausgeführt, und die hat Zugriff auf die Konfigurationsdateien. Ein normaler Benutzer hat kein Zugriff auf die Dateien	Ist das Ausführrecht für die Gruppe nicht gesetzt hat es keine Auswirkungen und ist auch falsch ! Werden in dieses Verzeichnis Dateien neu angelegt, ist es egal welcher Gruppe Sie angehören, die Dateien und Verzeichnisse bekommen die Gruppenangehörigkeit wie auch das Verzeichnis selbst. Wird meist bei Verzeichnissen gesetzt um einigen Benutzern ein Sharing an zubieten.
Set User-ID `s--`	
Ist das Ausführrecht für den Benutzer nicht gesetzt hat es keine Auswirkungen und ist auch falsch ! Siehe SGID Bit jedoch nur jetzt für die Benutzerkennung. Wird meist für Administrationstools benötigt.	Ist das Ausführrecht für den Benutzer nicht gesetzt hat es keine Auswirkungen und ist auch falsch ! Soweit mir bekannt ist hat das keine Auswirkungen.

8.2.4 Numerische Notierung von Rechten

Rechte werden im allgemeinen auch als Zahlen dargestellt. Dazu wird die Binärkombination im Octalen System wiedergegeben. Wobei für jede Rechte Sparte genau eine Ziffer reserviert ist. Die folgende Tabelle zeigt eine Übersicht aller möglichen Rechtekombinationen:

Tabelle 8.3: Übersicht der numerischen Schreibweise von Rechten

Rechtekombination	Binärkombination	Oktale Ziffer
—	`000`	0
–x	`001`	1
-w-	`010`	2
-wx	`011`	3
r–	`100`	4
r-x	`101`	5
rw-	`110`	6
rwx	`111`	7
sticky	`001`	1
sgid	`010`	2
suid	`100`	4

Somit kann man ein Recht wie `4751` auch wie folgt umschreiben: Der Eigentümer besitzt Lese, Schreib und Ausführrecht, die Gruppe bekommt Lese und Ausführrecht und alle anderen bekommen nur Ausführrecht und durch das SUID bekommt der Prozess die Rechte des Eigentümers

8.2.5 Standard Rechte neuer Dateien/Verzeichnisse

Wenn eine Datei erstellt wird, gilt folgendes:

Verzeichnisse werden immer mit dem Recht `rwxrwxrwx / 777` erstellt

Dateien werden immer mit dem Recht `rw-rw-rw- / 666` erstellt

Mit `umask` kann dieses Verhalten geändert werden. Man gibt dort an welche Rechte gefiltert werden sollen. `umask` erwartet die Angabe der Okalzahl. Wird diese auf z.b. `027` gesetzt, bekommen Verzeichnisse nur noch `rwxr-x--- / 750` und Dateien nur noch `rw-r----- / 640` rechte.

`umask` wirkt sich nur auf neu erstellte Dateien / Verzeichnisse aus ! Eine Trennung zwischen Datei und Verzeichniss kann nicht stattfinden.

8.3 Befehle zur Datei/Verzeichnis Verwaltung

Das ganze beschriebene kann natürlich auch entsprechend verwaltet werden. Wenden wir uns dazu ersteinmal den Verzeichnissen zu

8.3.1 Befehle zur Verzeichnisverwaltung

pwd Mit `pwd`[2] *(Print Working Directory)* kann festgestellt werden in welchem Verzeichnis man sich aktuell befindet.

cd Mit `cd`[3] *(Change Directory)* kann ein Verzeichnis gewechselt werden. Das kann entweder im absoluter oder relativer Form passieren.

ls Mit `ls`[4] *(List)* kann der Inhalt eines Verzeichnisses aufgelistet werden. Ohne Angabe von Parametern listet `ls` das aktuelle Verzeichnis auf. Ansonsten versucht es die Verzeichnisse oder Dateien aus den Argumenten auf zulisten. Verwenden Sie `ls -l` um eine lange Ausgabe zu bekommen und `ls -a` um auch die versteckten Dateien aufzulisten. Beides kann kombiniert werden `ls -la` z.B.

mkdir Mit `mkdir`[5] *(Make Directory)* können Verzeichnisse angelegt werden. Das zu erstellende Verzeichnis kann entweder absolut oder relativ angegeben werden, es können auch mehrere gleichzeitig erstellt werden

rmdir Mit `rmdir`[6] *(Remove Directory)* können Verzeichnisse entfernt werden. Jedoch nur leere Verzeichnisse. Auch hier gilt: entweder absolut oder relativ, auch mehrere können angegeben werden

8.3.2 Befehle zur Dateiverwaltung

file Mit `file`[7] *(Filetype)* kann der Dateityp ermittelt werden. Remember: UNIX kennt die Unterscheidungen mittels Suffix nicht

type Mit `type`[8] *(Type)* kann man feststellen wo sich ein ausführbares Programm befindet, sofern es sich im Suchpath befindet. Oder ob es sich um eine Funktion oder ein Aliasnamen handelt.

which Mit `which`[9] *(Which)* kann man feststellen wo sich ein ausführbares Programm befindet, sofern es sich im Suchpath befindet. Ohne Prüfung der Funktionen etc.pp.

cat Mit `cat`[10] *(Concatenate)* kann man sich den Inhalt einer regulären Datei anschauen

more Mit `more`[11] *(More)* kann man sich die Datei auch seitenweise anschauen. `more` kann auch in Pipes eingebaut werden. z.B. `ps | more` oder `ls | more`. Zum Blättern einfach ein SPACE drücken und ein 'q' zum Verlassen der Anzeige

[2] `pwd` - Siehe Kommandoreferenz Seite 365
[3] `cd` - Siehe Kommandoreferenz Seite 253
[4] `ls` - Siehe Kommandoreferenz Seite 319
[5] `mkdir` - Siehe Kommandoreferenz Seite 327
[6] `rmdir` - Siehe Kommandoreferenz Seite 377
[7] `file` - Siehe Kommandoreferenz Seite 275
[8] `type` - Siehe Kommandoreferenz Seite 401
[9] `which` - Siehe Kommandoreferenz Seite 407
[10] `cat` - Siehe Kommandoreferenz Seite 251
[11] `more` - Siehe Kommandoreferenz Seite 329

find Mit `find`[12] können Dateien verschiedenster Arten im Dateibaum gesucht werden. Man kann nach Dateinamen oder Rechten oder Groesse suchen.

cp Mit `cp`[13] *(Copy)* können Dateien und Verzeichnisse kopiert werden. `cp` erwartet immer mindestens 2 Parameter. Zum einen 1 Quelldatei und den Namen der Zieldatei. Ist das Ziel jedoch ein Verzeichnis, dann wird die Datei in das Verzeichnis kopiert unter gleichem Namen. Somit können $1 - n$ Quelldateien aufgelistet werden, dann jedoch muß der letzte Parameter ein Verzeichnis sein, dort hin werden dann die ganzen Quelldateien kopiert

rm Mit `rm`[14] *(Remove)* können Dateien und Verzeichnisse inkl. Unterdateinen gelöscht werden. Ein Mülleimer gibt es nicht

mv Mit `mv`[15] *(Move)* können Dateien bewegt/umbenannt werden. Hier gilt das gleiche Verhalten wie bei `cp` wobei die Originaldatei nach der Kopie gelöscht wird

8.3.3 Befehle zur Rechteverwaltung

chmod Mit `chmod`[16] *(Change Mode)* können die Rechte und Spezialrechte geändert werden. Nur der Eigentümer oder der Administrator kann diese Aktion ausführen.

chgrp Mit `chgrp`[17] *(Change Group ID)* kann die Gruppenzugehörigkeit einer Datei geändert werden. Nur der Eigentümer oder der Administrator kann diese Aktion ausführen.

chown Mit `chown`[18] *(Change Owner ID)* kann der Eigentümer einer Datei geändert werden. Vorsicht: ein Benutzer kann sich so den Boden unter den Füssen wegziehen. Nur der Eigentümer oder der Administrator kann diese Aktion ausführen.

umask Mit `umask`[19] *(Un-Mask)* kann man den Filter der Rechte für neue Datenen / Verzeichnisse einstellen. Jeder Benutzer kann die wählen wie er will.

[12] `find` - Siehe Kommandoreferenz Seite 277
[13] `cp` - Siehe Kommandoreferenz Seite 263
[14] `rm` - Siehe Kommandoreferenz Seite 375
[15] `mv` - Siehe Kommandoreferenz Seite 331
[16] `chmod` - Siehe Kommandoreferenz Seite 259
[17] `chgrp` - Siehe Kommandoreferenz Seite 257
[18] `chown` - Siehe Kommandoreferenz Seite 261
[19] `umask` - Siehe Kommandoreferenz Seite 403

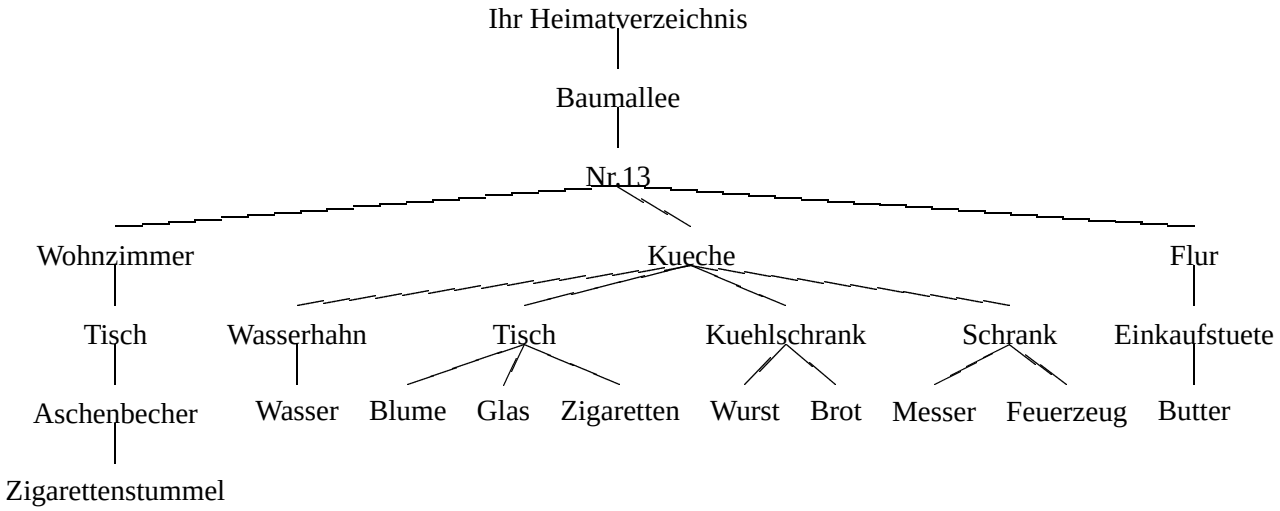
8.4 Praktikum

8.4.1 Umgang mit der Verzeichnisstruktur

Diese Uebungsaufgaben vertiefen die Verwendung der Programme cp, mkdir, rmdir, mv, ls, cd ... etc.pp. Bitte denken Sie bei der Uebung daran dass Leerzeichen im Dateinamen unueblich sind und nicht verwendet werden sollten Ziel der Uebungen soll ein Haus als VZ-Struktur sein.

1. Wechseln Sie in Ihr Heimatverzeichnis
2. Erstellen Sie ein Verzeichnis mit dem Namen 'Baumallee'
3. Erstellen Sie in diesem Verzeichnis ein weiteres VZ mit dem Namen 'Nr.13' und wechseln Sie in dieses VZ hinein
4. Erstellen Sie ein VZ fuer das Wohnzimmer mit dem Namen 'Wohnzimmer'
   (a) In dem Wohnzimmer befindet sich ein Tisch, also erstellen Sie ein VZ fuer den Tisch
   (b) Auf dem Tisch befindet sich ein Aschenbecher, also erstellen Sie im 'Tisch' VZ ein weiteres VZ mit dem Namen 'Aschenbecher'
   (c) Im Aschenbecher befinden sich Zigarettenstummel. Dafuer erstellen Sie eine Datei mit dem Namen 'Zigarettenstummel' im Aschenbecher VZ. (Dateierstellen mit $ touch ¡dateiname¿)
5. Erstellen Sie in Ihrem Haus noch ein VZ fuer die Kueche (VZ)
   (a) In der Kueche befindet sich ein Kuehlschrank (VZ)
      i. Im Kuehlschrank liegt eine Wurst (Datei)
      ii. Weiterhin befindet sich im Kuehlschrank ein Brot (Datei)
   (b) In der Kueche steht auch ein Tisch (VZ)
      i. Auf dem Tisch steht eine Blume (Datei)
      ii. Und eine Packung Zigaretten (Datei)
      iii. Und ein Glas (VZ)
   (c) In der Kueche ist noch ein Schrank (VZ) in dem sich ein Feuerzeug (Datei) und ein Messer (Datei) befindet.
   (d) Meist befindet sich in der Kueche auch ein Wasserhahn (VZ) mit Wasser (Datei)
6. Im Flur (VZ) steht eine Einkaufstuete (VZ) mit Butter (Datei).

Nach der Übung sollten Sie folgenden Verzeichnisbaum haben :



Und so gehts weiter :

1. Sie kommen abends nach Hause und wollen kurz noch etwas Essen. Sie nehmen also die Butter und stellen diese auf den Tisch in der Kueche (Datei verschieben !)
2. Weiterhin nehmen Sie sich noch die Wurst und das Brot aus dem Kuehlschrank und stellen es auf den Tisch
3. Weiterhin bentigen Sie noch das Messer
4. Sie essen und bekommen Durst. Also fllen Sie das Glas mit Wasser (Datei kopieren !)
5. Da Sie das ganze Glas leertrinken ist es leer.
6. Sie stellen alles wieder in den Kuehlschrank zurueck. Auch das Messer. Dann muessen Sie es naechstes mal nicht wieder suchen
7. Jetzt sind Sie zufrieden und wollen im Wohnzimmer gemuetlich beim TV eine Rauchen. Dazu ist das Rauch-Werkzeug auf den Tisch im Wohnzimmer zu stellen.
8. Leider ist der Aschenbecher so voll, dass keine Kippe mehr reinpassen wuerde. Also schuetten Sie den Aschenbecher in die mitgebrachte Einkaufstuete.
9. Sie schlafen ein und die Zigarette verbrennt den ganzen Tisch. Das war dumm. In der Wohnung koennen Sie nicht laenger wohnen und ziehen in das Haus mit der Nummer 34 (VZ) in der gleichen Strasse um.
10. Sie nehmen Ihre Gertschaften aus der Kueche und den Flur mit (Verschieben !)
11. Den Wasserhahn sollten Sie jedoch Aufgrund der Installation nicht mitnehmen
12. Sie sind Umgezogen ! Das Haus mit der Nummer 13 explodiert danach und Sie sind gluecklich

Kapitel 9

JOBCONTROL

Das Jobcontrol wird durch die Bourne Shell implementiert und dient zur Kontrolle von Jobs (Prozessen). Da UNIX ein Multitastking System ist muß man auch von der Shell aus mehrere Prozesse gefahrlos in den Hintergrund bringen können. Sonst wäre es ja auch witzlos. Die Bourne Shell muß jedoch mit `/bin/jsh` gestartet werden!

9.1 Hintergrundprozesse

Um in einer Shell ein Prozess in den Hintergrund zu befördern kann das besondere Zeichen & (Ampersant) der Shell verwendet werden. Der Prozess geht in den Hintergrund und arbeitet dort einfach weiter. Der Prozess jedoch stellt seine Ausgabe auf das aktuelle Terminal, welches dann zur scheinbaren Zerstörung des Bildschirmes fuehren kann jedoch ist das nicht weiter Tragisch. Erwartet der Prozess im Hintergrund eine Eingabe via Tastatur dann wird es kompliziert. Wird eine Taste gedrückt wird die entwender den Vordergrund oder dem Hintergrundprozess zugewiesen, das kann dann keiner mehr so genau sagen.

Ein Hintergrundprozess besitzt eine PID genauso wie andere Prozesse auch. Man kann also Hintergrundprozesse via `kill`[1] entsprechend beseitigen. Weiterhin bekommt der Prozess eine Job-ID von der Shell. Beide Werte werden nach dem Start entsprechend angezeigt.

```
$ find /home/whurst -name jobcontrol >/tmp/find-log &
[1] 814
$ ps
   PID TTY      TIME CMD
   814 pts/6    0:00 find
   810 pts/6    0:00 sh
$ .......... etwas spaeter ...........
[1]+  Done    find /home/whurst -name jobcontrol >/tmp/find-log
$ ps
   PID TTY      TIME CMD
   810 pts/6    0:00 sh
$ cat /tmp/find-log
/home/whurst/GUIDE/unix/jobcontrol.tex
$
```

Bildschirmausschnitt 9.1.1: Anwendungsbeispiel für Hintergrundprozesse

In diesem Beispiel wird `find`[2] in den Hintergrund gebracht und die Ausgabe in einer Datei gespeichert, damit diese sich nicht mit den Prozessen die im Vordergrund ablaufen vermischt.

[1] `kill` - Siehe Kommandoreferenz Seite 301

[2] `find` - Siehe Kommandoreferenz Seite 277

Prozesse die durch eine Shell in den Hintergrund gebracht werden, werden beim Verlassen der Shell abgebrochen und beendet. Damit der Befehl das Überlebt mu `nohup`[3] verwendet werden. Damit ist es dann auch möglich, das Sie sich einloggen, ein Prozess starten, sich Ausloggen und nach Hause gehen. Kommen Sie dann morgen wieder ist die Aufgabe erledigt. Weil der Prozess das beenden der Shell überlebt hat.

```
$ nohup find / -name irgendwas >/tmp/find-log &
[1] 833
$ ps
   PID TTY      TIME CMD
   833 pts/6    0:00 find
   810 pts/6    0:00 sh
$ exit

raven login: whurst
password:

$ ps
   PID TTY      TIME CMD
   854 pts/5    0:00 sh
$ ps -e -u whurst
   PID TTY      TIME CMD
   833 ?        0:12 find
   854 pts/5    0:00 sh
$ kill 833
$ ps -e -u whurst
   PID TTY      TIME CMD
   854 pts/5    0:00 sh
$
```

Bildschirmausschnitt 9.1.2: Anwendungsbeispiel für NOHUP Hintergrundprozesse

Man sieht nach dem Login hat der nohup Prozess keine Terminalleitung mehr und wird mit einem einfachen `ps`[4] nicht mehr angezeigt.

9.2 Stoppen von Vordergrundprozessen

Um den aktuell laufenden Prozess zu stoppen, muß man ein Control-Zeichen eingeben. Dieses Zeichen jedoch ist nicht überall das gleiche. Um das entsprechende Zeichen rauszubekommen kann `stty`[5] mit der Option *-a* verwendet werden. Meist jedoch ist es CTRL-Z. Mit der Kombination kann ein aktueller Vordergrundprozeß unterbrochen werden. Der Prozeß wird vom Kernel blockiert und die Shell meldet sich zurück. Mit `jobs`[6] können nun alle Jobs angezeigt werden.

[3] `nohup` - Siehe Kommandoreferenz Seite **??**
[4] `ps` - Siehe Kommandoreferenz Seite **??**
[5] `stty` - Siehe Kommandoreferenz Seite 391
[6] `jobs` - Siehe Kommandoreferenz Seite 299

```
$ sleep 1000
^Z
[1]+  Stopped                 sleep 1000
$ jobs
[1]+  Stopped                 sleep 1000
$ sleep 2000
^Z
[2]+  Stopped                 sleep 2000
$ jobs
[1]-  Stopped                 sleep 1000
[2]+  Stopped                 sleep 2000
$
```

Bildschirmausschnitt 9.2.1: Anwendungsbeispiel zum Stoppen

9.3 Jobcontrol

Um die einzelnen Jobs zu kontrollieren können die Befehle `bg`[7] und `fg`[8] verwendet werden. `bg` setzt ein Job wieder in den Hintergrund und mit `fg` wird ein Hintergrundprozess zum Vordergrundprozeß.

[7] `bg` - Siehe Kommandoreferenz Seite 249
[8] `fg` - Siehe Kommandoreferenz Seite 273

```
$ jobs
[1]-  Stopped                 sleep 1000
[2]+  Stopped                 sleep 2000
$ bg 1
[1]- sleep 1000 &
$ ps
   PID TTY      TIME CMD
   863 pts/7    0:00 sleep
   838 pts/7    0:00 sh
   864 pts/7    0:00 sleep
$ ps -f
     UID   PID  PPID  C    STIME TTY      TIME CMD
  whurst   863   838  0 12:12:09 pts/7    0:00 sleep 1000
  whurst   838   836  0 11:49:50 pts/7    0:00 /bin/sh
  whurst   864   838  0 12:12:26 pts/7    0:00 sleep 2000
$ jobs
[1]-  Running                 sleep 1000 &
[2]+  Stopped                 sleep 2000
$ fg 1
sleep 1000
$ jobs
[2]+  Stopped                 sleep 2000
$
```

Bildschirmausschnitt 9.3.1: Anwendungsbeispiele zur Kontrolle

9.4 Zusammenfassung

Tabelle 9.1: Jobcontrol

Programm / Datei	**Bedeutung**
&	Stellt ein Prozeß in den Hintergrund
nohup[9]	Stellt den Prozeß in den Hintergrund ohne diesen zu Verlassen
jobs[10]	Anzeige der laufenden bzw. gestoppten Jobs der Shell
fg[11]	Stellt den Job wieder in den Vordergrund
bg[12]	Stellt den Prozeß in den Hintergrund

[9] nohup - Siehe Kommandoreferenz Seite **??**
[10] jobs - Siehe Kommandoreferenz Seite 299
[11] fg - Siehe Kommandoreferenz Seite 273
[12] bg - Siehe Kommandoreferenz Seite 249

Kapitel 10

FILTERPROGRAMME

Ein Filterprogramm ist ein Programm der eine Eingabedatei umwandelt um eine neue Ausgabedatei zu erstellen. Unter UNIX ist fast jedes Programm ein solcher Filter. Filterprogramme werden meist in Pipe-Konstuktionen eingesetzt.

10.1 Pipes

Eine Pipekonstruktion wird von der Shell ausgeführt. Als besonderes Zeichen kommt der senkrechte Balken — zum Einsatz. Den man auch Pipe nennt.

Fast jedes UNIX Programm gibt seine Ergebnisse auf dem stdout Kanal aus, andere Programme können auch von stdin lesen. Genau das macht sich die Pipe zunutze und verbindet zwei Programme miteinander. Eine Pipe hat immer zwei Programme die wie folgt bezeichnet werden :

```
<LHS Program> | <RHS Program>
```

Die Pipe verknüpft das LHS[1] Programm und leitet dessen Ausgabe vom stdout in den stdin Kanal des RHS[2].

10.1.1 Einfaches Beispiel

Als ganz simples Beispiel soll mal das Programm head (siehe 41 auf Seite 287) herhalten. Benutzt man das Programm head mit einem Dateinamen, dann verwendet head den Inhalt der Datei um die ersten 10 Zeilen anzuzeigen :

```
$ head /etc/passwd
root:x:0:1:Super-User:/root:/bin/bash
daemon:x:1:1::/:
bin:x:2:2::/usr/bin:
sys:x:3:3::/:
adm:x:4:4:Admin:/var/adm:
lp:x:71:8:Line Printer Admin:/usr/spool/lp:
uucp:x:5:5:uucp Admin:/usr/lib/uucp:
nuucp:x:9:9:uucp Admin:/var/spool/uucppublic:/usr/lib/uucp/uucico
ftp:x:21:21:Anonymous FTP Account:/var/ftp:/bin/false
listen:x:37:4:Network Admin:/usr/net/nls:
$
```

Bildschirmausschnitt 10.1.1: Einfaches Pipe Beispiel mit head und einem Dateinamen

[1] LHS-Left Hand Side
[2] RHS-Right Hand Side

Wird jedoch einem Filterprogramm, wie `head` eines ist, kein Dateinamen übergeben, dann lesen alle Filterprogramme aus dem `stdin` Kanal. Ruft man ein Filter ohne Dateinamen auf das geht der Cursor einfach nur in die nächste Zeile und bleibt dort. Es entsteht der Eindruck als wenn der Task abgemiert sei - Nein! Der Prozess liest von Standard-Eingabe und die ist im normalfall mit der Tastatur verbunden. Der Filter wartet bis das Ende der Datei eintrifft um dann entsprechend zu handeln. Das EOF[3] kann mittels CTRL-D simuliert werden. Das folgende Beispiel zeigts :

```
$ head
Das
habe
ich
gerade
auf
der
Tastatur
ein-
gegeben
.
Weil head
nun von
der Tastatur liest
^DDas
habe
ich
gerade
auf
der
Tastatur
ein-
gegeben
.
$
```

Bildschirmausschnitt 10.1.2: Einfaches Pipe Beispiel mit `head` und ohne einem Dateinamen

Eine Pipe verbiegt nun die Standardbelegung von `stdin` und als Eingabekanal dient nun der Ausgabekanal eines anderen Programms. Im folgenden Beispiel dient das Programm `cat` (siehe 41 auf Seite 251) als LHS :

[3] `EOF`-End of File

```
$ cat /etc/passwd | head
root:x:0:1:Super-User:/root:/bin/bash
daemon:x:1:1::/:
bin:x:2:2::/usr/bin:
sys:x:3:3::/:
adm:x:4:4:Admin:/var/adm:
lp:x:71:8:Line Printer Admin:/usr/spool/lp:
uucp:x:5:5:uucp Admin:/usr/lib/uucp:
nuucp:x:9:9:uucp Admin:/var/spool/uucppublic:/usr/lib/uucp/uucico
ftp:x:21:21:Anonymous FTP Account:/var/ftp:/bin/false
listen:x:37:4:Network Admin:/usr/net/nls:
$
```

Bildschirmausschnitt 10.1.3: Einfaches Pipe Beispiel mit `head` und `cat`

Das RHS darf jetzt nicht mehr mit einem Dateinamen aufgerufen werden. Weil sonst hat man eine **Broken Pipe**. Das Passiert immer wenn das RHS Programm nicht von stdin liest, aber ein LHS Programm Daten auf stdout gibt und die beiden sind mittels Pipe verbunden.

Unter Solaris 8 wird keine Fehlermeldung 'Broken Pipe' ausgegeben

Das folgende Beispiel zeigt ein solches Unglück :

```
$ cat /etc/passwd | head /etc/group
root::0:root
other::1:
bin::2:root,bin,daemon
sys::3:root,bin,sys,adm
adm::4:root,adm,daemon
uucp::5:root,uucp
mail::6:root
tty::7:root,tty,adm
lp::8:root,lp,adm
nuucp::9:root,nuucp
$
```

Bildschirmausschnitt 10.1.4: Einfaches Pipe Beispiel mit einer doppeldeutigen Bedeutung

Der Datenstrom vom `cat` wird einfach versenkt.

10.2 Einfache Filterprogramme

head, tail, more, less, grep

10.3 Datenstrom veränderen Filterprogramme

sort, tr, uniq,

10.4 Datenstrom vernichtene Filterprogramme

wc

Kapitel 11

VI

Der *vi* ist eine Waffe !

Warum der *vi* ? Nun früher noch zu Terminalzeiten gab es ein zeilenorientierten Editor mit dem Namen ex. Dieser funktionierte ähnlich wie der edlin unter MS-DOS. Früher gab es tausende von verschiedenen Terminalemulationen und tausende von unterschiedlichen Betriebssystemen. Der Gedanke war einfach : Wir wollen ein Editor haben der auf allen Betriebssystemen gleich funktioniert. Das Ergebnis war der ex. Der ex kann den Text immer nur zeilenweise Anzeigen und auch die Editierung war immer nur Zeilenweise möglich. Irgendwann wurden die Leitungen schneller und man schrieb den ex so um das man den Text im Vollbild hatte, so wie heute jeder andere Editor. Aber ein war Bedingung : Er muss auf allen Betriebssystemen vorhanden sein. Und so schieb man den *vi* und implementierte Ihn in den UNIX Standard.

D.h. der *vi* ist auf **ALLEN UNIX Derivaten** vorhanden.

Der *vi* funktioniert überall gleich !

11.1 Gewöhnungsbedürftig

Jedes Mal wenn Sie eine neue Software bekommen und wollen diese Testen, muessen Sie sich umgewöhnen. Angenommen Sie kennen MS-Office auswendig, muessen jetzt mit ApplixWare arbeiten oder bekommen eine neue Version des MS-Offices.

Um den Editor *vi* zu verstehen, muessen Sie alle anderen Editoren vergessen. Ihnen wird nach einer weile auffallen, daß Sie mit dem *vi* wesendlich schneller, effektiver Arbeiten koennen als mit jeden anderen Editor. Ich will garnicht bestreiten das der Editor leicht sei, aber ist MS-Office etwa einfacher ?

11.2 Die 3 Modies

11.2.1 Der Kommandomodus

Cursorbewegungen innerhalb des Textes

Kopieren und Löschen von Texten

11.2.2 Der ex-Modus

Dateien Operationen

Suchen von Textstellen

11.2.3 Der Eingabemodus

11.3 Einstellungsmöglichkeiten mit `:set`

11.4 Macros mit `:map`

11.5 *vi* Klones

11.5.1 gvim

11.6 Advanced Setup

TEIL II

SHELLPROGRAMMIERUNG

- Environment einer Shell
- Funktionen
- Das erste Shellscript
- Kontrollstrukturen
- Pipe Konstruktionen mit Controllstrukturen

Kapitel 12

ENVIRONMENT EINER SHELL

Als Environment wird die Umgebung einer Shell bezeichnet. In einem Environment können Informationen jeder beliebigen Art und Weise abgelegt werden. Viele Programme kann man mittels Environment konfigurieren. Auch in der Shellprogrammierung wird das Environment genutzt um dort diverse Informationen abzulegen.

12.1 Aufbau einer Environmentvariablen

In einem Environment kann man einer Variablen einen Wert zuweisen. Die Variable selbst hat irgendeinen Namen. Der Wert kann beliebig sein. Die Shell macht dabei keine Unterschiede ob es sich um eine Zahl oder eine Zeichenkette handet. Die Zuweisung geschieht einfach durch das Gleichheitszeichen und besitzt folgende Syntax :

```
<varname>=<wert>
```

12.1.1 Zuweisungen von Werten

Um einer Variablen ein Wert zuzuweisen kann man nun einfach auf der Kommandozeile wie folgt eingeben :

```
$ Var1=Irgendein
$ Var2=Text
```

Bildschirmausschnitt 12.1.1: Beispiel einer einfachen Wertzuweisung an eine Environmentvariable

Wenn man jedoch Leerzeichen eingeben will, bekommt man Schwierigkeiten mit der Shell, Sie errinnern sich bestimmt das das Leerzeichen ein besonderes Zeichen der Shell ist und als Argumenttrenner benutzt wird. Um ein Leerzeichen in eine Variable zu Transportieren muß es vor der Shell versteckt werden :

```
$ Leer1=Irgendein\ Text
$ Leer2="Irgendein          Text"
```

Bildschirmausschnitt 12.1.2: Beispiel einer Wertzuweisung mit Leerzeichen an eine Environmentvariable

12.1.2 Auslesen von Variablen

Um auf die Inhalte der Variablen zuzugreifen muß man der Shell durch ein besonderes Zeichen dieses verlangen auch mitteilen. Die Shell verwendet hier das einfache Dollar Zeichen, gefolgt von dem Variablennamen, welches etwa folgender Syntax entspricht :

```
$<varname>
```

Den Inhalt der Variablen kann man nun überall in der Kommandozeile einsetzen lassen, wie die folgenden Beispiele zeigen :

```
$ echo $Leer1
Irgendein Text
$ echo $Var1 $Var2
Irgendein Text
$ lsopt=-la
$ ls $lsopt
total 12
drwxr-xr-x   2 whurst   whurst      1024 Apr 17 20:47 ./
drwxr-xr-x  80 whurst   whurst      5120 Apr 24 19:29 ../
```

Bildschirmausschnitt 12.1.3: Beispiel der Verwendung von Variableninhalten

Heimtückisch wird es dann wenn in einer Variablen Leerzeichen oder sonstige besonderen Zeichen der Shell gepeichert sind. Das kann unter umständen zu sehr grossen Katastrophen führen. Das folgende Beispiel zeigt es einmal falsch und einmal richtig :

```
$ echo $Leer2
Irgendein Text
$ echo "$Leer2"
Irgendein          Text
$ echo '$Leer2'
$Leer2
```

Bildschirmausschnitt 12.1.4: Beispiel der Verwendung von Variableninhalten mit Leerzeichen

Nun warum druckt das erste echo die Leerzeichen nicht mit. Eigendlich ganz simpel : Die Shell bekommt die Eingabezeile `echo $Leer2` sie entdeckt dort ein Dollarzeichen mit einem Variablennamen dahinter. Also ersetzt sie das Argument mit dem Inhalt von `Leer2`. Danach hat die Shell folgenden Ausdruck im Speicher : `echo␣Irgendein␣␣␣␣␣␣␣␣␣␣Text`. Nachdem die Shell merkt das es nichts mehr zum Ausflösen gibt ruft sie das Programm echo auf und übergibt dem echo zwei Parameter. Die Leerzeichen gelten ja als Argumenttrenner. Und schon ist es passiert, echo gibt alle Argumente der Reihe nach durch ein Leerzeichen aus.

Im zweiten Beispiel Passiert folgendes: Die Shell bekommt `echo "$Leer2"`. Da die doppelten Anführungszeichen die besondere Bedeutung des Dollars nicht aufheben ersetzt die Shell wieder den Ausdruck mit dem Inhalt. Somit bekommt sie folgende Zeile in den Speicher `echo␣"Irgendein␣␣␣␣␣␣␣␣␣␣Text"`. Jetzt ruft sie das Programm echo auf. Aber echo bekommt jetzt nur noch ein Argument und nicht zwei ! Und so funktionierts.

Ja beim dritten wird der Shell das Dollar versteckt und sie sieht es nicht.

12.1.3 Gültigkeitsbereiche von Environmentvariablen

Eine definierte Variable ist immer nur in der eigenen Shell gültig. Definiert man eine Variable innerhalb einer Shell und ruft ein anderes Programm auf, ist die Variable nicht mehr verfügbar für das Programm. Verlässt man das Programm wieder und kehrt zur alten Shell wieder zurück ist die Variable wieder da.

```
$ Var1=test
$ echo $Var1
test
$ sh
$ echo $Var1

$ exit
$ echo $Var1
test
$
```

Bildschirmausschnitt 12.1.5: Variablengültigkeitsbereich: Lokale Shell

Wird in dem neu aufgerufenen Programm die Variable überschrieben, hat das keine Auswirkungen auf die Variable der alten Shell :

```
$ Var1=test
$ sh
$ Var1=notests
$ exit
$ echo $Var1
test
$
```

Bildschirmausschnitt 12.1.6: Variablengültigkeitsbereich: Lokale Shell mit Überschreiben

Um eine Variable über die lokale Shell hinaus zu tragen muß diese exportiert werden. Um das zu erreichen gibt es den Befehl `export` (siehe 41 auf Seite 271)

```
$ Var1=test
$ export Var1
$ sh
$ echo $Var1
test
$ Var1=notests
$ exit
$ echo $Var1
test
$
```

Bildschirmausschnitt 12.1.7: Variablengültigkeitsbereich: Exportierte Variablen

Wie man hier sieht bringt das Überschreiben nicht viel. Die alte Shell besitzt immer noch den alten Wert. Eine Exportierung muß nur einmal gesetzt werden und vererbt sich in die anderen Programme hinein.

```
$ Var1=test
$ export Var1
$ sh
$ sh
$ sh
$ echo $Var1
test
$
```

Bildschirmausschnitt 12.1.8: Variablengültigkeitsbereich: Exportvererbung

12.2 Standard und System Variablen

Diese Art von Variablen werden meist vom Administrator vorbelegt, der Benutzer kann diese entsprechend erweitern, einige anderen sind für das System und für einige andere Programm hoch wichtig. Es folgt nun eine zwanglose Auflistung einiger Wichtigen Variablen. Spezielle Variablen können in der Referenz nachgeschlagen werden.

HOME In dieser Variablen wird das Heimatverzeichnis gespeichert. Wird der Befehl `cd` (siehe 41 auf Seite 253) ohne Parameter aufgerufen, wechselt `cd` dort in dieses Verzeichnis welches in `HOME` sich befindet

IFS Im IFS[1] werden die Trennzeichen gepeichert die die Shell zur Trennung benutzt. Werden innerhalb der Shellprogrammierung Werte eingelesen, dann werden die Werte dort in einzelne Argumente zerlegt. In der `IFS` stehen die Zeichen die die Shell dann als Trennungsmarkierer benutzt. Standard `IFS` ist ein Leerzeichen ein Tabulator und das Returnzeichen.

LANG In dieser Variablen wird die Sprachumgebung eingestellt. Fast alle Programme besitzen unterhalb von `/usr/lib/locale` eigene Übersetzungstabellen. Wenn nun ein Programm eine ausgabe tätigt, schaut es erst einmal in `/usr/lib/locale/$LANG` nach um festzustellen ob es eventuelle auch Texte in der eingestellten Sprache gibt. Ist `LANG` nicht definiert wird immer die Locale `C` genommen, die wird auch genommen falls sich in `LANG` ein falscher Wert befindet.

LOGNAME Speichert den Namen des eingeloggten Benutzers

MAIL Speichert den Dateiennamen zur Maildatei. Programme die mit EMails arbeiten verwenden die Variable für die Maildatei.

[1] `IFS`-Internal Field Separator

MAILCHECK Die Shell prüft ab und an ob Sie neue Mail bekommen haben. Ist dem so, zeigt Sie es durch ein simples *You have mail.* an. Den Intervall kann man über diese Variable entsprechend ändern. Standardwert ist 600, was gut und gerne 10 Minuten entspricht. Wird der Wert auf 0 (NULL) gesetzt prüft die Shell die Mail jedesmal bevor sie den Prompt anzeigt

MANPATH Das Programm `man` (siehe 41 auf Seite 323) benötigt die Information, wo es suchen soll nach Manpages. Dort können die Pathnamen durch ein Doppelpunkt voneinander getrennt aufgelistet werden. Standard : Kein Path.

PAGER Werden Manpages angezeigt, oder Mails angezeigt, oder andere Sachen, d.h. wenn ein Programm irgendetwas Seitenweise anzeigen will werwendet es meist das Programm welches sich hinter `$PAGER` verbirgt. Standard ist `more` kann aber auch ein anderes Programm sein.

PATH Wenn die Shell nach Programmen sucht, sucht sie in den hier aufgelisteten Verzeichnissen. Die einzelnen Verzeichnisse muessen durch Doppelpunkte voneinander getrennt werden. Standard : `/bin:/usr/bin`

PS1 In dieser Variablen wird das aussehen des Promptes angegeben. Standardmässig ist hier nur ein Dollar und ein Leerzeichen eingestellt.

PS2 Wird auf der Kommandozeile ein Befehl eingegeben und man hat eine unsaubere Anführungszeichen Konstruktion gewählt und man drückt Return. Dann erwartet die Shell von Ihnen das Ende der Zeichenkette. Um diese Eingabe vom Prompt zu unterscheiden sollte man `PS2` entsprechend konfigurieren. Standard ist eine Spitzeklammer

SHELL Hier hinterlässt die Shell sich selbst.

TERM Bestimmt das Terminal. Fast alle Programme holen sich aus dieser Einstellung das verwendetet Terminal. Der Eintrag ist ein verweis auf die Konfigurationsdatei des Terminals die in `/usr/share/lib/terminfo` abgelegt ist, oder man verwendet `infocmp` (siehe 41 auf Seite 293). Standardeinstellung ist abhängig vom Betriebssystem

TZ Viele Programme benötigen die Angabe der Zeitzone. Diese Angabe beziehen sich fast alle Programme aus dieser Variablen. Standard : Keine ! Muß bei der Installation angegeben werden. Wir hier in Deutschland verwenden *CET*

12.2.1 Anpassung der eigenen Umgebung

Die Shell liest eine eventuell vorhandene `.profile` im Heimatverzeichnis ein. In dieser Datei kann ein Benutzer seine eigenen Einstellungen verwalten. Man sollte nicht vergessen diese Variablen auch zu exportieren, sofern diese fuer andere Programme interessant sind. Das folgende Beispielscript zeigt wie man sich Zugang zu den optionalen Programmen schaft :

Quelltext 12.2.1 Beispiel einer .profile

```
# Meine Private .profile

# Optionale Programme (wzb vim)
PATH=$PATH:/opt/sfw/bin
export PATH

# Manpages der optionalen Programme
MANPATH=$MANPATH:/opt/sfw/man
export MANPATH

# Den vim zum Standardeditor machen
EDITOR=/opt/sfw/bin/vim
export EDITOR
```

Es ist noch mehr möglich. Damit die Änderungen auch wirken, muß man sich neu einloggen.

Kapitel 13

FUNKTIONEN

Die Shell erlaubt die Definierung von Funktionen. Diese Funktionen können als kürzel auf der Kommandozeile benutzt werden. Ziel ist es einfach Tiparbeit zu sparen, oder ein bereits vorhandenes Programm immer mit anderen Kommandozeilenoptionen aufzurufen.

Wenn ein Befehl auf der Shell eingegeben wird, sucht die Shell zuerst alle Funktionen mit dem Namen ab. Findet die Shell keine Funktion sucht sie das Programm im Suchpath und führt es entsprechend aus.

Eine Funktion kann beliebig viele Kommandos beinhalten, die der Reihe nach abgearbeitet werden. Was für Programme das sind, ist Ihnen überlassen.

13.1 Struktur einer Funktion

Um eine Funktion zu definieren muß man eine bestimmte Syntax beibehalten. Ansonsten kommt es zu Fehlern.

```
<function-name> ()  <commando>; [<commando>;]
```

Um z.B. das Kommando `cd` und danach das Kommando `ls` auszuführen, muß man erst einmal ein Funktionsnamen sich überlegen. Ich wähle jetzt einfach mal `cdls`. Man kann nun eine solche Funktion schreiben :

```
$ cdls () { cd; ls; }
```

Bildschirmausschnitt 13.1.1: Die erste Funktion

Die Funktion wechselt ins Heimatverzeichnis und zeigt danach den Inhalt des Verzeichnisses an. Das ist zwar ganz toll, jedoch ist der Wechsel ins Heimatverzeichnis nicht immer das was man benötigt. Wenn man nun in ein anderes Verzeichnis wechslen möchte muß man das Argument welches `cd` bekommen soll mit angegebenen werden.

13.2 Funktionsaufruf mit Argumenten

Wenn man eine Funktion mit Argumenten aufruft muß man im innern der Funktion auf diese Argumente zurück greifen können. Die Shell bietet uns die Möglichkeit auf 9 (NEUN) Argumente zuzugreifen. Der Platzhalter dafür ist `$1-$9`. Das Beispeil zeigt die Verwendung mit einem Argument :

```
$ cdls () { cd $1; ls; }
$ cdls /usr/apache
bin      htdocs   include  libexec  man      perl5
$ pwd
/usr/apache
$
```

Bildschirmausschnitt 13.2.1: Funktion mit Argument

Hier wurde `/usr/apache` als Argument der Funktion übergeben. Da es das erste ist, ist es in der Funktion unter dem Namen `$1` ansprechbar. Das Programm `cd` bekommt also `/usr/apache` als Argument. `cd` handelt entsprechend und wechselt dorthinein.

Man kann auch noch dem `ls` einige Anzeigeoptionen mitgeben. Man definiert einfach für seine Funktion wie folgt: Man muß die Funktion mit Verzeichnis und mit Optionen für `ls` aufrufen.

```
$ cdls () { cd $1; ls -F $2; }
$ cdls /usr/apache -l
Gesamt 12
drwxr-xr-x   2 root     bin          512 Mai 15 19:48 bin/
drwxr-xr-x   3 root     bin          512 Mai 15 19:48 htdocs/
drwxr-xr-x   3 root     bin         1024 Mai 15 19:48 include/
drwxr-xr-x   2 root     bin         1024 Mai 15 19:48 libexec/
drwxr-xr-x   5 root     bin          512 Mai 15 19:48 man/
drwxr-xr-x   3 root     bin          512 Mai 15 19:48 perl5/
$
```

Bildschirmausschnitt 13.2.2: Funktion mit Argumenten

Ein netter Nebeneffekt tritt ein wenn man keine Argumente an die Funktion übergibt. Dann bekommt weder `cd` ein Argument, springt also in das Homeverzeichnis, und `ls` wird einfach ohne Optionen Aufgerufen.

13.3 Funktionen die Programme verdecken

Funktionen die Programme verdecken erkennt man daran, daß der Funktionsname gleich eines Programmnamens ist. Wenn man eine solche Funktion schreibt kann man damit das Standardverhalten eines Programms elegant verändern.

Damit es jedoch innerhalb der Funktion nicht zum Funktionsaufruf kommt, die Funktion hat ja den gleichen Namen wie das Programm, sollte man das Programm immer mit absolutem Path notieren. Ansonsten haben Sie eine Endlosschleife.

```
$ ls () { /bin/ls -F; }
$ ls
bin/      htdocs/   include/  libexec/  man/      perl5/
$
```

Bildschirmausschnitt 13.3.1: Umdefinierung von `ls`

Das das Programm `ls` jedoch eine variable Anzahl von Argumenten verarbeiten kann, muß auch die Funktion in der Lage sein alle möglichen Argumente zu verarbeiten. Da man mit `$1-$9` nur neun Argumente ansprechen kann, muß man `$@` verwenden. Das Dollar-At wird mit allen übergebenen Argumenten ersetzt, egal wie viele das sind. Das sollte man immer dann tun, wenn man ein Programm überschreiben will.

```
$ ls () { /bin/ls -F $@; }
$ ls -l -a -i /usr/apache /tmp
/tmp:
Gesamt 802
    726100 drwxrwxrwt   7 root     sys          784 Jun  7 21:00 ./
    726100 drwxr-xr-x  26 root     root        1024 Jun  7 06:08 ../
   1196362 drwxrwxr-x   2 root     root         176 Jun  7 04:56 .X11-pipe/
   1695296 drwxrwxr-x   2 root     root         176 Jun  7 04:56 .X11-unix/
   1694936 -rw-r--r--   1 whurst   whurst         0 Jun  7 19:47 sdtvolcheck5
   1842833 -rw-r--r--   1 whurst   whurst         4 Jun  7 04:56 speckeysd.lo

/usr/apache:
Gesamt 16
     93723 drwxr-xr-x   8 root     bin          512 Mai 15 19:48 ./
         2 drwxr-xr-x  35 root     sys         1024 Jun  7 05:12 ../
    187174 drwxr-xr-x   2 root     bin          512 Mai 15 19:48 bin/
    286313 drwxr-xr-x   3 root     bin          512 Mai 15 19:48 htdocs/
    192751 drwxr-xr-x   3 root     bin         1024 Mai 15 19:48 include/
    203754 drwxr-xr-x   2 root     bin         1024 Mai 15 19:48 libexec/
    341403 drwxr-xr-x   5 root     bin          512 Mai 15 19:48 man/
    220239 drwxr-xr-x   3 root     bin          512 Mai 15 19:48 perl5/
$
```

Bildschirmausschnitt 13.3.2: Umdefinierung von `ls` mit variablen Argumenten

13.4 Verwaltung von Funktionen

Um alle Funktionen anzeigen zulassen kann `set`[1] verwendet werden. Da Funktionen wie auch Environmentvariablen im Environmentspeicher sich befinden.

Funktionen können nicht exportiert werden. Sie können jedoch auf ReadOnly gesetzt werden, mit `readonly`[2]. Das eignet sich ganz gut in der /etc/profile um den Benutzer zu zwingen entsprechende Optionen zu benutzen. Leider lässt sich das Umgehen, indem man einfach eine neue Shell startet, die Funktionen werden ja nicht exportiert.

Funktionen können mit `unset`[3] auch wieder gelöscht werden, wie Environmentvariablen auch.

[1] `set` - Siehe Kommandoreferenz Seite **??**
[2] `readonly` - Siehe Kommandoreferenz Seite **??**
[3] `unset` - Siehe Kommandoreferenz Seite **??**

Kapitel 14

DAS ERSTE SHELLSCRIPT

Ein Shellscript ist nur eine gewöhnliche ASCII Textdatei in der einfach nur die Programme aufgelistet sind, die man auch auf der Kommandozeile eingeben hätte können. Mit einem ShellScript kann eine solche Arbeit jedoch vermindert werden.

14.1 Besonderheiten an einem Shellscript

Es gibt zwei Punkte die man beachten muß.

Die erste Zeile in einem Shellscript benötigt die Angabe des Interpreters. Da es unter UNIX keine Dateiextension gibt, muß der Kernel wissen mit was er den ASCII Text interpretieren soll. Um den Kernel dieses Mitzuteilen muß in der ersten Zeile ein `#!` stehen. Gefolgt vom absoluten Path des Interpreters. Da wir nur Bourne Shellscripts schreiben wäre dieses `/bin/sh`. Aber hier sich auch andere Interpreter denkbar, wie Perl, C-Shell, tc-Shell oder AWK oder andere. Wird diese Zeile weggelassen wird immer die Shell in `$SHELL` mit der Ausführung beauftragt. Das ist dann für den ärgerlich der die `csh` verwendet. Die sind nicht kompatibel.

Die Rechte der Datei müssen das Leserecht und Ausführrecht aufweisen. Beliebt sind `755` oder andere Kombinationen.

14.2 Verwenden von Shellscripts

Wenn man ein Shellscript aufrufen will, benutzt man einfach den Namen des Scriptes. Ist das Script im Suchpath (`$PATH`) dann kann das Script ohne Path aufgerufen werden. Wichtig dabei ist jedoch das man kein Namen wählt der bereits im System genutzt wird, wie `ls` oder so.

14.3 Hello World

Das folgende Beispiel zeigt ein Simples Shellscript mit dem Namen `helloworld` :

Quelltext 14.3.1 Beispielscript `helloworld`

```
#!/bin/sh
#
# Das ist ein einfaches Shellscript. Es gibt nur "Hello World" aus
#

echo Hello World
```

In einem Script koennen auch Environmentvariablen verwendet werden :

Quelltext 14.3.2 Beispielscript `helloworld2` mit Variablen

```
#!/bin/sh

Var1=Hello
Var2=World

echo $Var1 $Var2
```

Ein Shellscript kann aber auch noch andere Programme aufrufen. Das folgende Script wechselt in das /usr/bin Verzeichnis und startet dort ein `ls -la` mit `more`. Als Namen könnte man hier `lsubin` z.B. waehlen :

Quelltext 14.3.3 Beispielscript `lsubin`

```
#!/bin/sh

cd /usr/bin
ls -la | more
```

Nachdem das Script beendet wird befindet man sich im alten Verzeichnis. Das `cd` hat keine Auswirkungen auf die aktuelle Shell.

14.4 Praktikum - Einfache Scripts

- Erstellen Sie in Ihrem Heimatverzeichnis ein Verzeichnis für die Scripts die wir schreiben werden. Nennen Sie das Verzeichnis am Besten bin oder auch anderst.

- Nehmen Sie das Verzeichnis mit in Ihren Suchpath mit auf. Editieren Sie dazu die .profile wie im Beispiel **??** auf Seite ??. Loggen Sie sich erneut ein und prüfen Sie ob Ihr Path gesetzt ist

- Schreiben Sie ein Shellscript in Ihrem Scriptverzeichnis, das ein `ls -la` simuliert. Nennen Sie das Script `la`. Testen Sie es aus indem Sie irgendwo im Verzeichnisbaum ein `la` aufrufen.

- Schreiben Sie ein Script mit dem Namen `lsubin` (siehe **??** auf Seite ??). Verwenden Sie jedoch für den `ls -la` Befehl ihr eigenes Script.

- Schreiben Sie ein Script mit dem Namen `infome`, das etwa folgende Ausgaben tätigt, die Werte in Spitzenklammern müssen Sie natürlich mit dem Sinn ersetzen :

```
$ infome
InfoMe Version 1.0
------------------
Hallo <Benutzername> ich will dich mal Informieren :

   Dein Benutzername : <Benutzername>
Deine Sprachumgebung : <Sprachumgebung>
 Dein Lieblingspager : <Pager>
    Deine Heimat ist : <Heimatverzeichnis>
Du befindest dich in : <aktuelles Verzeichnis>
       Dein Suchpath : <Suchpath>

Viel Spass noch
$
```

Bildschirmausschnitt 14.4.1: Beispiel Ausgabe von `infome`

Kapitel 15

KONTROLLSTRUKTUREN

Kapitel 16

Pipe Konstruktionen mit Controllstrukturen

TEIL III

LOKALE ADMINISTRATION

- Installation
- Gerätedateien
- UFS Dateiensystem Aufbau
- Systemstart
- Terminalkonfiguration
- Benutzermanagement
- Rechtemanagement
- Quota
- Druckerkonfiguration
- Backup und Archivierung
- Zeitgesteuert Aktionen ausführen
- Softwaremanagement

INSTALLATION

17.1 Hardware Configuration Assistant (intel)

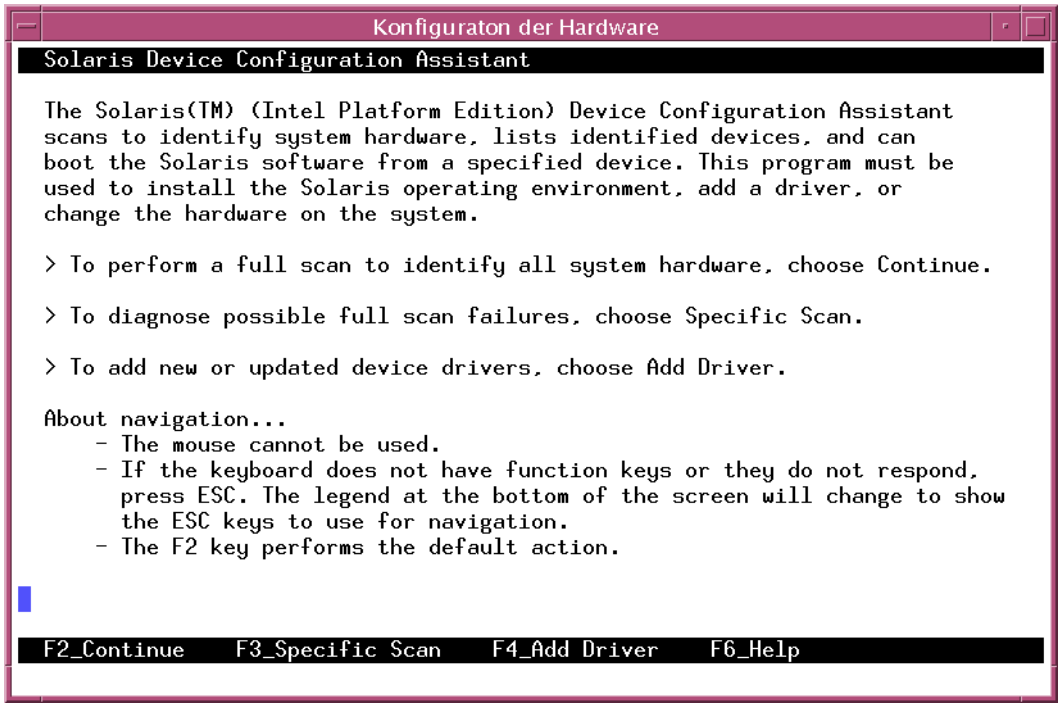


Abbildung 17.1: b

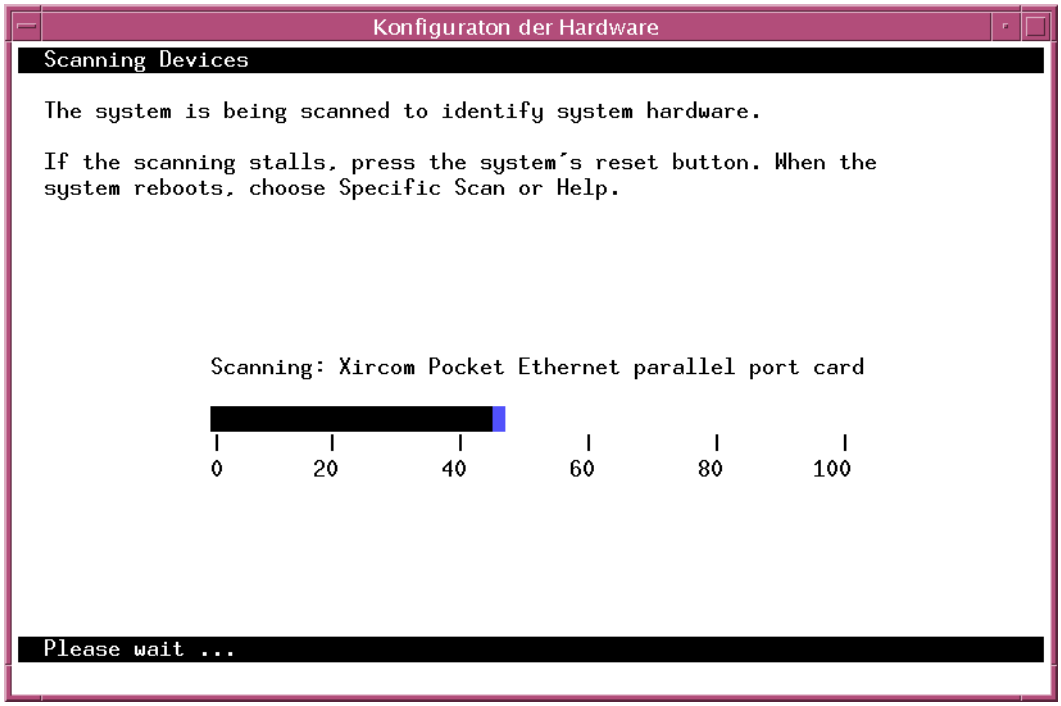


Abbildung 17.2: b

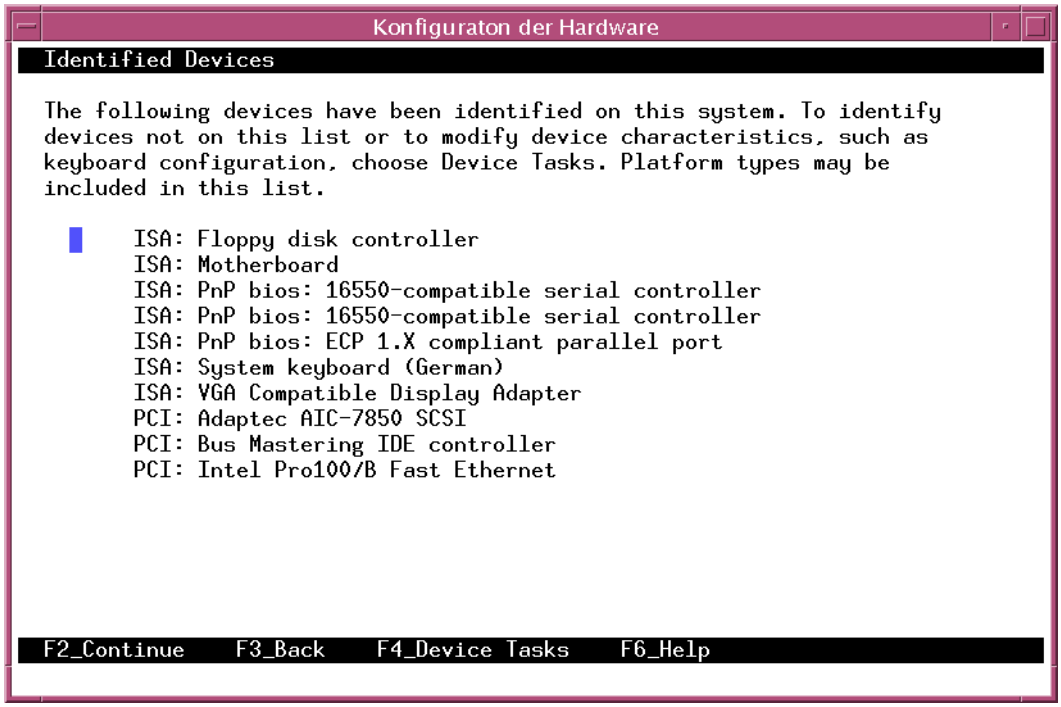


Abbildung 17.3: b

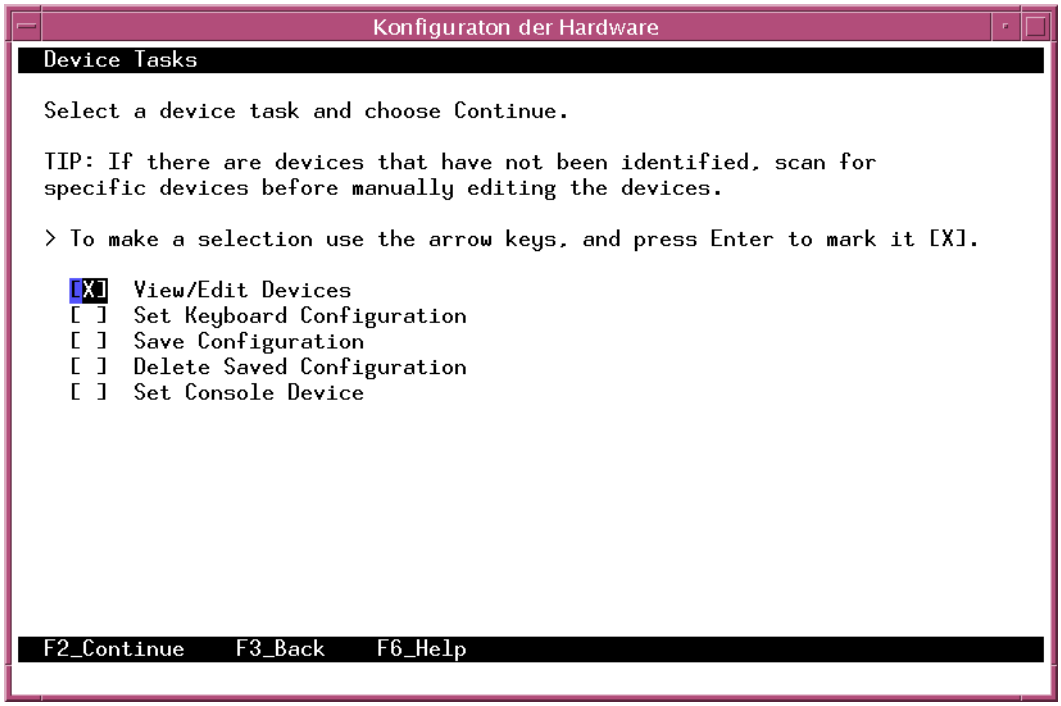


Abbildung 17.4: b

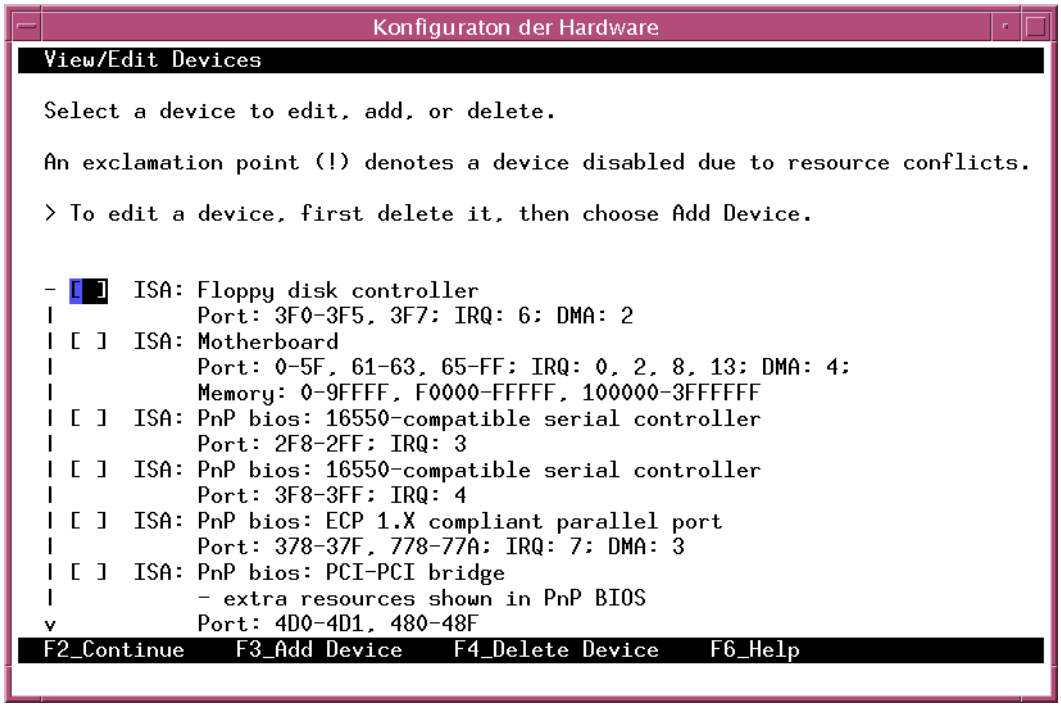


Abbildung 17.5: b

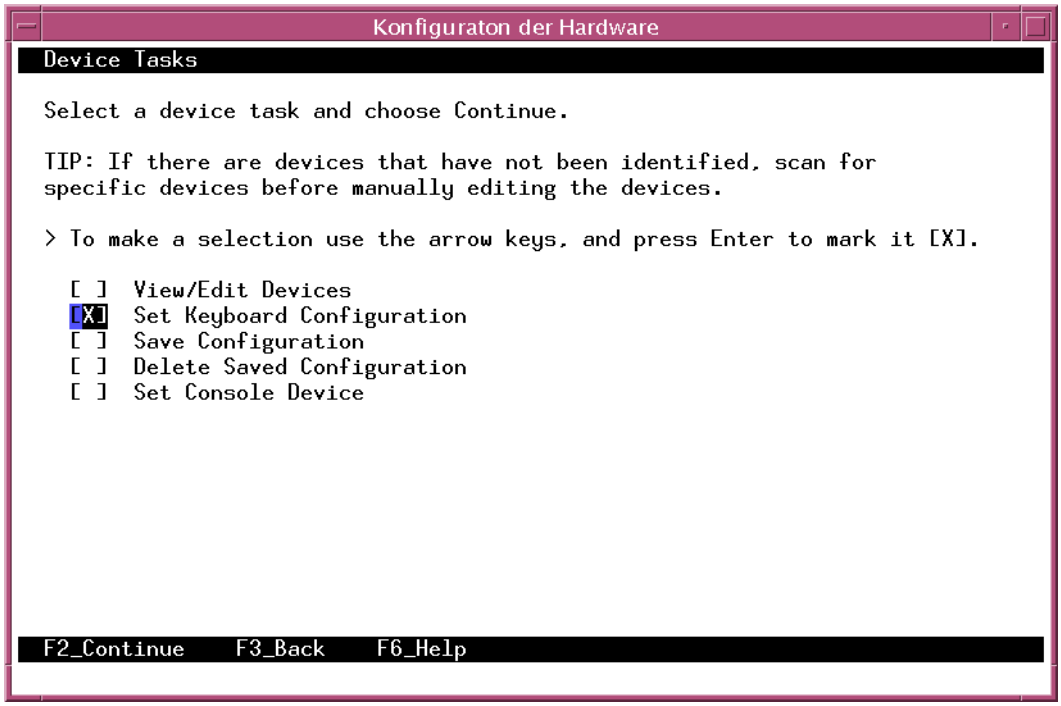


Abbildung 17.6: b

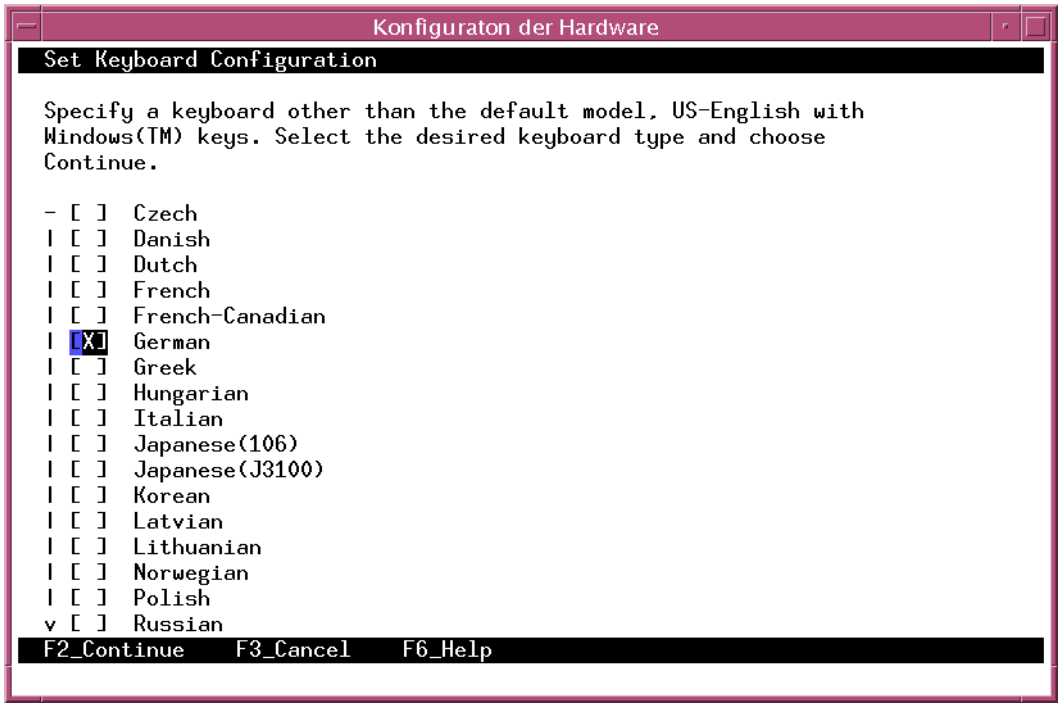


Abbildung 17.7: b

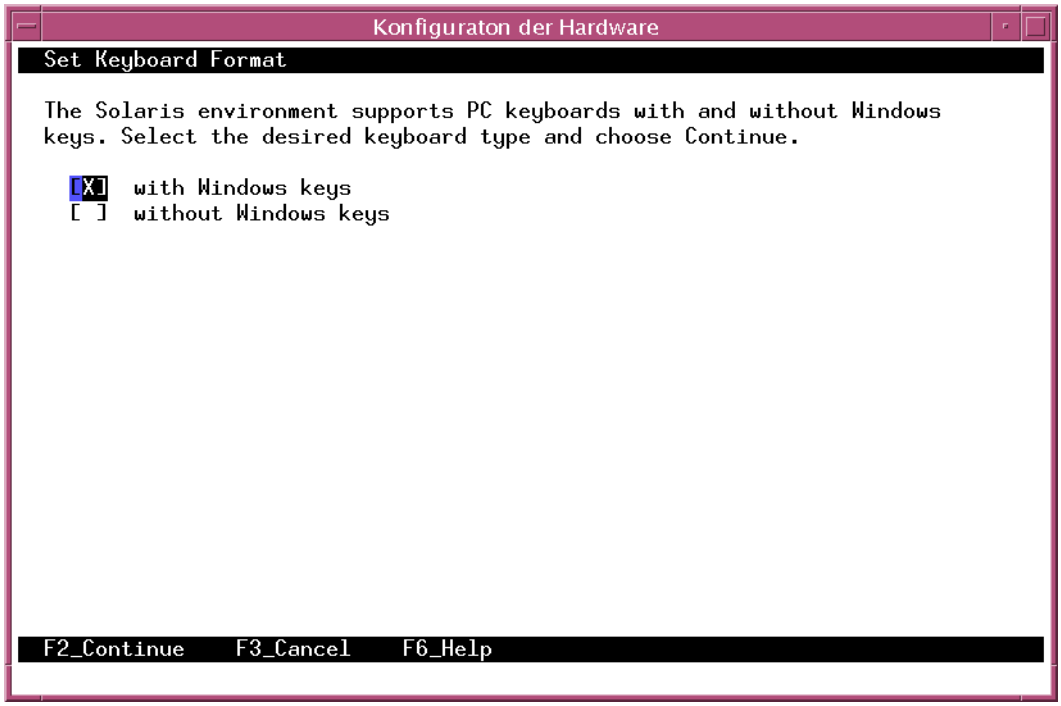


Abbildung 17.8: b

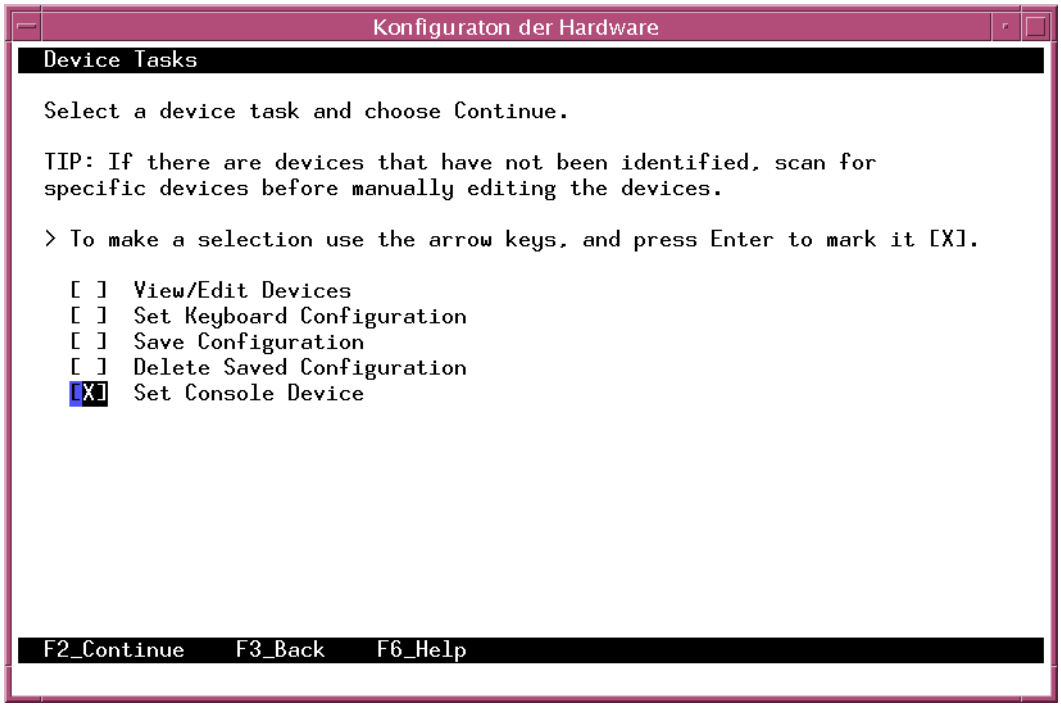


Abbildung 17.9: b

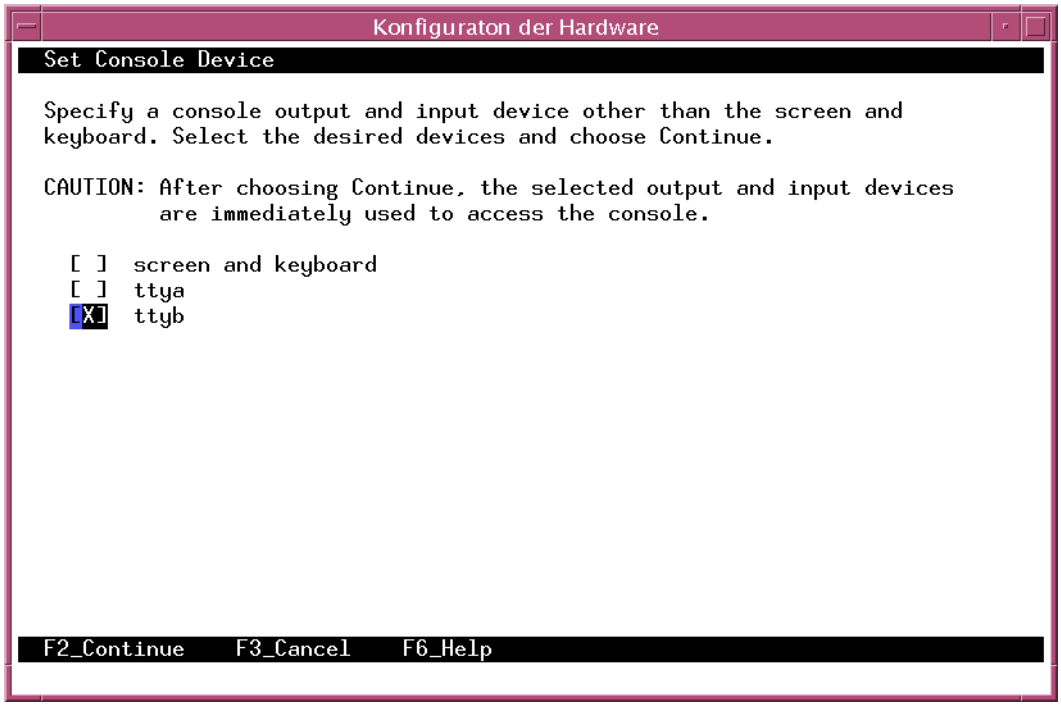


Abbildung 17.10: b

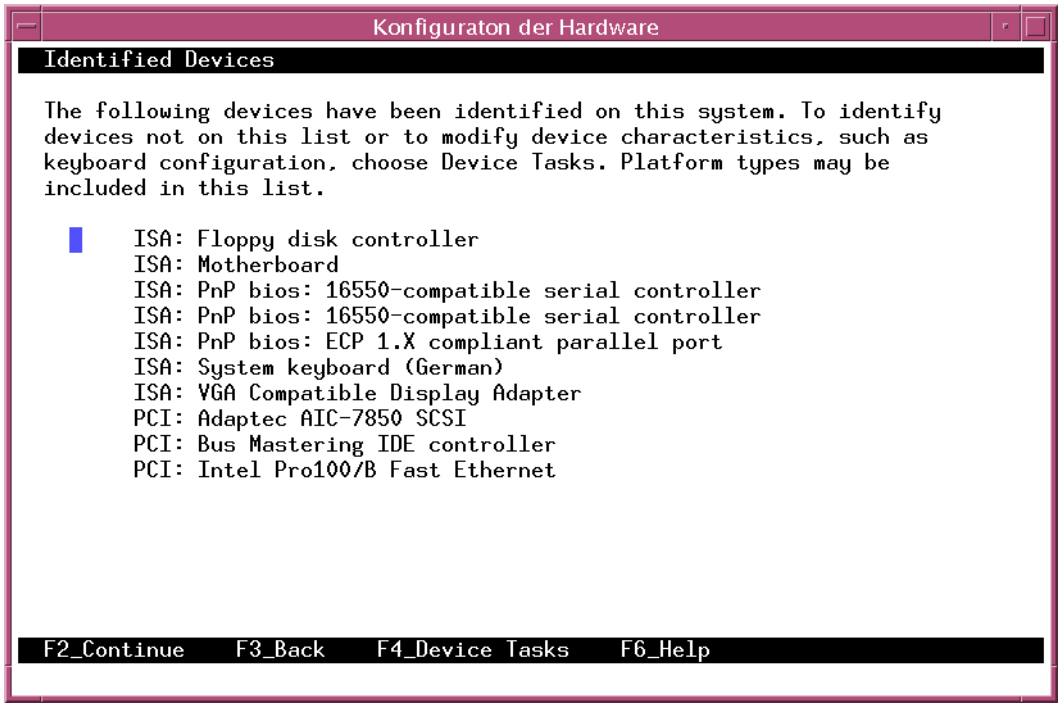


Abbildung 17.11: b

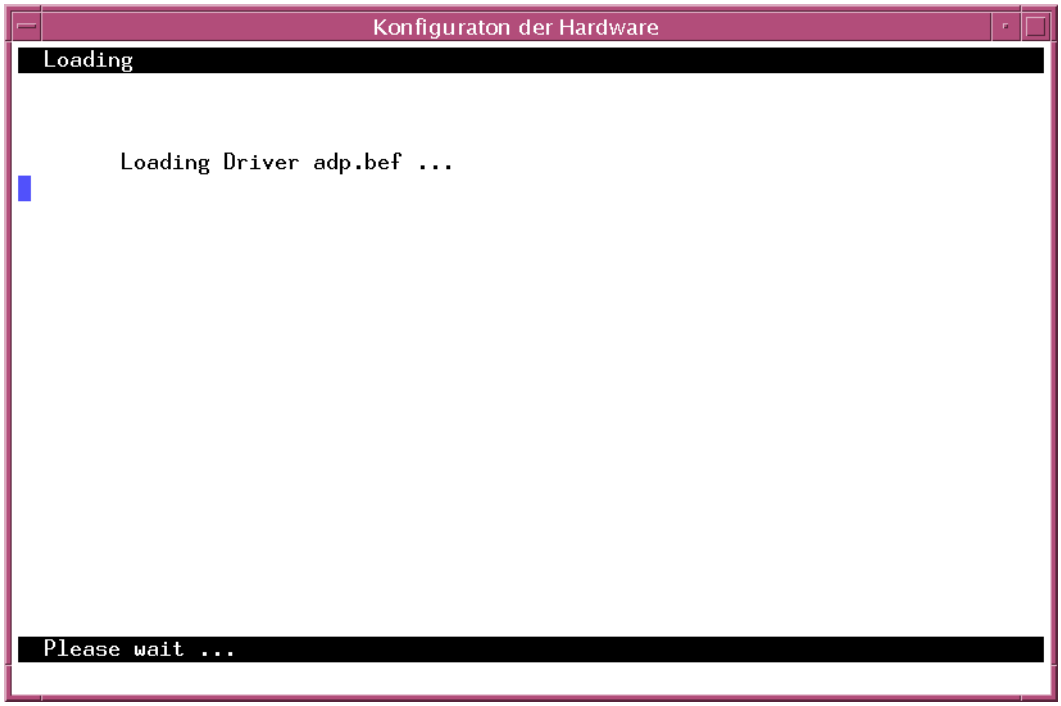


Abbildung 17.12: b

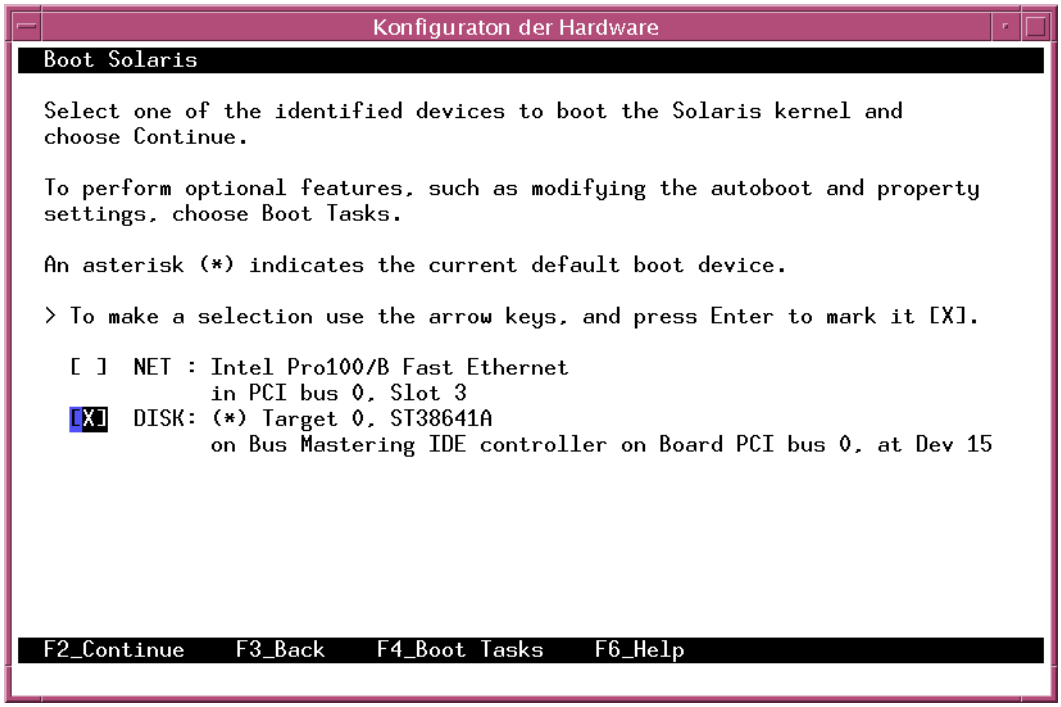


Abbildung 17.13: b

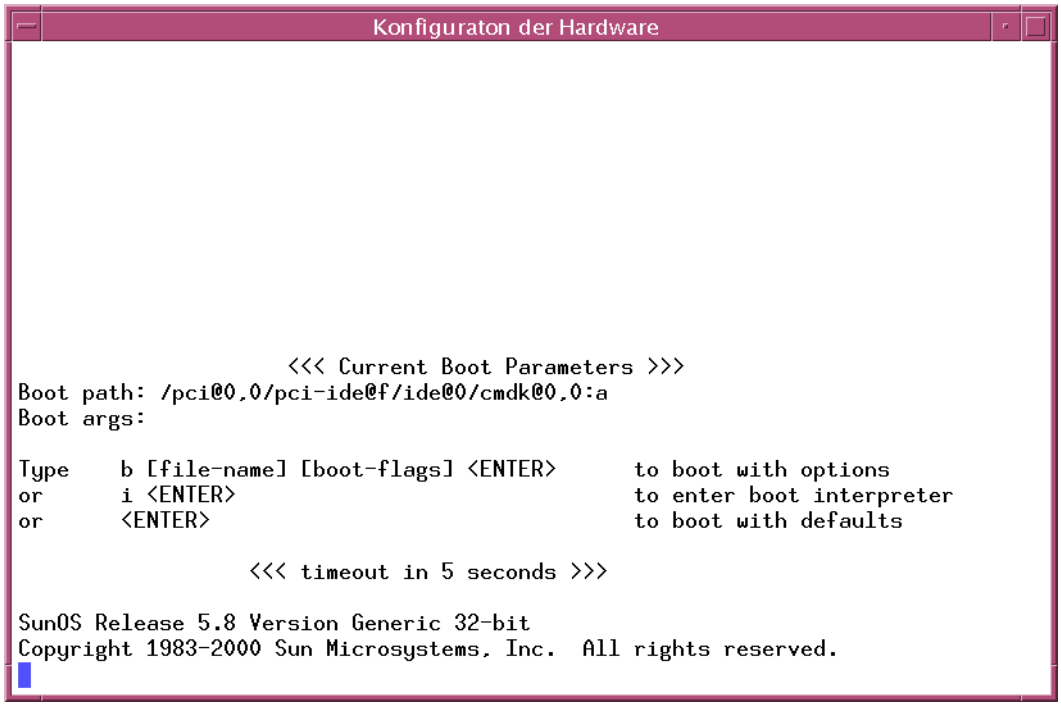


Abbildung 17.14: b

Kapitel 18

GERÄTEDATEIEN

Auf SPARC Rechnern springt die CPU, nach dem Poweron, in das OBP[1]. Das OBP ist vergleichbar mit dem BIOS eines PCs. Das OBP ist die Firmware des Rechners. Mit dem OBP können diverse Einstellungen des Rechners rausgefunden oder gesetzt werden. Dazu gehört auch die Angabe des Bootdevices. Die Einstellungen des OBP können auch vom laufendem Solaris System mit `eeprom`[2] entsprechend angezeigt oder geändert werden.

18.1 Terminologie

18.1.1 Hardware Path

oder auch Physikalischer Gerätename. Er identifiziert ein Hardware gerät. Wobei die Geräte sich als Dateibaum zeigen und man so einen Path zum Gerät erhält. Z.B. ein SCSI Controller liegt am PCI Bus. Also könnte man /pci/scsi sagen. Ja nun so einfach ist es nicht. Es wird noch die Slotnummer des Slots angegeben, der SCSI Controller bekommt noch die Vendor und Minor Nummer des Herstellers verpasst. Bis man dann am Schluß auf den wirklich unmerkbaren Konstrukten kommt :

/pci@1f,0/pci@1,1/ide@3 z.B. Definiert den PCI Bus Nummer 0 mit der Portaddresse 1f. An diesem PCI Bus ist im Slot 1 das 2. Gerät ein IDE Controller. Der IDE Controller ist der 4. Controller überhaupt.

/iommu/sbus/espdma@5,8400000/esp@5,8800000/sd@0,0:a ist noch besser. Der physikalische Path kommt von einer SPARC. Dort am iommu (IO Baustein) existiert ein SBUS. An diesem SBUS befindet sich im Slot 6 an der Addresse 8400000 ein DMA Kontroller. An diesem Controller befindet sich im Slot 6 an der Addresse 8800000 ein SCSI Controller. An diesem SCSI Controller befindet sich eine Platte mit der SCSI-ID 0 und der LUN 0. Das :a gibt die Slice 0 an.

18.1.2 Logische Geräte

Da ein physikalische Gerätename sehr abhängig und auch sehr aussage unkräftig ist, werden die Hardwarepath in logische Gerätenamen umgewandelt. Ein UNIX System arbeitet immer mit den logischen Gerätenamen. Unter Solaris sind die logischen Gerätenamen ein symbolischer Link zum Hardwarepath und man kann Ihn jederzeit ändern. (Sollte man vielleicht nicht gleich ausprobieren).

/dev/ecpp0 z.B. ist ein logisches Gerät für die parallele Schnittstelle auf einer SPARC. In Wirklichkeit ist es jedoch ein Link auf /devices/pci@1f,0/pci@1,1/ebus@1/ecpp@14,3043bc:ecpp0

18.1.3 Vendor-ID

Eine Vendor-ID ist eine Nummer für das Gerät selbst. Jedes Gerät bekommt eine Herstellernummer verpasst. Diese Nummer wird bei einem Intelsystem z.B. wenn das BIOS bootet angezeigt (meistens). Ein Adaptec 7950 SCSI Controller hat z.B. die Vendor-ID 7950. Der taucht im Hardwarepath in der folgenden Form auf /pci@0,0/pci7950@4,0/...

[1] `OBP`-OpenBoot PROM

[2] `eeprom` - Siehe Kommandoreferenz Seite **??**

18.1.4 Major Nummer

Der UNIX Kernel benutzt für die verschiedenen Geräte eine Nummer. Mit dieser Nummer kann der UNIX Kernel ein Bezug zum Gerät aufbauen. Jeder UNIX Kernel definiert andere Major Nummern. Unter Solaris ist z.B. die Major Nummer für eine para. Schnittstelle die 153, unter Linux kann es z.B. die 6 sein.

18.1.5 Minor Nummer

Mit der Minor Nummer wird einem Gerät gesagt welches Subgerät man haben will. z.B. SCSI-Festplatten werden alle am gleichen SCSI-Controller betrieben. Um die Festplatten anzusprechen benötigt man die Major Nummer des SCSI-Controllers und die Minor Nummer der Festplatte. Minornummern fangen meist bei 0 (NULL) an und werden einfach hochgezaehlt.

18.1.6 Instanz

Eine Instanz ist nur eine Abkürzung für ein physikalisches Gerät mit Minornummer. z.B. sd0 ist der erste SCSI-Controller.

18.1.7 Gerätedatei

Eine Gerätedatei ist unter UNIX ein sonderfall. Da UNIX keine sogennante Regestry hat muß UNIX irgendwie zwischen einer normalen Datei und einer Gerätedatei unterscheiden können, damit UNIX weiß das man nun ein Gerät am Wickel hat. Um das UNIX Dateiensystem für Gerätedateien zu benutzen hat man einfach die Größe einer Datei zur Aufnahme von Major und Minor Nummern benutzt. Somit werden die Gerätedateien für das UNIX System auch innerhalb des UNIX Systems erkannt. Meistens liegen diese in /dev und /devices. Die Gerätedateien sind die logischen Gerätedateien.

18.2 Gerätedateien Rekonfiguration

Wenn das System hochfährt werden die physikalischen Gerätedateien in Instanzen gewandelt. Dafür wird die /etc/path_to_inst benutzt. Bei der Rekonfiguration werden alle Gerätedateien neu angelegt. Das Programm `drvconfig`[3] erstellt den kompletten /devices Baum mit den physikalischen Gerätedateien. `drvconfig` benutzt die /etc/name_to_major um der Instanz die richtige Majornummer zu geben.

`devlinks`[4] erstellt nun die logischen Gerätenamen in das /dev Verzeichnis. Dabei benutzt es die /etc/devlink.tab fuer ganz spezielle Gerätedateien.

18.2.1 Das OBP

Nach dem PowerON zeigt das OBP den Banner an und inizialisiert den Speicher und die anderen Geräte. Der Banner zeigt Informationen über den verwendeten Rechner an :

[3] `drvconfig` - Siehe Kommandoreferenz Seite **??**
[4] `devlinks` - Siehe Kommandoreferenz Seite **??**

```
ok banner
```

```
Sun Ultra 5/10 UPA/PCI (UltraSPARC-IIi 333MHz), Keyboard Present
OpenBoot 3.19, 192 MB (50 ns) memory installed, Serial #11605854.
Ethernet address 8:0:20:b1:17:5e, Host ID: 80b1175e.

ok go
```

Abbildung 18.1: OBP: Der Banner einer Ultra-10

Anhand der Grafik die angezeigt wird kann man schon erkennen welche Grafikkarte eingebaut und aktiv ist. Ein normales Sun Logo kennzeichnet eine PGX, ein 3D Sun Logo als Raytracing kennzeichnet eine CG6, eine Weltkugel hingegen kennzeichnet eine Creator 3D.

Das OBP kann man jederzeit, auch im laufendem Solarissystem, aufrufen. Das OBP ist Kommandozeilenorientiert und als Prompt wird `ok` angezeigt.

Um in das OBP zu gelangen muß man **STOP-A** drücken. Danach können dem OBP Befehle mitgeteilt werden. Anderst als bei PCs wo es Menues gibt. Ein Befehl wäre `banner` mit dem man sich den Banner nochmal anschauen kann.

18.2.2 Gerätemanagment im OBP

Alle Geräte und Bussysteme einer SPARC werden mittels Dateibaum abgebildet. Es existieren im OBP die Befehle `cd` und `ls` mit dem man sich durch den Gerätewald hangeln kann.

```
ok cd /
ok ls
f006dcb0 SUNW,ffb@1e,0
f006d428 SUNW,UltraSPARCIIi@0,0
f005fa24 pci@1f,0
f004d4f8 virtual-memory
f004cf18 memory@0,0
f002ccf0 aliases
f002cc80 options
f002cb48 openprom
f002cadc chosen
f002ca6c packages
ok cd pci
ok ls
f0060800 pci@1
f0060218 pci@1,1
ok cd pci@1,1
ok ls
f0085fd0 ide@3
f007dc00 SUNW,m64B@2
f0076438 network@1,1
f0061568 ebus@1
ok cd ide
ok ls
f0088f1c cdrom
f0088870 disk
ok
```

Bildschirmausschnitt 18.2.1: OBP: Gerätemanagement

Dieses Beispiel geht runter auf /pci@1f,0/pci@1,1/ide@3. Die Gerätenamen haben folgendes Format :

```
< driver-name > @ < unit-address > : < device-arguments >
```

driver-name Der Treibername identifiziert den Typ der Hardware. Meist wird der Herstellername in Grossbuchstaben davor gestellt. z.B. SUNW,ffb

unit-address Die Hardwareaddresse des Gerätes. Das format der Addressierung ändert sich zwar nicht, jedoch hat es bei jedem Gerät eine andere Bedeutung. z.B pci@1f,0 sagt aus das sich der PCI Bus an der Addresse 1f angemeldet hat und die Null sagt ïch bin im lokalem System die Nummer eins". Bei SUNW,m64B@2 jedoch hat die 2 die Aussage der Slotnummer auf dem PCI Bus.

device-arguments Die Argumente an ein Device sind optional. Auch hier gilt, die Angabe der Optionen sind Contextabhängig. Optionen findet man meist bei Festplatten, CD-ROMs etc pp. z.B. disk2:a wird als erste Slice auf der Platte identifiziert.

18.2.3 Einstellungen zum Bootdevice

Einer SPARC muß man genauso wie einem PC sagen von welchem Device das Betriebssystem gebootet werden soll. Um zu Booten muß das OBP das Device bekommen, damit es von diesem Device Booten kann. Dazu existiert der Befehl `boot`.

```
ok boot /iommu/sbus/espdma@5,8400000/esp@5,8800000/sd@0,0:a
....
```

Bildschirmausschnitt 18.2.2: OBP: Booten von Platte

Mit dem Befehl wird von Slice 1 der SCSI-Platte mit der ID 0 angeschlossen am SCSI Controller der sich im 5. Slot des SBUSes befindet gebootet. Diese Art ist sehr umständlich. Man kann jedoch auch Aliasnamen im OBP Speichern und diese dann verwenden :

```
ok nvalias disk /pci@1f,0/pci@1,1/ide@3/disk:a
ok boot disk
....
```

Bildschirmausschnitt 18.2.3: OBP: Booten von Platte mit Aliasnamen

Das ist kürzer. Mit `nvunalias` kann der Name wieder gelöscht werden. Im Normalfall legt man `disk`, `cdrom` an. `net` wird bereits vom System erstellt.

Um sich alle bootbaren Devices anzeigen zulassen existiert der Befehl `show-disks` der alle Devices anzeigt. Man wählt ein Gerät aus und der Path landet in einer Zwischenablage den man mit **CTRL-Y** ansprechen kann :

```
ok show-disks
a. /pci@1f,0/pci@1,1/ide@3/cdrom
b. /pci@1f,0/pci@1,1/ide@3/disk
q. NO SELECTION

Enter Selection, q to quit: a
/pci@1f,0/pci@1,1/ide@3/cdrom has been selected.
Type ^Y ( Control-Y ) to insert it in the command line.
e.g. ok nvalias mydsk ^Y
     for creating devalias mydsk for
/pci@1f,0/pci@1,1/ide@3/cdrom
ok nvalias cdrom ^Y:f
ok boot cdrom
```

Bildschirmausschnitt 18.2.4: OBP: Aliasnamen vereinbaren

Mit `devalias` kann man sich alle Aliasnamen anschauen.

Einschränkungen wie ”Booten nur vom ersten Gerät aus”wie es in der Intelwelt so ist, gibt es hier nicht. Es kann von allem gebootet werden wo sich Daten draufbefinden.

18.2.4 Automatischer Bootvorgang

Dem OBP muß gesagt werden das es automatisch Booten soll. Dazu muß die Variable *auto-boot?* auf *true* gesetzt werden. Und in der Variablen *boot-device* müssen alle bootfähigen Devices eingetragen werden. Im Normalfall ist das schon gemacht. Mit dem `printenv` kann man sich alle Variablen anschauen :

```
ok printenv
Variable Name       Value       Default Value
.....
auto-boot?          false       true
boot-device         disk cdrom  disk cdrom net
.....
ok setenv auto-boot? true
ok
```

Bildschirmausschnitt 18.2.5: OBP: Anzeigen und setzten von Variablen

Kapitel 19

UFS DATEIENSYSTEM AUFBAU

19.1 Gerätenamen

Unter Solaris werden die Hardwaredevices im Verzeichnis /devices erstellt. Die Hardwaredevices dort können bei jedem Bootvorgang neu generiert werden (Booten mit der Option -r). Wenn das System danach seine Devices konfiguriert erstellt es im /dev Verzeichnis die logischen Gerätenamen. Und zwar nur die, die auch wirklich nur existieren.

Solaris geht nun hin und nummeriert die Controller durch. Der erste Controller bekommt die 0 (NULL), der zweite die 1 (EINS) usw.

Danach wird an jedem Controller das eigentliche Laufwerk gesucht. (Target) und nummeriert diese durch.

Sofern es SCSI sachen sind und eventuell CD-Wechsler hat jedes dieser Geräte logische Unternummern (LUN[1]). Auch diese werden durchnummeriert. Sollte ein Gerät keine LUN's haben wird immer 0 (NULL) voreingestellt

Und ganz am schluß werden die Slices durchnummeriert.

Um die Zahlen auch auseinanderzuhalten werden den Zahlen Buchstaben vorgestellt. in c für den Controller, ein t für das Target, ein d für das Device (LUN) und ein s für das Slice. Auf Intel-Solaris existiert noch das p für Partition. (SPARC Rechner kennen keine Partitionen)

Man kann es also wie folgt darstellen :

$$c < controller > t < target - id > d < device > s < slice >$$

Beispiele

Das Device c1t1d0s2 wäre das Laufwerk mit der SCSI-ID 1 am zweiten Controller. Angesprochen wird das Slice 2.

Bei IDE würde c0t1d0s2 die Slave-Platte am Master-Controller mit der Slice 2 angesprochen werden.

19.1.1 Logische und physische Geräte

Ein physischer Gerätename beschreibt immer das komplette Slice mit allen Informationen. Das muß man zur Erstellung von Dateisystemen benutzen. Man nennt diese Geräte auch raw (Roh) Gerät und sind immer unter /dev/rdsk zu finden.

Ein logischer Gerätename beschränkt sich auf das beinhaltete Dateiensystem, sprich die Daten. Dieses Gerät kann man mounten. Diese Geräte findet man unter /dev/dsk.

[1] LUN-Logical Unit Number

19.2 Partitionierung

Auf SPARC Rechner gibt es keine Partitionen punkt

19.3 Slices

Solaris ist in der Lage pro Laufwerk 8 Slices zu verwalten. Diese Slices werden von 0 (NULL) bis 7 durchnummeriert. Jedes Slice kann ein Dateiensystem enthalten, welches man später dann benutzen kann.

Das Slice mit der Nummer 2 ist jedoch etwas ganz besonderes und sollte möglichst nicht anderst verwendet werden (es geht zwar, aber gut ist es nicht). Das Slice 2 beschreibt im Normalfall die gesammte Platte. und wird auch meist *overlap* bezeichnet.

Die folgende Grafik zeigt ein möglichen Aufbau eines Laufwerkes in Slices :

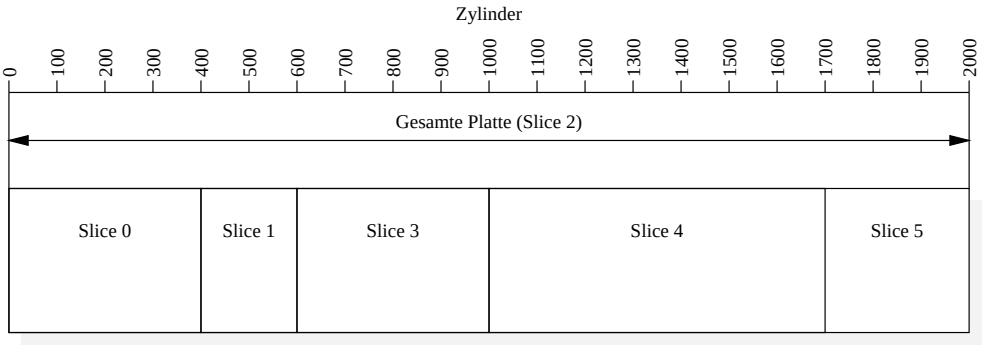


Abbildung 19.1: Einteilung eines Laufwerkes in Slices

Anhand der Grafik kommt man zu folgender Tabelle :

Tabelle 19.1: Slice Tabelle

Slice	Startzylinder	Endzylinder
Slice 0	0	399
Slice 1	400	599
Slice 2	0	1999
Slice 3	600	999
Slice 4	1000	1699
Slice 5	1700	1999
Slice 6	unbenutzt	
Slice 7	unbenutzt	

Welches Datiensystem sich innerhalb einer Slice befindet ist vorerst egal. Es kann ein UFS, ext2 (Linux), pcfs (DOS) oder sonstiges sein.

Übertrieben gesehen ist es nix anderes als eine Partitionierung von Festplatten unter Intel-Betriebssystemen.

19.4 Aufbau eines UFS Dateiensystems

Wenn in einer Slice ein UFS installiert wird, wird diese Slice vom UFS in mehrere Bereiche eingeteilt. Die folgende Grafik schaut der Slice 4 ins innere :

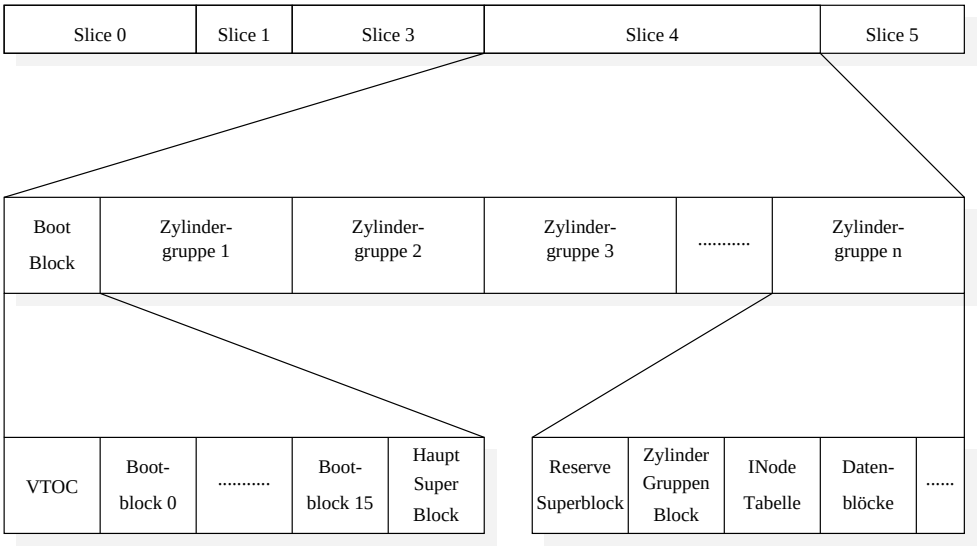


Abbildung 19.2: Aufbau des UFS Dateiensystems

Die ganzen Blöcke werden wir uns mal in Detail anschauen :

VTOC Der VTOC[2] speichert Informationen über die Platte selbst. Sprich welche Slices es gibt, wie groß diese sind (also die Slicetable). Wieviele unbenutzte Blöcke es gibt, etc.pp. Der VTOC ist auf jedem Laufwerk immer einmal auf Sektor 0 (NULL) und auf jedem Slice vorhanden. Man kann `prtvtoc`[3] benutzen um sich die Daten anzuschauen. Um jedoch auf diesen Block zu gelangen muß man das *raw* Device benutzen.

Bootblöcke In den Bootblöcken wird das Bootprogram abgelegt. Das benötigt das System wenn es bootet. Das Bootprogramm läd daraufhin den Kernel. Wenn der Bootblock jedoch einmal kaputt geht kann das Programm `installboot`[4] verwendet werden um das Bootprogramm neu zu erstellen.

Hauptsuperblock Der Hauptsuperblock beinhaltet Informationen über die Größe und Anzahl der Zylindergruppen. Die maximale Anzahl von INodes, die Anzahl der freien und somit auch die Anzahl der belegten INodes. Da diese Informationen tot-wichtich sind wird der Superblock auch mehrfach gesichert, in jerder Zylindergruppe einmal.

Reserve Superblock Hier wird eine Kopie des Hauptsuperblockes abgelegt. Im Reseversuperblock wird jedoch noch das Datum des letzten schreibenen Zugriffes gespeichert. Sollte der Hauptsuperblock mal kaputt gehen kann auf die Reseveblöcke der Zylindergruppen zugegriffen werden.

[2] `VTOC`-Volume Table Of Contents
[3] `prtvtoc` - Siehe Kommandoreferenz Seite 363
[4] `installboot` - Siehe Kommandoreferenz Seite 297

Zylindergruppenblock In dem Zylindergruppenblock werden die Informatioenen zu einer Zylindergruppe abgelegt. Aus wievielen Zylindern die Gruppe besteht (Standardmäßig 16). Wieviele freie Datenblöcke und INodes es gibt. Und wann die letzte Veränderung in dieser Zylindergruppe vorgenommen wurde. Zylindergruppen vermindern eine Fragmentierung.

INode Tabelle Jede Datei benötigt Informatioen, wzB wie groß diese ist, welche Rechte es so gibt und wo der Inhalt sich auf dem Datenträger befindet. Genau diese ganzen Informationen befinden sich in einer INode. Eine INode ist nicht sehr groß, meist 128 Byte. (Noch weiss ich nicht wie ACL's gespeichert werden, und wo).

Datenblöcke Ein Datenblock speichert den Dateninhalt von Dateien und Verzeichnissen. Ein Datenblock ist im Normalfall 8 KBytes gross und kann auf 4 KBytes reduziert werden. Wird jedoch nur von ausgewählten Hardwareplattformen unterstützt. Ein Interner Mechanismus erlaubt die Einteilung eines Datenblockes in 8 Teile je 1 KByte, welches sich fuer sehr kleine Dateien als Vorteil rausstellt. (Wie dem genau jedoch funktioniert weiss ich auch noch nicht)

19.4.1 Aufbau einer INode

Eine INode ist ein Informationsknotenpunkt für eine Datei. Jede Datei muß eine INode haben. Die Anzahl der INodes stehen also gleichzeitig für die Anzahl der möglichen Dateien auf einem Datenträger. Dabei spielt es keine Rolle wie groß diese Dateien sind. Nur in Ausßnahmefällen muß die Anzahl der INodes beachtet werden.

Die folgende Grafik zeigt den Aufbau einer INode (nur das wichtigste)

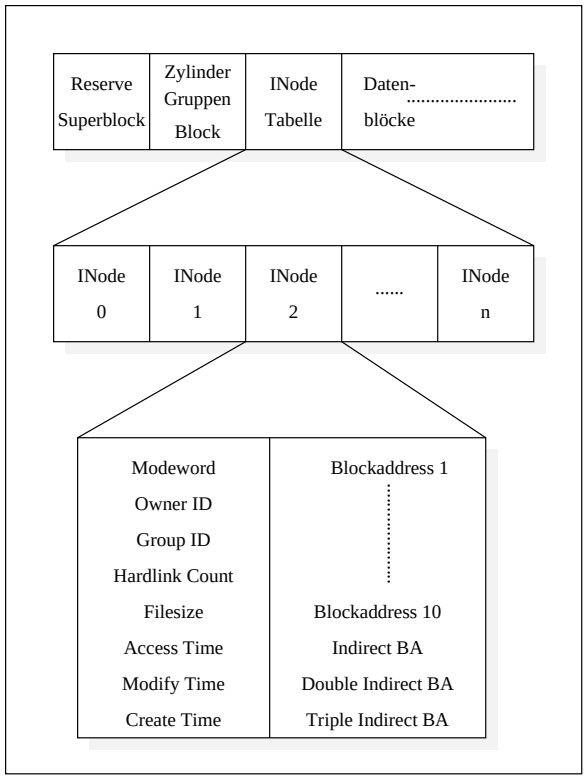


Abbildung 19.3: Basis Aufbau einer INode

Modeword Das Modeword speichert die Rechte und den Typ der Datei. Siehe dazu 8.2.1 auf Seite 34.

Owner ID Die Owner ID speichert den Eigentümer der Datei

Group ID Die Group ID speichert die Gruppenangehörigkeit der Datei

Hardlink Count Speichert die Anzahl der verweise auf eine INode. Mit dem Programm `ln`[5] können Hardlinks erstellt werden. Wird eine Datei gelöscht, wird diese erst wirklich gelöscht wenn nach der dezimierung der Hardlick Count der Zähler auf 0 (NULL) steht.

Access Time Speichert das Datum des letzen Zugriffes auf den Dateiinhalt. Dieser Zeitstempel jedoch wird nicht mehr geführt und wird beim Erstellen der Datei auf die aktuelle Uhrzeit gesetzt und dann nie wieder beachtet. Es existieren auch keine Tools mehr die auf Access Time zugreifen können

Modify Time Speichert das Datum der letzten Änderung des Dateiinhaltes. Das kann von Backupprogrammen entsprechend ausgenutzt werden. Das Programm `touch`[6] setzt das Modifydatum auf das aktuelle.

Create Time Speichert das Datum der Erstellung der INode, und somit der Datei

[5] `ln` - Siehe Kommandoreferenz Seite 303
[6] `touch` - Siehe Kommandoreferenz Seite 395

Blockaddress 1-10 In einer INode gibt es 10 Einträge die die Nummer eines Datenblockes aufnehmen können. Diese Datenblöcke speichern den Inhalt einer Datei. Man nennt die ersten 10 Blockaddressen auch *direkte Datenblockaddressen*. Die folgende Grafik verdeutlicht die Funktionsweise der Blockaddressen.

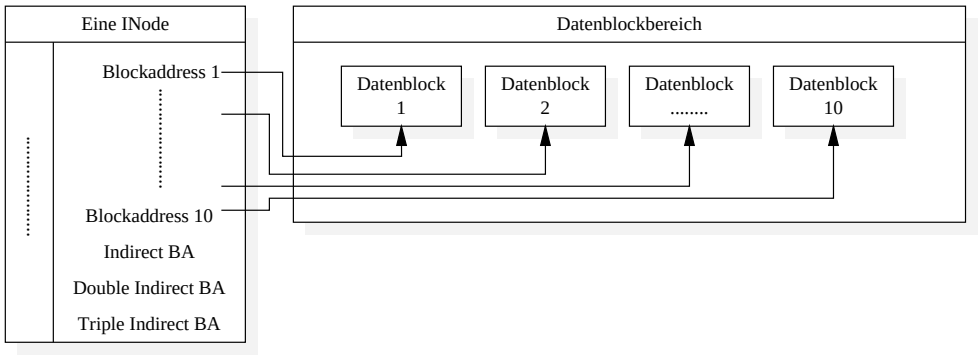


Abbildung 19.4: Direkte Datenblockaddressen

Wenn ein Datenblock die Größe von 8 KBytes besitzt dann können durch diese 10 Addressen die ersten 80 KByte einer Datei direkt eingelesen werden.

Indirect Blockaddress Die Indirekte Blockaddresse ist auch bekannt als die Blockaddresse 11 (elf). Wird die Datei größer als 80 KBytes dann kann der Mechanismus der Blockaddressen 1-10 nicht weiter geführt werden weil sonst wäre eine INode risiggross. Wenn man nun jedoch noch Blockaddressen benötigt nimmt sich das Betriebssystem einfach ein Datenblock aus dem Datenblockbereich und funktioniert diesen so um das der Datenblock keine Daten mehr enthält sondern weitere Blockaddressen. Ist ein Datenblock 8 KBytes groß und ein Eintrag einer Blockaddresse 32 bit groß ist, dann können in einem Datenblock 2048 weitere Datenblockaddressen gespeichert werden. Somit kann eine Datei dann 16 MBytes + 80 KBytes groß sein.

Die Grafik zeigt den Aufbau

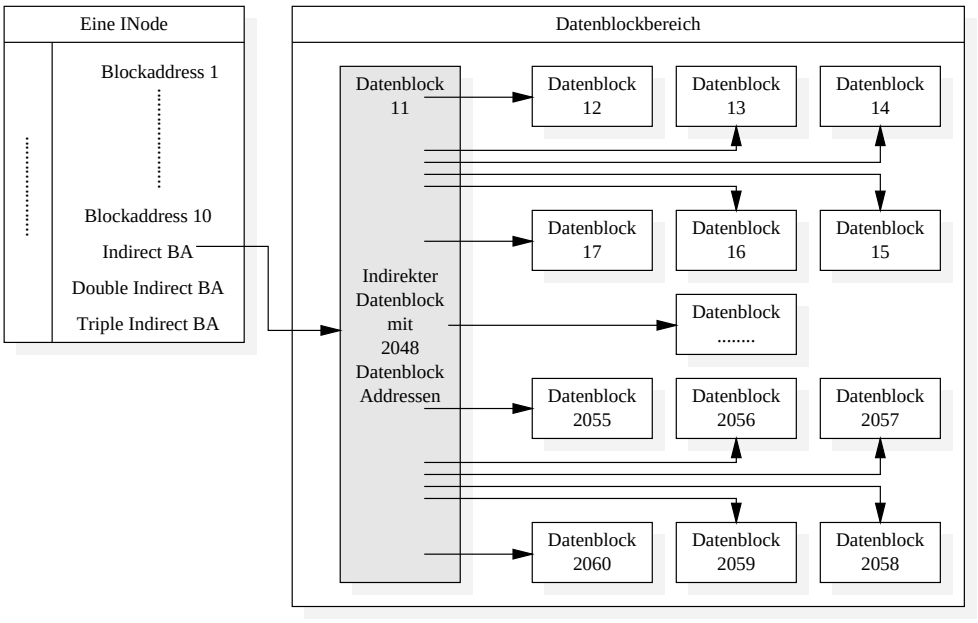


Abbildung 19.5: Blockaddresse 11

Double Indirect Blockaddress Die Doppeltindirekte Blockaddresse verweist auf ein Indirekten Datenblock der Addressen zu indirekten Datenblöcken enthält. Das ganze ist nun zweifach. Mit dieser Technik kann man mit der Blockaddresse 12 insgesammt $2048 * 2048 * 8Kbytes = 32GBytes$ ansprechen. (Plus die lumpigen 16 MBytes aus der Blockaddresse 11)

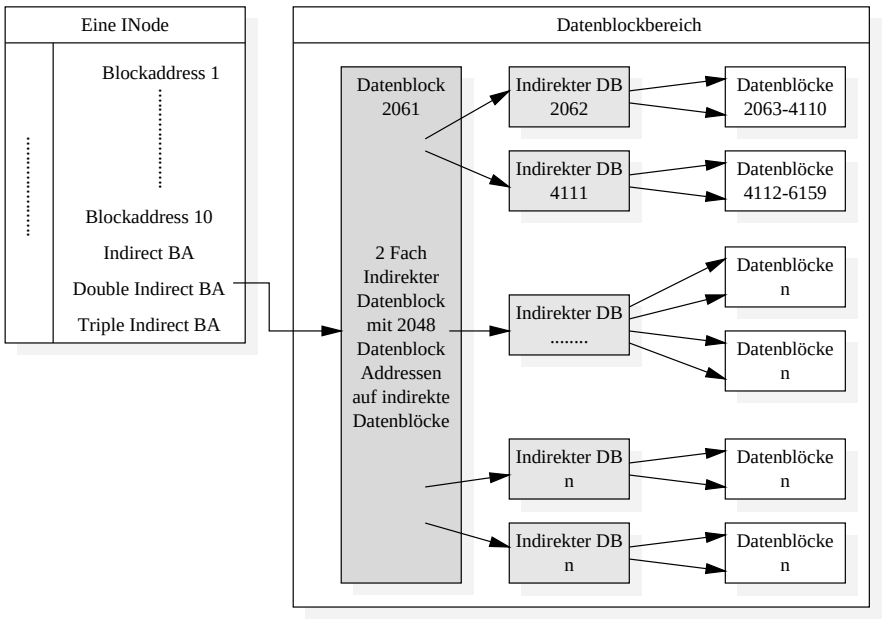


Abbildung 19.6: Blockaddresse 12

Triple Indirect Blockaddress Das gleiche in Gruen nur diesmal 3-fach Indirekt. Die Blockaddresse 13 kann also maximal $2048^3 * 8KBytes = 64TBytes$ Ansprechen.

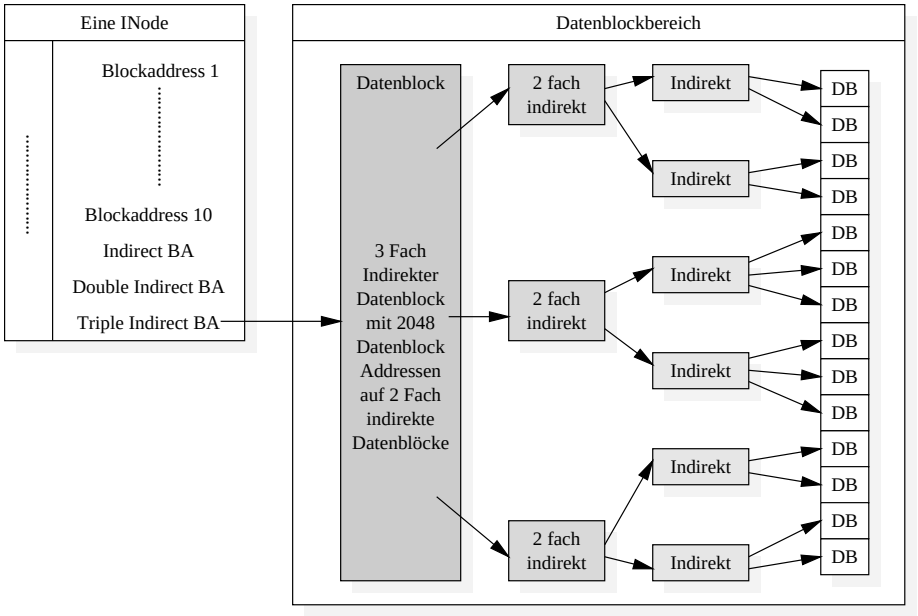


Abbildung 19.7: Blockaddresse 13

19.4.2 Aufbau einer Verzeichnissdatei

Der aufmerksame Leser hat mit sSicherheit gemerkt, daß der Dateiname nicht in der INode steht. Das ist auch korrrekt so, es würde ja auch kein Sinn machen.

Um die Dateienorganisation mittels Verzeichnissen zu lösen, wird ein Verzeichnis wie eine ganz-normale Datei behandelt. Ein Verzeichnis hat also auch einen INode Eintrag und belegt Speicher in den Datenblöcken. Unter Solaris kann man sich eine Verzeichnisdatei mittels `od`[7] auch anschauen. Eine Verzeichnisdatei ist im Endeffekt eine normale Tabelle in inetwa folgenden Aufbau besitzt :

Tabelle 19.2: Eine Verzeichnissdatei als Tabelle

INode	Dateinamenlänge	Dateiname
37	2	ls
38	4	less
39	5	touch

Man sieht anhand der Verzeichnissdatei, daß nur der Dateiname mit der INode zusammen gespeichert wird. Um den Dateityp zu ermitteln muß das Betriebssystem also zuerst die INode lesen. (Erfordert das `x` Recht auf das Verzeichniss).

Um nun ein Unterverzeichnis zu erstellen, wird in der Verzeichnissdatei ein Eintrag mit dem Namen des Unterverzeichnisses eingetragen. Das System reserviert eine INode und legt dort ein Verzeichnis an. Die Dateien `.` (punkt) und `..` (punkt-punkt) werden automatisch angelegt. Dabei beschreibt `.`

[7] `od` - Siehe Kommandoreferenz Seite 343

(punkt) das aktuelle Verzeichnis und hat somit die gleiche INode wie das Verzeichnis selbst. Die .. (punkt-punkt) Datei wird mit der INode des über geordneten Verzeichnis verbunden.

Die folgende Grafik zeigt eine Verzeichnisstruktur, wo gerade ebend das Verzeichnis subdir erstellt wurde :

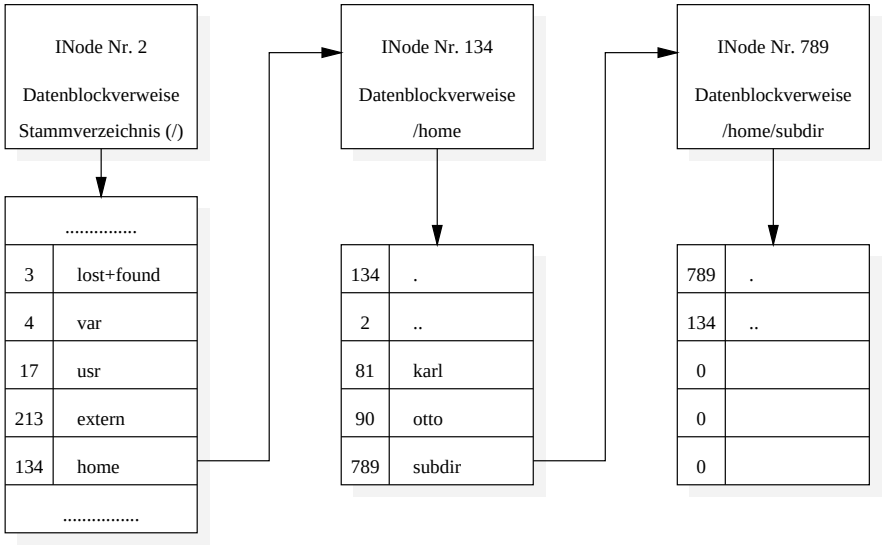


Abbildung 19.8: Verzeichnisstruktur nach der Erstellung von subdir

Wie man jetzt jedoch sieht gibt es mehrere Verweise auf die INode 134. Ein Verweis ist im Stammverzeichnis, eins im Unterverzeichnis subdir und zu guter letzt im Verzeichnis selbst. Genau diese Anzahl wird in dem Hardlink Count auch eingetragen.

19.4.3 Hardlinks

Ein Harddlink ist nichts anderes als ein weiterer Eintrag in der Verzeichnissdatei. Angegeben wird einfach ein anderer Name der Datei, die jedoch auf eine bereits existierende INode sich bezieht. Ein Hardlink kann immer nur innerhalb eines Slices angewandt werden. (Andere Slices fangen auch mit der INode 2 an). Die folgende Grafik muß nicht weiter beschrieben werden.

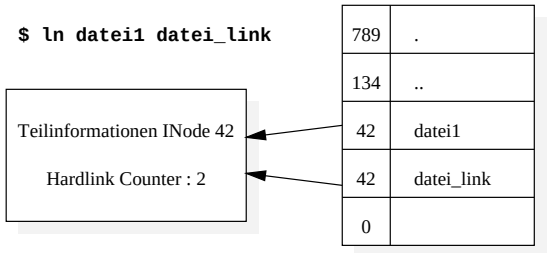


Abbildung 19.9: Ein Hardlink im Detail

19.5 Erstellen von Dateiensystemen

19.6 Prüfen von Dateiensystemen

19.7 Filesystem Tuning und Überwachung

19.8 Mounten von Dateiensystemen

19.9 Zusammenfassung

19.9.1 Tools und Programme

Tabelle 19.3: Tools und Programme für Massenspeicherverwaltung

Programm / Datei	**Bedeutung**
`prtvtoc`[8]	Schreibt den VTOC einer Platte neu
`installboot`[9]	Installiert das Bootprogramm in die Bootblöcke
`ln`[10]	Erstellen von Hard und Softlinks
`touch`[11]	Setzten der Modifytime
[12]	
[13]	
[14]	
[15]	
[16]	
[17]	
[18]	
[19]	

19.10 Praktikum

Erstellen eines Slices für /export/home !!! (nachher für Quotas wichtig)

[8] `prtvtoc` - Siehe Kommandoreferenz Seite 363
[9] `installboot` - Siehe Kommandoreferenz Seite 297
[10] `ln` - Siehe Kommandoreferenz Seite 303
[11] `touch` - Siehe Kommandoreferenz Seite 395
[12] - Siehe Kommandoreferenz Seite **??**
[13] - Siehe Kommandoreferenz Seite **??**
[14] - Siehe Kommandoreferenz Seite **??**
[15] - Siehe Kommandoreferenz Seite **??**
[16] - Siehe Kommandoreferenz Seite **??**
[17] - Siehe Kommandoreferenz Seite **??**
[18] - Siehe Kommandoreferenz Seite **??**
[19] - Siehe Kommandoreferenz Seite **??**

Kapitel 20

SYSTEMSTART

Der Systemstart eines UNIX Systems ist eine wichtige Sache. Wenn das System startet, sollten alle Dienste ihren Dienst verrichten, ohne das jemand irgendwelche Aktionen ausführen muß. Gerade im Bereich Netzwerkdienste ist es wichtig das das System wieder 100%ig einsatzfähig ist.

Das Rebooten von UNIX Maschinen ist im Normalfall nicht notwendig. Es gibt außnahmen wie das hinzufügen neuer Hardware, oder neuer Kerneleinstellungen. Aber spätestens dann wenn das System von Ihnen sauberkonfiguriert wurde sollte das System einmal ein Testboot durchlaufen.

Das Booten eines UNIX System läuft, egal welche Hardwareplattform, immer nach dem gleichen Schema ab :

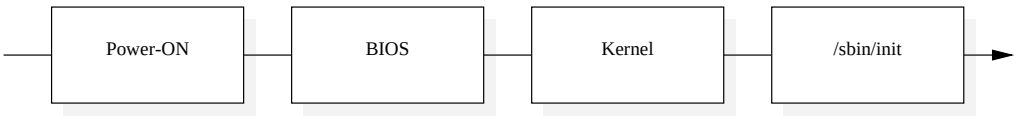


Abbildung 20.1: Allgemeiner Bootvorgang

Die einzelnen Teile wollen wir uns mal anschauen. Doch bevor wir das tun :

20.1 Terminologie

20.1.1 Startmodule

Ein Startmodul ist ein Script. Das Script hat die Aufgabe ein Dienst entweder zu starten oder zu stoppen. Alle Dienste sollten ein Startmodul besitzen, da jeder Dienst anderst gestartet oder gestoppt wird. Datenbanken z.B. benötigen zum Starten mehrere Programme. Zum Beenden müssen auch ganz gewisse Schritte durchgeführt werden, sonst kommt es zu einem Crash.

Ein Startmodul ist so aufgebaut das es immer ein Argument erwartet. Als Argument kann *start* oder *stop* zum Einsatz kommen. Wie der Name schon sagt: dient start zum Starten des Dienstes und zum Stoppen des Dienstes.

Alle Startmodule befinden sich im `/etc/init.d` Verzeichnis. Auf der Seite 114 befindet sich eine Auflistung mit den Standard-Startmodulen von Solaris 8.

Um ein solches Startmodul selber zu schreiben sollte man unbedingt folgendes Muster beibehalten :

Quelltext 20.1.1 Muster Startmodul

```
#!/bin/sh
#
# Startmodul fuer :
#

case "$1" in
start)
        echo "Starting <the name of the service>."

        /usr/sbin/foo.bar
        ;;
stop)
        echo "Shutting down <the name of the service>."

        pkill /usr/sbin/foo.bar
        ;;
*)
        echo "Unknown argument. Usage $0 {start|stop}
        exit 1
        ;;
esac

exit 0
```

Es kommt jetzt darauf an was der Dienst tatsächlich benötigt um zu startet bzw. un gestoppt zu werden. Die Ausgabe von Starting/Shutting down ist Optional und wird nur gemacht um den Benutzer zu informieren das etwas passiert.

20.1.2 Runlevel

Ein Runlevel ist die Beschreibung eines Betriebszustand eines UNIX Hosts. In älteren und in BSD Kompatiblen Systemen gibt es keine Runlevel. Das Solaris hat die Runlevel vom System V Standard übernommen.

Mit einem Runlevel kann der Betriebszustand des Hosts ganz einfach gewechselt werden. Da man in einem Runlevel verschiedene Dienste zum laufen bringen kann, kann man ein System innerhalb von Sekunden zu einem fast vollkommenen anderen Rechner machen. Man nutzt die Runlevels für Server nicht aus da diese Rechner eh nur ein Ziel haben. Bei Notebooks, Laptops jedoch kann man Runlevel definieren, der eine ist für zu Hause, der andere für das Büro, z.B.

Ein Runlevel wird mit einem einzelnen Zeichen beschriftet. Unter Solaris gibt es die Runlevel `0,1,2,3,4,5,6,s,S`. Andere UNIX System definieren noch ein paar mehr, oder weniger. Jeder Runlevel hat eine besondere Bedeutung, in jedem Runlevel werden die entsprechenden Dienste gestartet die nötig sind um das System in den entsprechenden Betriebszustand zu bringen. Die Runlevels im Detail :

Tabelle 20.1: Bedeutungen und Übersicht der Runlevel

Runlevel	Name	Bedeutung
`s,S`	singleuser	Single-User Modus. Wird das System in den Runlevel `s` gebracht, werden alle Prozesse runtergefahren. Alle Netzresorcen werden entfernt. Lokal gemountete Laufwerke werden entfernt. Der Loginmanager auf der Console wird auch entfernt. Es wird eigendlich alles entfernt. Nach den Aktionen muß man das Rootpassword eingeben und man bekommt eine Shell aufgemacht. Nun ist man relativ allein und man kann als Root machen was man will. Dieser Modus wird meist für sehr aufwendige Konfigurationsarbeiten benutzt. Mit `init S` kann man in den Runlevel wechseln. Oder man gibt beim Booten die Option *-s* an.
`0`	halt	Alle Dienste und Prozesse werden runtergefahren und gekillt. Alle Dateiensysteme werden geunmountet inkl. des Rootdevices. Danach ist das System angehalten. Auf einem Intelsystem steht danach unten eine Nachricht, man möge doch bitte RETURN drücken. Auf SPARCs kommt man so ins OBP. Durch `init 0` oder `halt` kommt man in den Runlevel hinein
`1`	admin	In diesem Runlevel sind alle lokalen Devices gemountet. Benutzer jedoch können sich nicht anmelden. Dieser Modus nennt sich Administrator Modus. Er kann zur Installation von Software dienen, sofern die Software es verlangt. Durch `init 1` kommt man in den Runlevel hinein
`2`	multiuser	Fast der Standardrunlevel. Er bringt das System in den Multiusermodus und macht es Netzwerkfähig. Durch `init 2` kommt man in den Runlevel hinein
`3`	multiuser / network	Der Runlevel 3 allein ist nutzlos. Er muß im Zusammenhang mit dem 2 Runlevel gestartet werden (Das passiert jedoch automatisch, siehe inittab). Der Runlevel 3 erweitert den Runlevel 2 um einige Dienste die nötig sind, damit das System auch als Server arbeiten kann. Durch `init 3` kommt man in den Runlevel hinein
`4`	unused	Unbenutzt und sollte ohne ihn vorher zu konfigurieren NICHT starten.
`5`	poweroff	Fährt das System sauber Runter und sofern die Option vorhanden ist schaltet der Host sich aus. Durch `init 5` oder `poweroff` kommt man in den Runlevel hinein
`6`	reboot	Macht das gleiche wie der Runlevel 5, jedoch rebootet er neu Durch `init 6` oder `reboot` kommt man in den Runlevel hinein

Ein Runlevelwechsel kann immer dann stattfinden, wenn der Administrator der Meinung ist: es wäre soweit.

Mit dem Programm `who`[1] kann der aktuell eingestellte Runlevel angezeigt werden :

[1] `who` - Siehe Kommandoreferenz Seite 409

```
$ who -rH
NAME       LINE         TIME           IDLE    PID  COMMENTS
   .       run-level 3  Jun 10 10:27     3      0  S
$
```

Bildschirmausschnitt 20.1.1: Anzeigen des aktuellen Runlevels

20.1.3 Runlevel Verzeichnis

Damit das Runlevelscript weiß welche Module zum Start/Stop eines Runlevels gehören gibt es die Runlevelverzeichnisse. Fast jeder Runlevel besitzt ein eigenes Runlevelverzeichnis. In diesem Verzeichnis werden die Start-Scripts und Kill-Scripts erstellt.

Tabelle 20.2: Übersicht der Runlevelverzeichnisse

Runlevel	Runlevelverzeichnis
S	/etc/rcS.d
0	/etc/rc0.d (obsolete: /etc/shutdown.d)
1	/etc/rc1.d
2	/etc/rc2.d
3	/etc/rc3.d
4	/etc/rc4.d
5	/etc/rc0.d (obsolete: /etc/shutdown.d)
6	/etc/rc0.d (obsolete: /etc/shutdown.d)

Da die Runlevel 0, 5, 6 primär das Runterfahren des Rechners erledigen, benutzen alle drei ein und das gleiche Runlevelverzeichnis.

20.1.4 Start-Stop-Scripts

Alle Start-Stop-Scripts liegen in den jeweiligen Runlevelverzeichnissen, damit der Runlevel weiss was er starten, bzw. Stoppen soll.

Die Start-Scripts werden kenntlich gemacht am Dateinamen. Die Start-Scripts fangen immer mit einem großen `S` an. Um festzustellen welche Dienste in einem Runlevel gestartet werden reicht ein `ls S*` im Runlevelverzeichnis aus. Die Kill-Scripts fangen mit einem großen `K` an.

Um auch die Reihenfolge der Scripte festzulegen werden einfach zweistellige Nummern hinter dem `K` bzw. hinter dem `S` gesetzt. Um sicherzustellen das die Start/Stop-Reihenfolge sichergestellt wird. Weil man sollte zuerst das Netzwerk starten, bevor man versucht eine Resource aus dem Netzwerk anzufordern.

Nach der Nummer ist es egal was kommt. Meist wird hier der Name des Modules eingetragen um stäter zu wissen das das für ein Modul ist.

Ein vollständiger Dateiname eines Scriptes im Runlevelverzeichnis wäre z.B. `S22acct` oder `K12apache`. Somit weiß man ob es gestartet oder gestoppt wird. In welcher Reihenfolge und dessen Name.

Alle Start-Script und Stop-Scripts sind Links auf die eigentlichen Startmodule. Wobei es im allgemeinen egal ist ob es sich um ein Hardlink oder ein Softlink handelt. Ich plediere für Softlinks.

Die folgende Grafik soll es noch einmal visualisieren :

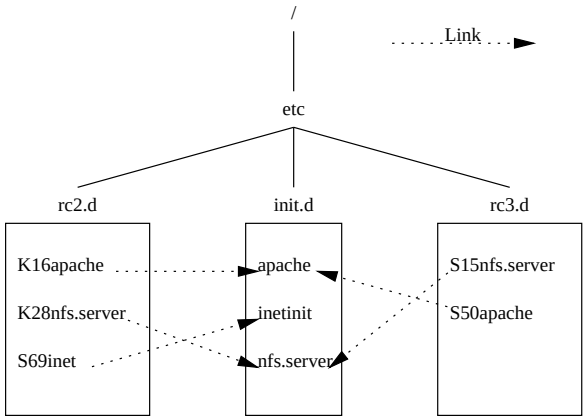


Abbildung 20.2: Die Links

20.1.5 Runlevel Script

Ein Runlevel Script ist verantwortlich für das Laden und Entladen aller Startmodule die für den Runlevel benötigt werden. Es gibt für jeden Runlevel ein entsprechendes Script. Wenn man in einen Runlevel wechselt, dann wird das Runlevelscript aufgerufen. Es wechselt in das Runlevelverzeichnis und startet dort alle Stop-Scripts. Danach startet es alle Start-Scripts. Diesen Lauf kann man nun beeinflussen indem man im Runlevelverzeichnis Veränderungen vornimmt.

/sbin/rc0 Zum einen geht es aus historischen Gründen in das `/etc/shutdown.d` Verzeichnis und startet dort ALLE Scripts. Zum zweiten geht er in das `/etc/rc0.d` Verzeichnis und Startet dort alle Stop-Scripts. Danach startet er alle Start-Scripts in `/etc/rc0.d` und hält das System an

/sbin/rc1 `/sbin/rc1` benutzt für das Runlevelverzeichnis das `/etc/rc1.d` startet vor den Wechsel in den Runlevel 1 dort alle Stop-Scripts danach alle Start-Scripts

/sbin/rc2 `/sbin/rc2` benutzt für das Runlevelverzeichnis das `/etc/rc2.d` startet vor den Wechsel in den Runlevel 2 dort alle Stop-Scripts danach alle Start-Scripts

/sbin/rc3 `/sbin/rc3` benutzt für das Runlevelverzeichnis das `/etc/rc3.d` startet vor den Wechsel in den Runlevel 3 dort alle Stop-Scripts danach alle Start-Scripts

/sbin/rc5 `/sbin/rc5` ist ein Hardlink auf `/sbin/rc0`

/sbin/rc6 `/sbin/rc6` ist ein Hardlink auf `/sbin/rc0`

/sbin/rcS `/sbin/rcS` hebt sich etwas ab, da es auch zur Systeminitialisierung benutzt wird. Es benutzt das `/etc/rcS.d` Verzeichnis. Wird es jedoch mit dem Argument *sysinit* aufgerufen, dann handelt es sich um die Systeminitialisierung.

Dokumentiertes Beispiel des `/sbin/rc2` Scripts

Quelltext 20.1.2 Dokumentiertes `/sbin/rc2` *(Teil 1)*

```
#!/sbin/sh
#
# Copyright (c) 1984, 1986, 1987, 1988, 1989 AT&T.
# All rights reserved.
#
# THIS IS UNPUBLISHED PROPRIETARY SOURCE CODE OF AT&T
# The copyright notice above does not evidence any
# actual or intended publication of such source code.
#
# Copyright (c) 1997-1999 by Sun Microsystems, Inc.
# All rights reserved.
#
#ident        "@(#)rc2.sh        1.16        99/02/23 SMI"

# Run Commands executed when the system is changing to init state 2,
# traditionally called "multi-user". Pick up start-up packages for mounts,
# daemons, services, etc.

PATH=/usr/sbin:/usr/bin

# Export boot parameters to rc scripts

set -- `/usr/bin/who -r`

_INIT_RUN_LEVEL="$7"        # Current run-level
_INIT_RUN_NPREV="$8"        # Number of times previously at current run-level
_INIT_PREV_LEVEL="$9"        # Previous run-level

set -- `/usr/bin/uname -a`

_INIT_UTS_SYSNAME="$1"        # Operating system name (uname -s)
_INIT_UTS_NODENAME="$2"        # Node name (uname -n)
_INIT_UTS_RELEASE="$3"        # Operating system release (uname -r)
_INIT_UTS_VERSION="$4"        # Operating system version (uname -v)
_INIT_UTS_MACHINE="$5"        # Machine class (uname -m)
_INIT_UTS_ISA="$6"        # Instruction set architecture (uname -p)
_INIT_UTS_PLATFORM="$7"        # Platform string (uname -i)

export _INIT_RUN_LEVEL _INIT_RUN_NPREV _INIT_PREV_LEVEL \
    _INIT_UTS_SYSNAME _INIT_UTS_NODENAME _INIT_UTS_RELEASE _INIT_UTS_VERSION \
    _INIT_UTS_MACHINE _INIT_UTS_ISA _INIT_UTS_PLATFORM
```

Der erste Teil beschäftigt sich mit der Standarisierung aller Parameter die benötigt werden. Es wird der `set` Mechanismus verwendet um die einzelnen Parameter rauszufiltern. Der Vorteil liegt im Detail: man benötigt keine externen Programme wie `cut` oder `tr`. Vielleicht sind diese Programme noch garnicht verfügbar und müssen erst noch gemountet werden. An den Bemerkungen kann man erkennen um welchen Inhalt es sich bei den einzelnen Variablen handelt.

Quelltext 20.1.3 Dokumentiertes `/sbin/rc2` *(Teil 2)*

```
# Export net boot configuration strategy. _INIT_NET_IF is set to the
# interface name of the netbooted interface if this is a net boot.
# _INIT_NET_STRATEGY is set to the network configuration strategy.
set -- '/sbin/netstrategy'
if [ $? -eq 0 ]; then
        if [ "$1" = "nfs" -o "$1" = "cachefs" ]; then
                _INIT_NET_IF="$2"
        fi
        _INIT_NET_STRATEGY="$3"
        export _INIT_NET_IF _INIT_NET_STRATEGY
fi
```

Im zweiten Teil wird geprüft wie das System hochgefahren wurde. Das Programm `netstrategy`[2] liefert 3 Parameter zurück. Im ersten wird das Filesystem des Rootdevice ausgegeben. Wenn es sich um NFS oder CacheFS handelt, dann muß das System über Netz gebootet worden sein. Ansonsten steht dort *ufs*. In der zweiten Ausgabe steht dann auch die Netzwerkkarte über die gebootet worden ist. In der dritten Ausgabe steht die Strategie. Das kann sein *none* wenn es lokal gebootet wurde oder *dhcp* bzw. *rarp*.

Diese Variablen können in den Startmodulen abgefragt werden. Das Startmodul `/etc/init.d/inetinit` und `/etc/init.d/inetsvc` benötigt diese Angabe um die Netzwerkkarte richtig zu konfigurieren.

[2] `netstrategy` - Siehe Kommandoreferenz Seite **??**

Quelltext 20.1.4 Dokumentiertes `/sbin/rc2` *(Teil 3)*

```
if [ $_INIT_PREV_LEVEL = S -o $_INIT_PREV_LEVEL = 1 ]; then
        echo 'The system is coming up.  Please wait.'
elif [ $_INIT_RUN_LEVEL = 2 ]; then
        echo 'Changing to state 2.'

        if [ -d /etc/rc2.d ]; then
                for f in /etc/rc2.d/K*; do
                        if [ -s $f ]; then
                                case $f in
                                        *.sh) . $f ;;
                                        *)    /sbin/sh $f stop ;;
                                esac
                        fi
                done
        fi
fi
```

Im dritten Teil prüft das Script ob es die Kill-Scripts aufrufen soll. War der vorherige Runlevel entweder `S` oder `1` dann muß er keine Kill-Scripts mehr aufrufen. Danach prüft das Script ob der Wechsel in den Runlevel `2` stattfinden soll. Ist dem so dann prüft es das vorhanden sein des Runlevelverzeichnisses, und ruft alle Kill-Scripts dort auf. Es prüft ob alle Scripts existieren und größer als 0 Bytes sind. Wie man dort erkennt werden die Kill-Scripts die mit `.sh` enden nicht mit dem Argument *stop* aufgerufen.

Quelltext 20.1.5 Dokumentiertes `/sbin/rc2` *(Teil 4)*

```
if [ $_INIT_PREV_LEVEL != 2 -a $_INIT_PREV_LEVEL != 3 -a -d /etc/rc2.d ]; then

        for f in /etc/rc2.d/S*; do
                if [ -s $f ]; then
                        case $f in
                                *.sh)  . $f ;;
                                *)     /sbin/sh $f start ;;
                        esac
                fi
        done
fi
```

Im vierten Teil wird geprüft ob die Start-Scripts ausgeführt werden sollen oder nicht. Wenn der vorherige Runlevel `2` oder `3` war und wenn das Runlevelverzeichnis nicht existiert, dann werden die Startscripts nicht nochmal aufgerufen.

Quelltext 20.1.6 Dokumentiertes `/sbin/rc2` *(Teil 5)*

```
#if [ ! -s /etc/rc2.d/.ports.sem ]; then
#        /sbin/ports
#        echo "ports completed" > /etc/rc2.d/.ports.sem
#fi
```

Der fünfte Teil ist auf dem älteren Solaris Versionen. Das Programm `/sbin/ports` initialisierte die Gerätedateien im `/dev` Verzeichnis für die seriellen Ports. Seit Solaris 7 oder 8 wird diese Aufgabe vom Programm `devfsadm` gelöst, welches vom Startmodul `/etc/init.d/devfsadm` entsprechend aufgerufen wird.

Quelltext 20.1.7 Dokumentiertes `/sbin/rc2` *(Teil 6)*

```
# Start historical section

if [ \( $_INIT_PREV_LEVEL = S -o $_INIT_PREV_LEVEL = 1 \) \
    -a -d /etc/rc.d ]; then

        for f in `/usr/bin/ls /etc/rc.d`; do
                [ ! -s /etc/init.d/$f ] && /sbin/sh /etc/rc.d/$f
        done
fi

# End historical section
```

Der sex-te Teil ist Geschichte. Sollte der vorherige Runlevel `S` oder `1` gewesen sein werden alle Scripts im `/etc/rc.d` Verzeichnis gestartet die keine Entsprechung im `/etc/init.d` Verzeichnis besitzen. Das `/etc/rc.d` Verzeichnis existiert jedoch auch nicht mehr unter Solaris.

Quelltext 20.1.8 Dokumentiertes `/sbin/rc2` *(Teil 7)*

```
if [ $_INIT_RUN_LEVEL = 2 ]; then
        if [ $_INIT_PREV_LEVEL = S -o $_INIT_PREV_LEVEL = 1 ]; then
                echo 'The system is ready.'
        else
                echo 'Change to state 2 has been completed.'
        fi
fi
```

Der siebte teil sorgt nur für eine hübsche Meldung auf dem Bildschirm in abhängigkeit wie man in den Runlevel `2` eingestiegen ist.

UND WER KUEMMERT SICH UM DEN START DER RUNLEVEL SCRIPTS ????

20.2 Bootvorgang auf SPARC Rechnern

BAUSTELLEN VERKEHR. LASSEN SIE BITTE KEINE GEGENSTAENDE RUMLIEGEN

20.3 Bootvorgang auf Intel Rechnern

20.4 Kernel

Wenn der Kernel geladen wurde, initialisiert er das System. Dazu liest er die /etc/system[3] ein um seine Kernelparameter einzustellen. Wenn man die /etc/system verändert sollte man mit aller höchster Sorgfalt dieses tun und das System neu booten.

Wird der Kernel gestartet springt er in ein nicht definierten Runlevel, den man auch System-Initialisierungs-Runlevel nennen kann. Wird dem Kernel jedoch beim Booten ein Argument mit übergeben kann der Kernel gleich in den Runlevel `s` springen. Auf SPARCs ist ein `ok boot -s` nötig.

Der Kernel mountet danach das Rootdevice und startet danach das Programm `/sbin/init`.

20.5 /sbin/init

Das Programm `init` ist und bleibt auch der erste Userprozeß des UNIX Systems. `init` überwacht den Betriebszustand und sorgt dafür das die Prozesse gestartet werden sollen die gestartet werden müssen. Wenn `init` gestartet wird stellt es zu erst einmal fest ob sich bereits ein `init` Prozeß im System befindet. Wenn das der Fall ist veranlasst die Kopie von `init` den Runlevelwechsel. Stellt `init` jedoch fest das es älleinïst springt es erst einmal in den Runlevel *system-init*.

20.5.1 Environment von `init`

Dazu liest `init` als erstes die /etc/default/init[4] um das Environment zusetzen. In dieser Datei steht die Zeitzone, die Landeseinstellungen und die Default umask.

20.5.2 Konfigurationsdatei von `init`

Danach liest er die /etc/inittab[5] ein und startet dort die entsprechenden Programme. Eine Normale /etc/inittab sieht ca. so aus :

[3] /etc/system - Siehe Konfigurationsdateien Referenz Seite 433
[4] /etc/default/init - Siehe Konfigurationsdateien Referenz Seite 417
[5] /etc/inittab - Siehe Konfigurationsdateien Referenz Seite 425

Quelltext 20.5.1 Original /etc/inittab von Solaris

```
ap::sysinit:/sbin/autopush -f /etc/iu.ap
ap::sysinit:/sbin/soconfig -f /etc/sock2path
fs::sysinit:/sbin/rcS sysinit >/dev/msglog 2<>/dev/msglog </dev/console
is:3:initdefault:
p3:s1234:powerfail:/usr/sbin/shutdown -y -i5 -g0 >/dev/msglog 2<>/dev/msglog
sS:s:wait:/sbin/rcS >/dev/msglog 2<>/dev/msglog </dev/console
s0:0:wait:/sbin/rc0 >/dev/msglog 2<>/dev/msglog </dev/console
s1:1:respawn:/sbin/rc1 >/dev/msglog 2<>/dev/msglog </dev/console
s2:23:wait:/sbin/rc2 >/dev/msglog 2<>/dev/msglog </dev/console
s3:3:wait:/sbin/rc3 >/dev/msglog 2<>/dev/msglog </dev/console
s5:5:wait:/sbin/rc5 >/dev/msglog 2<>/dev/msglog </dev/console
s6:6:wait:/sbin/rc6 >/dev/msglog 2<>/dev/msglog </dev/console
fw:0:wait:/sbin/uadmin 2 0 >/dev/msglog 2<>/dev/msglog </dev/console
of:5:wait:/sbin/uadmin 2 6 >/dev/msglog 2<>/dev/msglog </dev/console
rb:6:wait:/sbin/uadmin 2 1 >/dev/msglog 2<>/dev/msglog </dev/console
sc:234:respawn:/usr/lib/saf/sac -t 300
co:234:respawn:/usr/lib/saf/ttymon -g...$<stuff deleted>$...m,ttcompat
```

Da `init` zu erst in den Runlevel *sysinit* fällt startet es die

20.5.3 Der Lauf in den *sysinit* Runlevel

Programme `autopush`, `soconfig` und `rcS`.

autopush Konfiguriert die STREAMs und benutzt die Konfigurationsdatei /etc/ip.ap. Der Eintrag sollte nicht geändert werden.

soconfig Konfiguriert die Transportschnittstellen für das Netzwerk. Als Konfigurationsdatei kommt die /etc/sock2path zum Einsatz. Auch das sollte nicht geändert werden.

rcS Startet die Startmodule (siehe unten). Auch das sollte nicht geändert werden.

20.5.4 Festlegung des Default Runlevels

Das macht `init` danach. Normalerweise ist der Runlevel 3 Standard. Wenn er in den Runlevel 3 springt führt er auch hier wieder alle Programme der Reihe nach aus. Zubeachten ist daß der Runlevel 3 wohl das Script `rc2` als auch das Script `rc3` startet.

Die letzten beiden Eintrage sorgen dafür das man sich an der Console anmelden kann. Siehe dazu das Kapitel über Terminalverwaltung/SAF.

20.6 Zusammenfassung

20.6.1 Tools und Programme

Tabelle 20.3: Tools und Programme zum Systemstart

Programm / Datei	**Bedeutung**
who[6]	Anzeige der Initprozesse und des Runlevels
init[7]	Manager der Prozesse und Runlevelwechsel
halt[8]	Wechselt in den Runlevel 0 ohne jedoch rcS auszuführen
reboot[9]	Wechselt in den Runlevel 6
shutdown[10]	Wechselt in den angegebenen Runlevel. Kann vorher noch warten und eine Meldung an alle eingeloggten Benutzer ausgeben
poweroff[11]	Wechselt in den Runlevel 5
uadmin[12]	Programm zur Simulation von Systemcalls. Wird verwendet um z.B. zu Rebooten oder zu PowerOffen
/sbin/rc$< x >$	Runlevelscript

20.6.2 Konfigurationsdateien und Scripts

Tabelle 20.4: Config und Scripts zum Systemstart

Programm / Datei	**Bedeutung**
/etc/system[13]	In der /etc/system werden Kernelparameter eingestellt
/etc/inittab[14]	Das Program init verwendet die /etc/inittab als Konfigurationsdatei. In ihr liegen die Informationen was wie in einem Runlevel gestartet werden soll
/etc/default/init[15]	Zeitzone, Landeseinstellungen und Maske für init
/etc/init.d	Startmodulverzeichnis
/etc/rc$< x >$.d	Runlevelverzeichnis

20.6.3 Liste aller Standard-Startmodule

[6] who - Siehe Kommandoreferenz Seite 409
[7] init - Siehe Kommandoreferenz Seite 295
[8] halt - Siehe Kommandoreferenz Seite 285
[9] reboot - Siehe Kommandoreferenz Seite 373
[10] shutdown - Siehe Kommandoreferenz Seite 387
[11] poweroff - Siehe Kommandoreferenz Seite 361
[12] uadmin - Siehe Kommandoreferenz Seite **??**
[13] /etc/system - Siehe Konfigurationsdateien Referenz Seite 433
[14] /etc/inittab - Siehe Konfigurationsdateien Referenz Seite 425
[15] /etc/default/init - Siehe Konfigurationsdateien Referenz Seite 417

20.7 Praktikumsaufgaben

1. Schreiben Sie ein mini-script mit dem Namen `runlevel`. Das Script soll nur den aktuell laufenden Runlevel ausgeben, mehr nicht.

2. Schreiben Sie ein Script mit dem Namen `runprocess`. Das Script soll mit einem Runlevelbezeichner aufgerufen werden. Das Script soll darauf hin alle Prozesse in diesem Runlevel mit dessen Aktion ausgeben.

3. Schreiben Sie ein Startmodul mit dem Namen `rwho`. Das Startmodul soll den Dienst `rwhod` starten bzw. Stoppen. Das ist zuerst zu Testen. Starten Sie Das Script einmal mit dem Argument *start* und einmal mit *stop*. Zubeachten ist das `rwhod` das Verzeichnis `/var/spool/rwho` benötigt.

4. Legen Sie nun die benötigten Links an um den Dienst im Runlevel 3 zu starten um beim Runterfahren den Dienst zu stoppen. Testen Sie es indem Sie Rebooten.

Lösungsvorschläge

1. Es gib mehrere Lösungen. Man kann z.B. die Ausgaben von `who -r` durch die Pipe jagen und ersteinmal alle doppelten Leerzeichen entfernen und sich dann die 4. Spalte holen (Das vordere Leerzeichen wird nicht entfernt) :

Quelltext 20.7.1 Lösungsansatz für runlevel (Lösung 1)
```
#!/bin/sh
who -r | tr -s " " | cut -d" " -f 4
```

   oder man geht hin und übergibt dem laufenden Script einfach die Ausgabe von who als Argumente. Im Argument 3 landet dann der Inhalt

Quelltext 20.7.2 Lösungsansatz für runlevel (Lösung 2)
```
#!/bin/sh
set - ‘who -r‘
echo $3
```

   So wird es normalerweise gemacht, da es keine externen Programme wie `cut` oder `tr` benötigt. Für die Harten kann auch der awk genutzt werden :

Quelltext 20.7.3 Lösungsansatz für runlevel (Lösung 3)
```
#!/bin/sh
who -r | awk ’{ print $3; }’
```

   Ok, es gibt wohl 1 Mrd. Lösungen.

2. Die Shellvariante :

Quelltext 20.7.4 Lösungsansatz für runprocess (Lösung 1)
```
#!/bin/sh
if test $# -ne 1; then
        echo "Usage: $0 <runlevel>"
        exit 1
fi
grep ’^[^:]*:[^:]*’"$1"’[^:]*:’ /etc/inittab | cut -d: -f3-
```

   Wobei das `grep` auch anderst gemacht werden könnte :

Quelltext 20.7.5 Lösungsansatz für runprocess (Lösung 2)

```
#!/bin/sh
if test $# -ne 1; then
        echo "Usage: $0 <runlevel>"
        exit 1
fi

IFS=":
"

cat /etc/inittab | while read id rstate action process; do
        case "$rstate" in
        *$1*) echo $action:$process; ;;
        esac
done
```

Allerdings hat diese Lösung leichte Fehler und kann nicht für echte administratorische Aufgaben verwendet werden.

3. Das Startmodul könnte wie folgt aussehen

Quelltext 20.7.6 Lösungsansatz für Startmodul rwho

```
#!/bin/sh
#
# Startmodul f"ur den Dienst rwhod
#
case "$1" in
start)
        if test ! -d /var/spool/rwho; then
                mkdir -p -m 755 /var/spool/rwho
        fi
        /usr/sbin/in.rwhod
        ;;
stop)
        pkill /usr/sbin/in.rwhod
        ;;
*)
        echo "Usage $0 {start|stop}"
exit 1
        ;;
esac
exit 0
```

4. `ln -s /etc/init.d/rwho /etc/rc3.d/S20rwho`
   `ln -s /etc/init.d/rwho /etc/rc0.d/K10rwho`

Kapitel 21

TERMINALKONFIGURATION

System V und BSD Systeme unterscheiden sich sehr im Zusammenhang mit Terminals und dessen Zugriff. Das ältere SunOS 4.x arbeitet noch mit dem BSD Standard. Auch Linux arbeitet nach dem BSD Standard. Wir schauen uns mal das System V das SAF[1] an.

21.1 SAF

Das SAF System kann mehrere Serielle Ports mit Diensten belegen und Managen. Gerade dann wenn serielle Terminals zum Einsatz kommen wird SAF verwendet. Um Multiuserfähigkeiten auszunutzen, auch ohne Netzwerk, werden meist die Rechner mit Seriellen-Multiportkarten bestückt, z.B. 16 oder 32 Ports pro Karte, an die man dann serielle Terminals hängen kann. Somit wird ein System Multiuserfähig. Diese Technik wurde früher bei Mainframes verwendet. Heute ist es jedoch fast zweitrangig, da jeder Rechner eine Netzwerkkarte besitzt.

SPARC Maschinen unterstützen die Administration des OBPs über serielle Leitungen, somit ist der Rechner komplett fernwartbar und benötigen weder Grafikkarte noch ein Monitor oder Tastatur und eine Maus schon garnicht.

21.1.1 Grundlegender Aufbau von SAF

Der Aufbau von SAF zeigt die folgende Grafik :

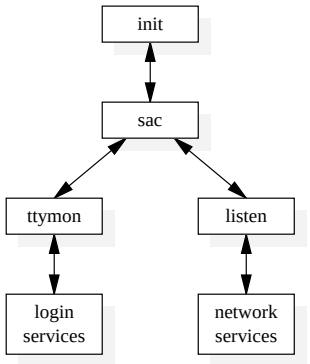


Abbildung 21.1: Grundlegenderaufbau von SAF

Man erkennt das `sac` von `init` gestartet wird. Und `sac` die weiteren Konfigurationen von `ttymon` und `listen` selbst durchführt. Jeder dieser Komponenten hat auch eine Funktion :

sac Der SAC[2] verwaltet die gesammte Konfiguration des SAF Systems

[1] SAF-Service Access Facility
[2] SAC-Service Access Controller

ttymon Der `ttymon` überwacht die angegebenen Ports und stellt dem nachfolgendem Prozeß ein Stream zur Verfügung

listen Der `listen` wurde auf älteren Solaris Versionen eingesetzt um dort z.B. Druckerrequests aus dem Netzwerk zu empfangen. Der Daemon ist jedoch mehr oder wendiger duch den `inetd` abgelöst worden

21.2 Der SAC

Der `sac`[3] hat die Aufgabe alle Monitore zu verwalten. Wobei der Begriff Monitor nicht mit einem Monitor(Sichtschirm) zu verwechseln ist. Ein Monitor hat die Aufgabe auf ein bestimmtes Ereignis zu warten um danach diverse Aktionen auszuführen und eventuell andere Prozesse zu starten.

Der `sac` muß vom `init` Prozeß gestartet werden. Dazu sollte ein entsprechender Eintrag in der /etc/inittab[4] vorhanden sein. Etwa so :

Quelltext 21.2.1 Eintragungen in der /etc/inittab für `sac`

```
sc:234:respawn:/usr/lib/saf/sac -t 300
```

Wenn der `sac` gestartet wird liest er seine Konfigurationsdatei /etc/saf/_sysconfig ein. Die solltem man jedoch nicht manuell bearbeiten.

Danach liest er die /etc/saf/_sactab ein und startet dort alle aufgeführten Portmonitore. Auch diese Datei sollte man nicht Manuell verändern. Der Start erfolgt so, daß `sac` sich als Parentprocess der Portmonitore einträgt.

Danach verfällt er in eine Warteschleife und prüft ständig das noch vorhanden sein seiner Portmonitore und startet diese gegebenfalls neu.

21.3 Konfiguration des SAC

Um den `sac` zu konfigurieren existiert das `sacadm`[5]. Eine Liste aller Konfigurierten Portmonitore kann man mittels `sacadm -l` bekommen.

[3] `sac` - Siehe Kommandoreferenz Seite 379
[4] /etc/inittab - Siehe Konfigurationsdateien Referenz Seite 425
[5] `sacadm` - Siehe Kommandoreferenz Seite 381

Quelltext 21.3.1 Beispiel Konfiguration eines Portmonitors

```
eagle # sacadm -l
PMTAG     PMTYPE    FLGS RCNT STATUS     COMMAND
zsmon     ttymon    -    0    ENABLED    /usr/lib/saf/ttymon #
eagle # sacadm -r -p zsmon
eagle # sacadm -l
No port monitors defined
eagle # sacadm -a -p mytty -t ttymon -c /usr/lib/saf/ttymon
        -v ‘ttyadm -V‘ -y "Mein TTY Monitor"
eagle # sacadm -l
PMTAG     PMTYPE    FLGS RCNT STATUS     COMMAND
mytty     ttymon    -    0    STARTING   /usr/lib/saf/ttymon #....
eagle #
```

Nach einer Weile wird aus dem *STARTING* ein *ENABLED*. Das Beispiel fügt nun ein neuen Portmonitor hinzu. Der Standard von Solaris ist deaktiviert, deswegen löschen wir alles.

21.4 Die Verbindung vom Portmonitor zum Service

Um einem Portmonitor zu sagen was er zu tun hat benötigt man das Programm `pmadm`[6]. Dem Programm muß man mitteilen welchen Portmonitor er mit welchem Servive verbinden soll. Auch hier kann man sich eine Liste aller Portmonitorverbindungen mit cmdpmadm -l anzeigen lassen.

Quelltext 21.4.1 Beispiel Konfiguration einer Portmonitor-Service Verbindung

```
eagle # pmadm -l
No services defined
eagle # pmadm -a -p mytty -s ttyb -i root -v ‘ttyadm -V‘
        -f u -m "‘ttyadm -l 9600 -s /usr/bin/login -d /dev/term/b‘"
eagle # pmadm -l
PMTAG       PMTYPE      SVCTAG      FLGS ID       <PMSPECIFIC>
mytty       ttymon      ttyb        u    root     /dev/term/b ....
                 .... - - /usr/bin/login - 9600 - login:  - - -  #
eagle #
```

Das Beispiel verbindet nun unseren Portmonitor *mytty* mit der Schnittstelle `/dev/term/b` und dem Service `/usr/bin/login`

21.5 Das Programm `ttymon`

Das Programm dient nur zur Schnittstelle für den korrekten `pmadm` Eintrag. Von daher muß es immer als *Substitute* in den Befehl eingegeben werden.

[6] `pmadm` - Siehe Kommandoreferenz Seite 357

Wichtig in diesem Zusammenhang ist der *ttylabel*. Der gibt nicht nur die Geschwindigkeit des Terminals an sondern auch noch diverse Einstellungen. Die Einstellung holt sich `ttyadm` aus der /etc/ttydefs[7]

21.6 Konfiguration über `admintool`

Ja grafisch ist es eventuell doch besser :

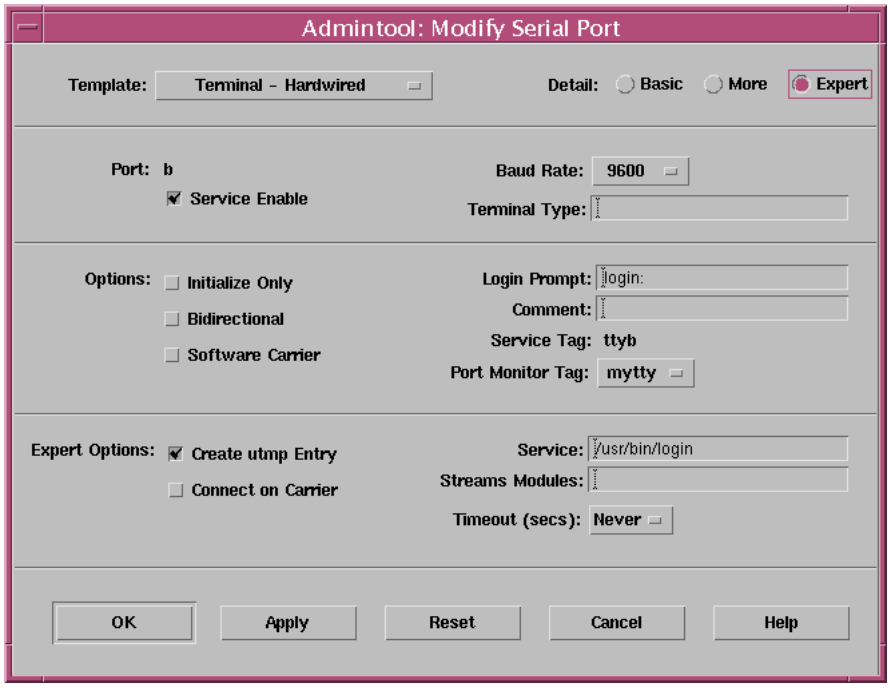


Abbildung 21.2: Konfiguration des EAGLE als Nullmodem Server

21.7 Terminalprogramm `tip`

Mit `tip`[8] kann nun auf der Gegenseite des Nullmodemkabels die Verbindung hergestellt werden. Dazu sollte man den Host in das Adressbuch /etc/remote[9] eintragen. Etwa so :

Quelltext 21.7.1 Beispieleintragung des Hosts bei Nullmodem

```
eagle-null:dv=/dev/cua/b:br#9600
```

Wichtig ist hier, daß die gleiche Baudrate wie am Server auch verwendet wird, ansonsten kommt es zu unlustigen Nebenwirkung. Weiterhin muß bei `tip` das Device unterhalb /dev/cua/* verwendet werden. Der Unterschied zwischen den /dev/term/* und /dev/cau/* ist der, daß bei den /dev/term/* die Steuerleitungen fehlen die `tip` benötigt um eine Verbindung aufzubauen.

[7] /etc/ttydefs - Siehe Konfigurationsdateien Referenz Seite 435
[8] `tip` - Siehe Kommandoreferenz Seite 393
[9] /etc/remote - Siehe Konfigurationsdateien Referenz Seite **??**

Danach kann man nun `tip`pen :

```
raven:whurst $ tip eagle-null
connected
 .. RETURN DRUECKEN ..
login: whurst
Password:
Last login: Sun Jun 11 21:46:30 on term/b
eagle:whurst $ uname -a
SunOS eagle 5.8 Generic i86pc i386 i86pc
eagle:whurst $ exit
logout

login: ^D
login: ^D
login: ^D
Connection Closed
[EOT]
raven:whurst $
```

Bildschirmausschnitt 21.7.1: Verbindungsaufbau durch Nullmodemkabel

21.8 Serielle Verbindungen über das Modem

Sorry, aber : Ich kann zwar mittels `tip` wunderbar rauswählen. Aber die Intelmühle will nicht abheben. Und wenn man dem Modem er selbst erlaubt legt es gleich wieder auf. Ich probiere noch rum und notiere es dann bei Gelegenheit.

21.8.1 Konfiguration des Dial-In Servers

21.8.2 Konfiguration von `tip`

Quelltext 21.8.1 Beispieleintragung des Hosts bei Modemwahl

```
eagle-modem:at=hayes:br#57600:dv=/dev/cua/b:pn=S0=0M0X3DT4:du:
```

21.9 Zusammenfassung

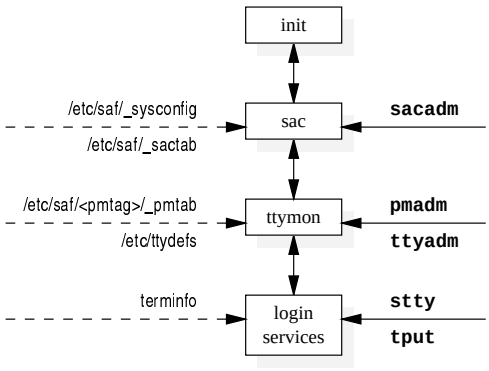


Abbildung 21.3: Der Portmonitor `ttymon` im Detail

21.9.1 Tools und Programme

Tabelle 21.1: Tools und Programme zum SAF

Programm / Datei	**Bedeutung**
`sac`[10]	Oberhaupt des SAF
`sacadm`[11]	Konfigurator des SAF/SAC
`pmadm`[12]	Verbindet ein Portmonitor mit einem Service
`ttyadm`[13]	Erstellt gültige Parameter für `pmadm`
`tip`[14]	Terminal für Serielle Leitungen

21.9.2 Konfigurationsdateien, Scripts und Devices

Tabelle 21.2: Config und Scripts zum Systemstart

Programm / Datei	**Bedeutung**
/etc/saf/_sysconfig	Konfigurationsdatei für den `sac`
/etc/saf/_sactab	Liste aller von `sac` zu startenden Portmonitore
/etc/ttydefs[15]	Terminaleinstellungen und Baudzahlen
/dev/term/*	Serielle Schnittstellen für SAF
/dev/cua/*	Serielle Schnittstellen für `tip`
/etc/remote[16]	Adressbuchdatei für `tip`

[10] `sac` - Siehe Kommandoreferenz Seite 379
[11] `sacadm` - Siehe Kommandoreferenz Seite 381
[12] `pmadm` - Siehe Kommandoreferenz Seite 357
[13] `ttyadm` - Siehe Kommandoreferenz Seite 397
[14] `tip` - Siehe Kommandoreferenz Seite 393
[15] /etc/ttydefs - Siehe Konfigurationsdateien Referenz Seite 435
[16] /etc/remote - Siehe Konfigurationsdateien Referenz Seite **??**

▲

Programm / Datei	**Bedeutung**
/etc/phones[17]	Telefonnummernverzeichnis für `tip`

[17] /etc/phones - Siehe Konfigurationsdateien Referenz Seite **??**

21.10 Praktikumsaufgaben

1. Finden Sie sich zu Gruppen zusammen. Wobei die Anzahl der Gruppen stark von der Anzahl der verfügbaren Nullmodem Kabel abhängt. Konfigurieren Sie nun einen Rechner aus Ihrer Gruppe als Terminalserver für Nullmodem Kabel. Die anderen Rechner konfigurieren Sie als Client. Konfigurieren Sie für `tip` auch ein entsprechenden Telefonbucheintrag in der `/etc/remote`.

2. Versuchen Sie die Baudrate zu erhöhen. Wer die höchste Baudrate hinbekommt hat gewonnen.

Lösungsvorschläge

1. So wie in den Unterlagen

Kapitel 22

BENUTZERMANAGEMENT

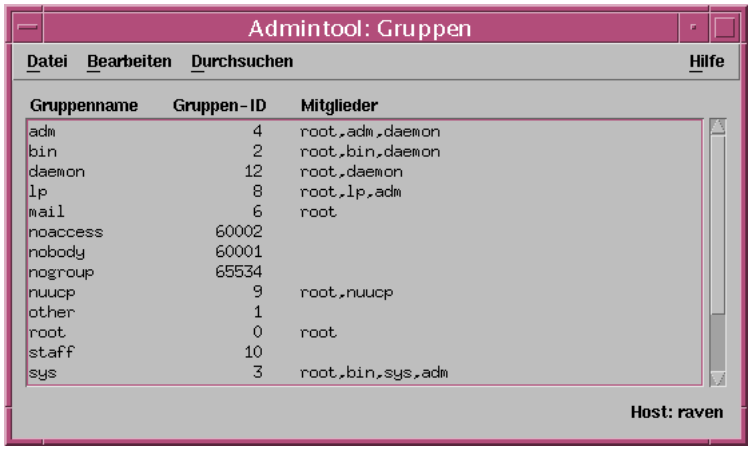


Abbildung 22.1: Group

Abbildung 22.2: Group

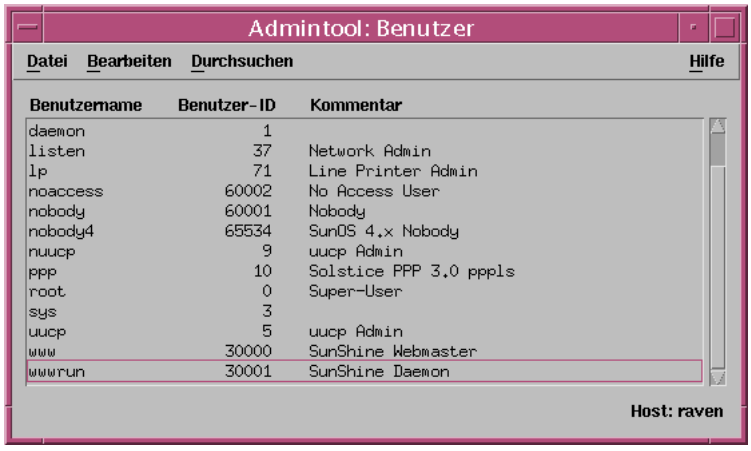


Abbildung 22.3: Group

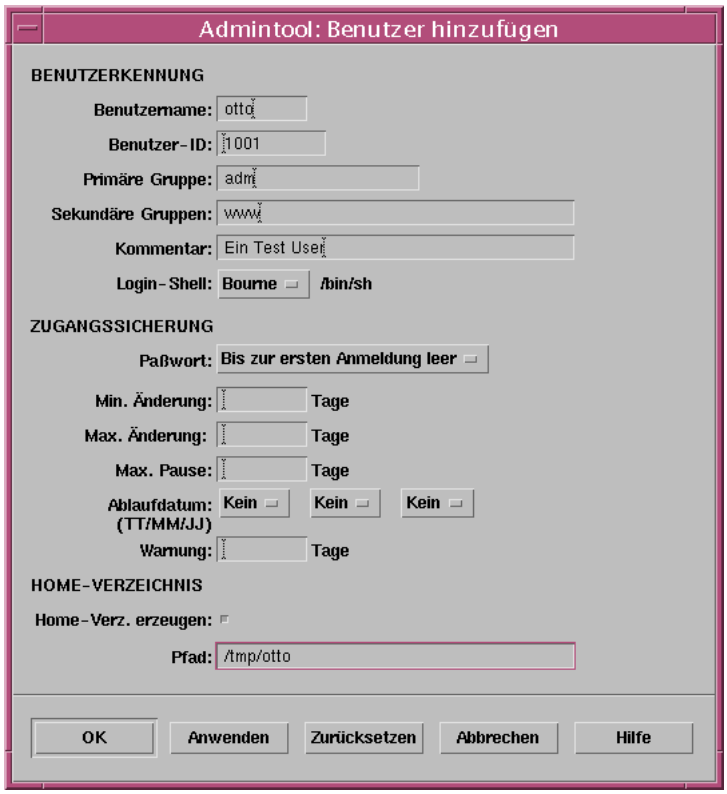


Abbildung 22.4: Group

Kapitel 23

RECHTEMANAGEMENT

23.1 UNIX Standardrechte

23.1.1 Eigentümer

23.1.2 Gruppe

23.1.3 Rest der Welt

23.2 ACL

Neu ab Solaris 7 sind die ACL[1]'s für Dateien und Verzeichnisse.

Jede Datei oder Verzeichnis besitzt von Grund her ein Eigentümer mit Rechten, eine Gruppe mit Rechten und die Rechte der anderen. Durch die ACL's können nun erweiterte Rechte an Benutzer und Gruppen vergeben werden. Da ACL's nicht Bestandteil des herkömmlichen UNIX ist kann es bei der Verwendung von heterogenen UNIX Versionen zu komplikationen kommen. Spätestens dann wenn man ein Backup mit ACL's macht und diese auf ein nicht ACL fähigem Betriebssystem (Linux) restort. In diesem Fall sind die ACL's verloren.

23.2.1 Aufbau einer ACL

Jede ACL besitzt die Rechte Einstellungen `default` die denen der normalen UNIX Rechte entsprechen. Ob man nun die default Einstellungen der ACL verändert oder mit `chmod` die Rechte ändert ist egal.

In jeder ACL können mehrere Benutzer mit unterschiedlichen Rechten abgelegt werden. z:b: Benutzer `otto` darf nur lesen und der Benutzer `karl` darf lesen und schreiben. Eigentümer der Datei ist `root` mit nur leserecht.

In jeder ACL können mehrere Gruppen mit unterschiedlichen Rechten abgelegt werden. Sprich wie bei den Benutzern auch.

In der ACL können keine weiteren 'Rest der Welt' Rechte verteilt werden. Wäre ja auch vollkommender Blödsinn, oder ?

In jeder ACL muß eine Maske eingestellt werden die sich auf eine ACL auswirkt. Diese Maske entspricht in etwa der `umask`. Mit der Maske kann man nun grundsätzlich das Schreiben verbieten obwohl die ACL definitiv sag: 'er darf schreiben'

23.2.2 Abfrage Reihenfolge

Besitzt eine Datei oder ein Verzeichnis eine ACL, dann geht das Betriebssystem wie folgt bei der Rechteprüfung vor :

[1] `ACL`-Access Control List

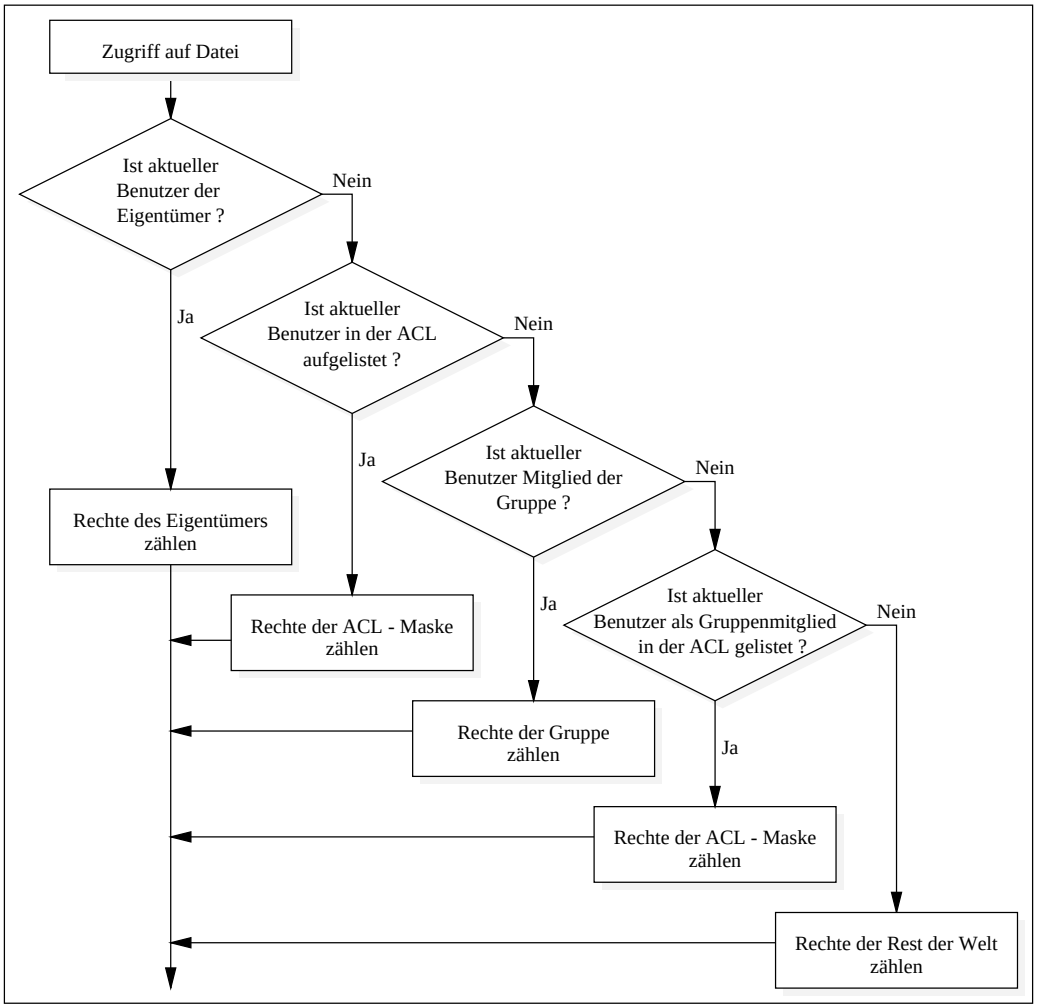


Abbildung 23.1: Rechteprüfung mit ACL

- Prüfung ob aktueller Benutzer der Eigentümer ist, wenn ja, dann zählen die Rechte des Eigentümers (bei ACL sind es dann die Default Rechte). Wenn User nicht Eigentümer ist, dann
- Prüfen ob Benutzer eine ACL besitzt. Wird eine ACL gefunden, dann werden die Rechte der ACL genommen und durch die Maske ver-ODER-t (AND). Dardurch erhält er die Effektivrechte der Datei. Besitzt der Benutzer keine ACL's dann
- Prüfen ob der Benutzer der Gruppenangehörigkeit der Datei entspricht. Ist dem so, dann zählen die Rechte der Gruppe für diesen Benutzer. Ist dem nicht so, dann
- Prüfen ob in den ACL's eine Gruppe aufgelistet ist dem der Benutzer angehört. Wurde eine solche Gruppe gefunden, dann werden die Rechte durch die Maske ver-ODER-t um seine Effektivrechte zu bestimmen. Besitzt der Benutzer keine Gruppenrechte, dann
- Rechte der 'Rest der Welt' werden genommen.

Sollte UNIX irgendeine Kombination in dieser Reihenfolge finden die passt werden die Rechte genommen und aus. Es findet keine Rechte Vermischung wie unter Novell statt. Beispiel : Dem

Benutzer `otto` gehört die Datei `text.dat` mit den Rechten `---r-----`, der Gruppe gehört er auch an. Er kann die Datei nicht lesen, weil er der Eigentümer ist. Das er auch noch der Gruppe mit Leserechten angehört ist egal. Soweit kannte man das ja schon. Setzt man nun ein ACL für den Benutzer `otto` mit Leserechten, ändert das an der Sachlage nix. Er kann diese trotz ACL nicht lesen.

23.2.3 Wichtiges für ACL

ACL's sind, wenn alle UNIX Versionen mitspielen, sehr gut geeignet um kurz mal jemanden ein Recht an eine Datei zu geben, welchem man vorher durch aufwendige Gruppendefinierungen hätte erledigen müssen. Aber Vorsicht ist jedoch angesagt.

Backup 's von Dateien mit ACL's sollten mit `tar` grundsätzlich mit der Option *p* gesichert und zurückgespielt werden. Wir ein Backup mit `cpio` gefahren sollte man den ASCII Header verwenden mit der Option für die ACL's. Meist *-H odc -P.*

Erkennung 's Zeichen einer Datei oder Verzeichnis mit einer ACL ist ein kleines Plussymbol am Ende der Rechteanzeige bei `ls -l`. z.B. `drwxr-xr-x+ ...`.

23.2.4 Setzen von ACL's

Mit dem Kommando `setfacl` (siehe 41 auf Seite 279) können ACL's gesetzt werden. Dazu verwendet man als erste Option den Modus. Dazu gehoeren das **setzen von ACL** mit *-s* wobei alte ACL's entfernt werden **löschen von ACL** mit *-d* wobei spezifizierte ACL's entfernt werden und **modifizieren von ACL** mit *-m* wobei hier neue oder alte Einträge geändert bzw. hinzugefügt werden können.

Als Rechte Beschreibung wird ein drei geteiltes Object durch Doppelpunkte verwendet. Zuerst kommt der `ACL Type`, der User, Group, etc.pp. sein kann. Gefolgt vom Usernamen oder Grouppennamen. Und danach das Recht mit voller `rwx` Angabe. Werden mehrere Berechtigungen abgesetzt werden die durch Kommas von einander getrennt.

Beispiel : Um einer Datei `text.dat` eine ACL hinzuzufügen, so daß der Benutzer `otto` zusätzliches Leserecht besitzt, wäre folgender Befehl notwendig : `setfacl -m u:otto:r-- text.dat`

23.2.5 Anzeigen und Übertragung von ACL's

Das Programm `getfacl` (siehe 41 auf Seite 279) zeigt die ACL's einer Datei oder Verzeichnis an. Die Ausgabe kann jedoch auch in eine Datei umgelenkt werden die wiederum `setfacl` verwenden kann um die ACL's zu setzen. Diese Vorgehensweise ist besonderst dann hoch interessant, wenn ein Backup auf einem anderen System eingelesen werden soll, welches jedoch keine ACL's unterstützt, oder zumindest eine andere Art und Weise der ACL Zuweisungen benutzt. Durch ein Script kann diese Datei entsprechend angepasst werden.

```
$ touch text.dat
$ setfacl -m g:bin:rw-,m:rw- text.dat
$ getfacl text.dat >text.dat.acl
$ touch text2.dat
$ setfacl -f text.dat.acl text2.dat
$ getfacl text2.dat

# file: text2.dat
# owner: whurst
# group: whurst
user::rw-
group::r--              #effective:r--
group:bin:rw-           #effective:rw-
mask:rw-
other:r--
$
```

Bildschirmausschnitt 23.2.1: Anzeigen und Übertragen von ACL's

In diesem Beispiel sieht man die Erstellung eines ACL (Maske ist wichtig !) und die Übertragung eines ACL via `setfacl -f`

23.2.6 Haupteinsatzgebiet

OK! Wir haben zwei Gruppen von Benutzern. Eine Gruppe heisst `lager` die ander ist `verwalt`. Beide Gruppen besitzen Zugang zu einem Verzeichnis. Die Gruppe `lager` mit lese und Schreibrechten, die Gruppe `verwalt` jedoch nur Leserechten. Und sonst darf kein anderer auf diese Daten Zugreifen.

Ein Teufelskreis. Er kann gelöst werden: Man erstellt einfach eine Gruppe mit dem Namen `egal` und packt dort alle Benutzer der Gruppe `lager` und `verwalt` rein. Man erstellt ein Fake-Verzeichnis und setzt dort die Gruppenangehörigkeit auf `egal` und ordnet die Rechte auf `050` ein. Unter diesem Verzeichnis nun wird das eigendliche Verzeichnis angelegt. Mach gibt es der Gruppe `lager` und setzt die Rechte auf `075`. Nun Besitzt `lager` lese und schreibrechte, der Rest der Welt bekommt leserechte, da jedoch durch das obere Verzeichnis nur alle Benutzer der Gruppe `egal` durchgekommen sind, sind der Rest der Welt nun die Benutzer aus der Gruppe `verwalt`.

Etwas Chaotisch !

Mit einer ACL *Gruppe Verwalt darf auch lesen* wäre allen geholfen.

Kapitel 24

QUOTA

24.1 Einsatz von Quotas

Der Einsatz von Quotas erlaubt die Beschränkung von Festplattenspeicherplatz eines Benutzers. Das ist in großen Unternehmen mit 1000 Benutzer sehr sinnvoll. Aber auch wenn es darum geht einem Benutzer etwas Platz auf einem Webserver zu geben, kann hier ein Limit vergeben werden.

Mit Quotas kann die Kapazität und die zu verwendeten INodes angegeben werden. Jedes Dateiensystem stellt nur ein begrenzten Platz und eine begrenzte INode Anzahl zur Verfügung. Wenn nun mehrere Benutzer sich ein Dateiensystem teilen ist es von Vorteil jedem Benutzer eine Begrenzung einzustellen. Sonst kommt es zur Situation das ein Benutzer sich alle Resourcen greift und der Rest steht auf dem Schlauch.

24.2 Planung der Quotas

Zur Planung muß es einige bekannte Größen geben. Zum einen benötigt man die Anzahl der Benutzer, die Anzahl der verfügbaren INodes und die verfügbare Kapazität des Dateiensystems. Beides kann mit dem Programm `df` (siehe 41 auf Seite 267) ermittelt werden. Wir wollen nun mal mit folgenden Voraussetzungen arbeiten :

Benutzer : 2000 Kapazität : 10 GB /export/home INodes : 600000

Nun kann man einfach die Zahlen Teilen, doch ergeben sich dann echte Grenzwerte die man nicht mehr verändern kann, sollten einige User hinzukommen. Von daher sollte man eventuell auch mit beachten wieviele Benutzer noch in Laufe der Zeit hinzukommen. Das kommt jenach Firma darauf an. Das kann man nicht in eine Formel packen.

Vielleicht gibt es auch wieder einige Benutzer die mehr benötigen.

24.3 Berechnung der Quotas

Wir gehen jetzt zur Vereinfachung hin und Teilen die Werte und Runden diese ab. Einmal für die Kapazität :

$$KapazitätProBenutzer = \frac{GesammtKapazität}{AnzahlderBenutzer} = \frac{10737418240Bytes}{2000Benutzer} = 5368709Bytes = 5MB$$

Und für die INodes :

$$InodesProBenutzer = \frac{GesammtInodes}{AnzahlderBenutzer} = \frac{600000Inodes}{2000Benutzer} = 300$$

Mit dieser Rechnung könnte jeder Benutzer 300 Dateien erstellen darf aber nicht über 5 MB hinaus.

24.4 Soft und Hard Limit

Bei Quotas gibt es zwei Grenzen. Das Softlimit und das Hardlimit. Das Softlimit kann ein Benutzer kurzzeitig ausnutzen. Ein Hardlimit ist die echte Grenze. Darüberhinaus ist nicht möglich. Zurück zum Softlimit. Wenn ein Benutzer das Softlimit erreicht bekommt er eine entsprechende Meldung. Wenn ein Benutzer das Softlimit übergeht wird intern ein Zeitzähler aktiviert. Nach ablauf des Zeitzählers wird der aktuelle Verbrauch an Resourcen zum Hardlimit, und zwar solange bis der Benutzer wieder unter dem Softlimit liegt. Den Zeitzähler kann man Konfigurieren und liegt Standardmässig auf 7 Tagen.

Mit diesem Wissen könnte man nun z.B. hingehen und das Softlimit auf 250 Dateien und 4,5 MB setzen und das Hardlimit auf 350 Dateien und 5,5 MB. Das Dumme ist nur, verhalten sich alle Raudiehaft geht es nach hinten los. Geht man hin und setzt das Hardlimit auf 300 Dateien mit 5 MB und das Softlimit auf 250 Dateien und 4,5 MB dann hat man gute Karten. Und das tun wir auch so.

24.5 Vorbereitung zur Benutzung von Quotas

Quotas können nur auf gesammte Dateiensysteme angewendet werden. Ein Verzeichnis mitten in einem Dateiensystem mit Quotas zubelegen ist nicht möglich. Folglich muß es ein Mountpoint sein. Damit Quotas vom System auch erkannt werden muß das Dateiensystem mit einer Speziellen Option gemounten werden. Dazu muß man in der /etc/vfstab (siehe **??** auf Seite ??) bei den Mountoptionen die Option `rq` eingetragen werden :

Quelltext 24.5.1 Mountpointeintrag mit Quota in der /etc/vfstab

```
/dev/dsk/c0t0d0s7 /dev/rdsk/c0t0d0s7 /export/home ufs 2 yes rq
```

Als nächstes muß man das Quota Kontrollfile `quotas` mit Hand erstellen. Fehlt dieses wird Quota nicht installiert. Dazu muß man einfach nur eine leere Datei im Root-Verzeichnis des Dateien Systems erstellen und die Rechte auf `600` für den Benutzer `root` setzen.

```
# cd /export/home
# touch quotas
# chmod 600 quotas
#
```

Bildschirmausschnitt 24.5.1: Erstellen der Quota Kontrolldatei

Danach sollte man das System neu booten damit alles 100%-ig funktioniert. Beim Booten wird einem dann `UFS Quotacheck` angezeigt.

24.6 Editieren von Quotas

Zur Editierung von Quotas muß das Programm `edquota` (siehe 41 auf Seite 269) verwendet werden. Es startet den eingestellten Editor und gibt Ihnen die aktuellen Einstellungen in Form einer Textzeile. Man sollte das Format nicht verändern. Gab es für diesen Benutzer noch keine Quotas dann ist vorerst alles auf NULL gesetzt, und somit deaktiviert. Um dieses Zuändern muß man nun diese Textzeile entsprechend ändern :

```
# edquota otto
im Editor ...
fs /export/home blocks (soft = 4500, hard = 5000) inodes (soft = 250, hard =
#
```

Bildschirmausschnitt 24.6.1: Quotas für ein Benutzer

Nach dem Abspeichern und verlassen des Editors sind die Quotas aktiv. Hat man jedoch 5 Millionen Benutzer dann hat man jetzt ein recht aufwendigen Job. Deswegen kann man Quotaeinstellungen von einem Benutzer zu einem anderen Benutzer kopieren

24.7 Kopieren von Quotas

Wenn man ein Benutzer mit Quotas belegt hat und möchte die gleichen Einstellungen für einen anderen Benutzer setzen kann man das Programm `edquota -p` verwenden. Das Programm stellt keine weiteren Anfragen mehr und kann in Benutzererstellungsscripten aufgerufen werden um Aufgaben zu automatisieren. Im folgenden Beispiel werden die Einstellungen des Benutzers `otto` für `karl` und `peter` kopiert :

```
# edquota -p otto karl peter
#
```

Bildschirmausschnitt 24.7.1: Kopieren von Quotas

Das Beste ist wenn man ein Benutzer als Schablone erstellt. Der Benutzer muß sich nicht einloggen können und dient nur als Kopieruser. Danach erstellt man ein Script der allen betroffenen Benutzern das Quota zuweisen. Hier kann man z.B. sich alle Benutzer aus einer Gruppe raus holen und den Benutzern das Quota zuweisen.

24.8 Zeitzähler Einstellungen

Den Zeitzähler kann man nicht einzelnd einstellen. Auch das Nachträgliche editieren ist nicht möglich. Das einzige was man machen kann ist die Standardeinstellung zu ändern, die jedoch wirkt sich nicht auf bereits bestehende Einstellungen aus. Man sollte also vorher den Zeitzähler konfigurieren.

Um dieses zu tun muß man `edquota -t` verwenden. Auch hier macht sich wieder ein Editor auf, allerdings hat die Textzeile ein anderes Format :

```
# edquota -t
im Editor ...
fs /export/home blocks time limit = 2 weeks, files time limit = 16 days
#
```

Bildschirmausschnitt 24.8.1: Einstellen des Standard Zeitzählers

Nach dem Speichern ist das Zeitlimit aktiv.

24.9 Überwachung von Quotas

Jeder Benutzer kann seine eigenen Quotas mit dem Befehl `quota` anzeigen lassen. Das folgende Beispiel zeigt den Aufruf von Quota des Benutzers `otto` der über das Softlimit hinweg aggiert :

```
$ quota
Over disk quota on /export/home, remove 10K within 6.9 days
Over file quota on /export/home, remove 10 file within 6.9 days
$
```

Bildschirmausschnitt 24.9.1: Ausgabe von `quota` bei Softlimit ausnutzern

Bei Benutzern die das Softlimit nicht erreicht haben kommt keine weitere Ausgabe. Um das Quota genauer anzeigen zulassen muß man mit `-v` arbeiten :

```
$ quota -v
Disk quotas for karl (uid 55403):
Filesystem     usage  quota  limit    timeleft  files  quota  limit    timeleft
/export/home      20   4500   5000                  3    250    300
$
```

Bildschirmausschnitt 24.9.2: Ausgabe von `quota -v` bei Benutzern

Beim `otto`, dem Raudy, siehts so aus :

```
$ quota -v
Disk quotas for otto (uid 55401):
Filesystem     usage  quota  limit    timeleft  files  quota  limit    timeleft
/export/home    4510   4500   5000    6.9 days    260    250    300    6.9 days
$
```

Bildschirmausschnitt 24.9.3: Ausgabe von `quota -v` bei `otto`

Nur der Administrator ist in der Lage einen kompletten Report oder Quotas von Benutzern anzeigen zulassen. Der Quotareport kann mit `repquota` erstellt werden und es werden nur die Benutzer angezeigt die über das Softlimit liegen

```
# repquota -a
/dev/dsk/c0t0d0s7 (/export/home):
                     Block limits                        File limits
User            used    soft    hard    timeleft    used    soft    hard    timeleft
otto      ++    4510    4500    5000    6.9 days     260     250     300    6.9 days
#
```

Bildschirmausschnitt 24.9.4: Ausgabe von `repquota -a`

24.10 Andere Administrative Kommandos

Die jedoch beim Systemstart entsprechend aufgerufen werden. Man kann aber auch das ganze Quoting im laufenden System machen und installieren.

quotacheck (siehe 41 auf Seite 367) prüft ein angegebenes Device auf die Quotas und erstellt die Kontrolldatei für Quota neu

quotaon (siehe 41 auf Seite 371) schaltet das Quota für ein Device ein

quotaoff (siehe 41 auf Seite 369) schaltet das Quota für ein Device aus

24.11 Zusammenfassung

24.11.1 Befehle für Quotas

Tabelle 24.1: Befehle für Quotas

Programm / Datei	**Bedeutung**
`quotaon`	Schaltet Quotas ein
`quotaoff`	Schaltet Quotas aus
`quotacheck`	Überprüft Dateiensystem und erstellt Quotas neu
`edquota`	Editierung von Quotas
`quota`	Anzeige der Quotas
`repquota`	Erstellt Reports von Softlimit Benutzern
`quotas`	Kontrolldatei im Root des Dateiensystems

24.11.2 Vorgehensweise

1. Dateiensystem Formatieren und Berechnung der Resourcen pro Benutzer
2. Quotas für Dateiensystem erstellen und neu Booten
3. Erstellen der Default-Timeleft
4. Erstellen eines Templatebenutzers mit den Quotawerten
5. Erstellen der Benutzer (sofern nicht schon vorhanden)
6. Kopieren der Templatequotas an die Benutzer

Kapitel 25

DRUCKERKONFIGURATION

Die Druckerkonfiguration unter Solaris kann auf zwei verschiedenen Wegen passieren. Zum einen gibt es die Kommandozeilen Version, die sehr kopliziert zu sein scheint, und zum Zweiten die grafische Installation des Druckers, die einfacher zu sein scheint.

Allgemein lässt sich jedoch eins definitiv sagen: Linux verwendet das BSD Drucksystem und Solaris das SystemV System. Beide sind Toll, System V jedoch wesendlich komplexer gestattet aber mit Zugriff auf Benutzerebene, BSD nur auf Hostebene.

Diese Unterlagen konzentrieren sich vorerst auf das SystemV-Solaris Druckkonzept.

25.1 Allgemeine Funktionsweise

Im System gibt es ein Systemdienst der alle Druckaufträge verwaltet und den Drucker entsprechend managt. Dieser Systemdienst wird meist beim Booten des Systems gestartet.

Der Benutzer benutzt ein Usertool um sein Druckauftrag abzusetzen. Dieses Tool informiert den Systemdienst über ein neuen Druckauftrag. Danach wird das Usertool verlassen, und der Systemdienst übernimmt alle weiteren Aufgaben. Durch diese Trennung wird die stabilität des Systems garaniert.

Der Systemdienst jagt die in Auftrag gegebenen Daten durch den Druckerfilter (auch Treiber genannt) und sendet die konvertierten Daten zum Drucker.

Man kann sich nun folgende Grafik und den Text dazu durch lesen um die Funktionsweise genau zu verstehen.

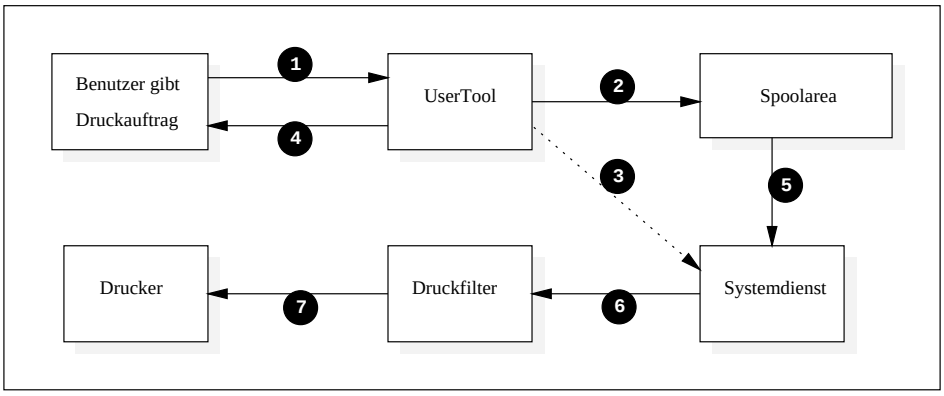


Abbildung 25.1: Funktionsweise des SystemV Drucksystems (Einfach)

1. Eine Anwendung des Benutzers gibt ein Druckauftrag mittels Usertool in Auftrag.
2. Das Usertool speichert die Rohdaten des Auftrages in der Spoolarea (Druckerquere)
3. Das Usertool informit den Systemdienst darüber, das ein neuer Druckauftrag in der Spoolarea vorliegt
4. Das Usertool wird verlassen und die Anwendung arbeitet weiter

5. Der Systemdienst wartet bis der Drucker frei ist und holt sich die Rohdaten des Druckerauftrages
6. Er schiebt diese durch den Druckfilter
7. Und das konvertierte Format geht zum Drucker

Anwendung Als Anwendung kann jedes beliebiges Programm zum Einsatz kommen. Z.B. der Netscape, die Shell oder StarOffice

Usertool Das Usertool zum Drucken ist das Programm `lp` (siehe 41 auf Seite 307). Alternativ kann das kompatible `lpr` verwendet werden.

Spoolarea Die Spoolarea liegt unter `/var/spool/lp/tmp/...`. Bei einem Druckserver für Abteilungen sollte man darauf achten das dort genügend Platz ist.

Systemdienst Der Systemdienst wird durch `/usr/lib/lpsched` (siehe 41 auf Seite 315) repäsentiert und wird meist durch das Startmodul `etc/init.d/lp` gestartet bzw. gestoppt.

Druckfilter Der Druckfilter ist meist ein Shellscript. Das Script bekommt die Rohdaten als stdin geliefert und muß die Druckerdaten zum stdout senden. Unter SystemV Systemen nennt sich der Druckfilter auch Interface.

Drucker Der Drucker selbst kann fast jeder beliebiger Drucker sein. Dieser kann lokal oder über das Netz angeschlossen sein.

25.1.1 Druckerarten

Das Druckersystem von UNIX kennt verschiedene Arten eines Druckers.

Physikalische Drucker Der physikalische Drucker ist der Drucker selbst. Der Drucker kann entweder ein HP, Epson oder sonstiges sein.

Logische Drucker Ein logischer Drucker ist ein nicht wirklich vorhandener Drucker. Ein logischer Drucker bekommt einen Namen. Im System wird dieser Name mit einem Druckerfilter und einem Drucker verbunden. Der gleiche Drucker kann jedoch mittels einem anderen Filter unter einen anderen Namen logisch erscheinen. Beispiel : Man hat ein Drucker auf den Formulare und Quelltexte gedruckt werden sollen. Für die Quelltexte schreibt man ein Filter und verbindet die beiden zum logischen Drucker mit dem Namen 'sourceprinter'.

Local Drucker Als ein lokaler Drucker wird ein Drucker bezeichnet der direkt an einer Workstation angeschlossen ist. Meist wird dies über eine Serielle oder Parallele Schnittstelle gemacht

Remote Drucker Als ein remote Drucker, wird ein Drucker bezeichnet der an einer Workstation direkt angeschlossen ist, auf dem jedoch ein Druckserver läuft über den man drucken kann. Der Druckauftrag muß also zuerst über das Netzwerk

Netzwerk Drucker Als ein Netzwerkdrucker wird ein Drucker bezeichnet der eine Netzwerkschnittstelle eingebaut hat, und der direkt im Netzwerk hängt. Auf diesem Drucker läuft ein Druckserver. Zur Konfiguration kann `telnet` verwendet werden.

25.1.2 Drucker Befehlsformate

Jeder Druckerhersteller gibt seinem Drucker eine eigene Sprache mit. Nur mit dem richtigen Treiber funktioniert der Ausdruck später auch. Die Firma Epson spielt mit ESC/P2 Sequenzen rum, HP setzt auf PCL, und so weiter. Wohin das führt sieht man unter Windows.

Unter UNIX hat sich eine Sprache durchgesetzt. Postscript. Postscript hat jedoch für den Otto-Normal Verbraucher einige Nachteile. Um die Seitenbeschreibungssprache Postscript druckfertig zumachen, benötigt der Drucker ein Leistungsstarken Prozessor (i960 oder Motorola). Weiterhin benötigt er Speicher. Damit wird zwar die CPU beim Druck entlastet jedoch kosten die Drucker meist etwas mehr.

Fast alle Programme unter UNIX liefern Postscript als Ausgabe zum Drucken. Wenn man nun kein Postscriptdrucker besitzt muß man sich ein Filter schreiben der das Postscript in die Druckersprache umwandelt. Dazu eignet sich GhostScript ganz hervoragend.

25.1.3 Kontrolle behalten

Das gesamte Drucksystem besitzt zwei Schalter die in den Datenfluß eingreifen. Der Erste nennt sich *accept/reject* mit den man die Zufuhr vom Anwender in die Spoolarea unterbinden kann. Der zweite nennt sich *enable/disable* mit dem die Zufuhr vom Interfaceprogramm zum Drucker unterbinden kann. Die folgende Grafik erläutert dieses :

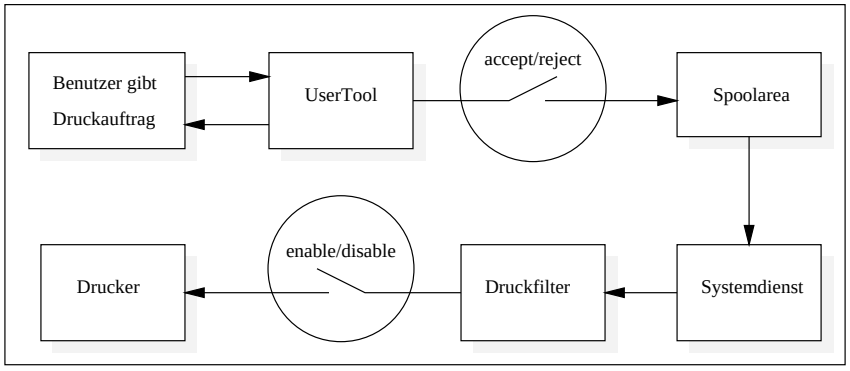


Abbildung 25.2: Kontrolle behalten

Beide Schalter haben Ihre Berechtigung :

accept/reject wird meist genutzt wenn ein Drucker ausgesondert worden ist, oder wenn dieser Beschädigt ist und längere Zeit nicht mehr verfügbar sein wird. Somit wird einfach ein Überlaufen der Quere verhindert

enable/disable wird meist benutzt wenn der Drucker kurze Zeit ausfällt oder wenn man Papier nachfüllen muß. Die Druckaufträge bleiben alle erhalten und werden nach der Freischaltung ge-

druckt. Für längere Ausfallzeiten ist diese Art nicht zu Empfehlen, weil sonst der Druckserver einfach überläuft.

25.1.4 Druckerklassen

Wenn es mehrere Drucker der gleichen Bauart gibt, dann ist es für den Benutzer eigentlich egal auf welchen Drucker er sein Auftrag sendet. Das SystemV Drucksystem bietet die Möglichkeit mehrere Drucker zu einer logischen Klasse zusammenzuführen. Der Benutzer kann dann anstatt des Druckernamens den Klassennamen angeben. Der Auftrag wird dann auf den nächsten frei verfügbaren Drucker dieser Klasse gedruckt. Sehr ungeeignet ist es jedoch wenn die Drucker verteilt über das gesammte Firmengelände stehen. Die Benutzer werden solange den Auftrag versenden bis er irgendwann einmal im eigenen Büro gedruckt wird, weil die anderen Drucker noch mit seinen vorherigen Auftröagen beschäftigt sind. Ungünstig.

25.2 Konfiguration eines lokal/logischen Druckers

Für die Konfiguration soll hier mal ein HP DeskJet 1200C an einer Ultra10 als Beispiel dienen. Da dieser Drucker kein Postscript kann muß ein Filter (Interface) her. Zum Glueck gibt es unter Solaris einen HP-Laser Treiber der die Druckersprache PCL beherrscht. Mir geht zwar dann die Farbe aus, aber das kann man immer noch ändern wenn es soweit ist.

25.2.1 Erste Vorbereitungen

Da der Drucker an der lokalen parallelen Schnittstelle angeschlossen werden soll, muß man das Devicefile entsprechend mit Rechten versehen. Nach einer Installation hat das Device die Rechte `666` welches sich für ein Druckserver als unglaublich dumm darstellt, weil jeder irgendwelche Daten durch eine einfache Umlenkung an den Port senden kann. Passiert das wärend des Druckens ist es vorbei.

Wir müssen also die Berechtigung so setzten das nur noch der Druckserver die Rechte besitzt :

```
# chown lp /dev/ecpp0
# chmod 600 /dev/ecpp0
#
```

Bildschirmausschnitt 25.2.1: Einstellen der Deviceberechtigungen

Zu beachten ist hier das Device sich anderst schimpt, wenn es eine serielle Schnitstelle wäre.

Weiterhin muß geprüft werden ob der Druckerdienst läuft. Das kann man feststellen indem man in der Prozessliste nach `lpsched` sucht. Läuft dieser nich muß er mittels Startmodul `etc/init.d/lp` gestartet werden.

```
# ps -ef|grep lp
    root  1891  1842  0 20:37:34 pts/6    0:00 grep lp
  whurst  1885  1883  0 20:29:42 ?        0:00 (dns helper)
# /etc/init.d/lp start
Print services started.
#
```

Bildschirmausschnitt 25.2.2: Starten des Druckdienst, sofern dieser nicht schon läuft

25.2.2 Einrichten eines Druckers

Als erstes muß man sich ein logischen Druckernamen einfallen lassen. Man kann die Drucker benennen wie man will, man sollte sich diese jedoch irgendwie merken können. Ich werde mal hingehen und den Drucker jetzt einfach mal `hpmonotext` nennen.

Zur Einrichtung muß das Programm `lpadmin` benutzt werden. Die folgenden Befehle werden unten der Reihe nach erklärt :

```
# lpadmin -p hpmonotext -v /dev/ecpp0
# lpadmin -p hpmonotext -T hplaser
# lpadmin -p hpmonotext -I simple
# lpadmin -p hpmonotext -D "HP1200C s/w"
```

Bildschirmausschnitt 25.2.3: Einrichten des Druckers via lpadmin

1. Zuerst muß der logische Drucker mit einer Schnittstelle konfiguriert werden. Als Schnittstelle kommt hier der Centronicsport zum Einsatz
2. Als nächstes muß man dem Solaris Standard Interface den Typ des Druckers mitteilen. Es ist ein `hplaser`
3. Weiterhim muß man Ihm sagen welche Dateitypen dieser Drucker versteht. Simple steht für einfache Texte.
4. Eine Beschreibung des Drucker füge ich auch mal hinzu

25.2.3 Freischalten des Druckers

Nun muß der Drucker nur noch freigeschaltet werden. Das geht mit den Schaltern. Die haben glücklicherweise den gleichen Namen wie die Programme :

```
# accept hpmonotext
destination "hpmonotext" now accepting requests
# /usr/bin/enable hpmonotext
printer "hpmonotext" now enabled
# lpstat -p hpmonotext
printer hpmonotext is idle. enabled since ... available.
#
```

Bildschirmausschnitt 25.2.4: Freischalten des Druckers

Mit `lpstat` (siehe 41 auf Seite 317) wird der Drucker noch einmal geprüft. Warum steht dort `/usr/bin/enable` ? Nun `enable` ist von der bash ein eingebautes Kommando ! Deshalb.

25.2.4 Erster Druckversuch

Man gebe ein `lp -d hpmonotext /etc/hosts` und erfreue sich am Ausdruck. Das *-d* steht für den logischen Druckernamen.

25.3 Konfiguration eines nicht Postscriptdruckers

Um ein nicht Postscriptfähigen Drucker in das System einzubinden, stellt sich als schwierig dar. Unter dem Solaris gibt es keine Treiber. Glücklicherweise gibt es das Paket *GhostScript* welches auch unter Linux eingesetzt wird, auch für Solaris. Es befindet sich im Paket SFWgs (oder so) und installiert sich nach `/opt/sfw`. In diesem Paket liegt ein eigenes Interface bei.

Der Filter kann noch etwas mehr. Er kann auch PDF's verarbeiten. Eine Liste der Ausgabeformate bekommt man mittels `/opt/sfw/bin/gs --help` da sieht man das er nicht nur viele Drucker unterstützt sondern auch Faxformate und Grafikformate.

Wir konfigurieren also einen neuen Drucker. Die Befehle werden unten wieder erklärt :

```
# lpadmin -p hpmonops -v /dev/ecpp0
# lpadmin -p hpmonops -I postscript,pdf
# lpadmin -p hpmonops -T unknown
# lpadmin -p hpmonops -D "HP1200C Postscript s/w"
# lpadmin -p hpmonops -i /opt/sfw/share/ghostscript/interfaces/GSinterface
# lpadmin -p hpmonops -o GS_DEVICE=laserjet
# lpadmin -p hpmonops -o PAPERSIZE=a4
# accept hpmonops
destination "hpmonops" now accepting requests
# enable hpmonops
printer "hpmonops" now enabled
#
```

Bildschirmausschnitt 25.3.1: Konfiguration eines nicht Postscriptdruckers

1. Der Drucker wird mit der Schnittstelle verbunden
2. Als zu akzeptierende Datentypen wird Postscript und PDF eingestellt
3. Der Typ des Druckers ist *unknown*. Das ist jedoch nur für Solaris wichtig, weil Solaris diesen Druckertyp ja nicht kennt.
4. Eine ultimative Beschreibung
5. Angabe des Interfaceprogramms welches verwendet werden soll. Es wird automatisch an den richtigen Platz kopiert
6. Dem GhostScript über die Druckeroptionen mitteilen welches Ausgabeformat wir gern hätten. Denkbar sind hier alle die in der Liste von `gs --help` stehen. Sollte jedoch irgendwie zum Drucker passen

7. Für den `gs` die Papiergroesse mitteilen
8. Freischalten des Druckers

Nun kann man auch Postscript und PDF Dateien drucken. Mit einem `lp -d hpmonops /opt/sfw/share/ghostscript/5.10/examples/tiger.ps` kann das Glueck versucht werden

Praktikum

1. Konfigurieren Sie zuerst die Gerätedatei so, daß nur der Benutzer `lp` lese und schreibrechte hat. Unter Solaris(intel) ist es `/dev/lp1` ist diese Datei nicht vorhanden, dann muß im Bios die Paralelle Schnittstelle umkonfiguriert werden auf Bi-Direktional ohne DMA. Ein anschließender Boot mit der Option *-r* behebt das Problem.

2. Konfigurieren Sie mittels `lpadmin` ein Drucker mit dem Namen *hpd5text* und weisen dem die Schnittstelle zu

3. Drucken Sie nun ein ein Text. Sollte es zum Treppeneffekt kommen Schalten Sie am Drucker von (CR=CR+LF) nach (CR=CR) um. DIP Schalter !

Lösungsansätze

1. ```
   chown lp /dev/lp1
   chmod 600 /dev/lp1
   ```

2. ```
   lpadmin -p hpd5text -v /dev/lp1
   ```

3. DIP Schalter sind am HP DeskJet 500 unter der Papierzuführung. Im Deckel selbst ist eine Beschreibung der einzelnen Schalter.

Kapitel 26

BACKUP UND ARCHIVIERUNG

26.1 Backupstrategien

26.2 Backupmedien

26.3 Archivierungsprogramme

26.3.1 cp

26.3.2 tar

26.3.3 cpio

26.3.4 pax

26.3.5 ufsdump

26.3.6 dd

26.4 Kompression

26.4.1 pack/unpack

26.4.2 compress/uncompress

26.4.3 GNU Zip

26.4.4 BZip2

ZEITGESTEUERT AKTIONEN AUSFÜHREN

Um ein System zu automatisieren benötigt man ein Mechanismus der es einem erlaubt zu einer bestimten Zeit eine Operation auszuführen. Genau das macht *cron* und *at*.

Mit dem Cron können die Benutzer und Administratoren zu bestimmten Zeiten ein Programm starten. Meist ist es ein Shellscript welches die automatischen Vorgänge entsprechend überwacht und im Fehlerfall auch korrekt reagiert.

Mit dem At Mechanismus kann ein Benutzer eine Aktion für die Zukunft starten. Der Start erfolgt nur einmal und muß jedesmal wieder ge-at-tet werden.

27.1 Der Cron Mechanismus

Jedes UNIX System kann ein Cronmechanismus implementieren. Um einem Benutzer jedoch zu ermöglichen, daß er z.B. jeden Tag eine Operation ausführen kann benötigt man ein Prozeß der alle Vorgänge entsprechend überwacht.

Unter Solaris wird der Systemprozeß `cron` durch das Script `/etc/init.d/cron` entsprechend gemanaged. Sofern der Link `/etc/rc2.d/S75cron` existiert wird der Daemon automatisch beim Hochfahren des Systems gestartet. Nachdem der Daemon läuft analysiert er die gesammten Cron-Tabellen (crontab) und handelt entsprechend.

27.1.1 Einstellen von Aktionen

Ein Benutzer kann seine eigenen Aktionen definieren. Um dieses zu tun benötigt er einen ASCII Editor wie den *vi* und muß diesen in der Environmentvariable `EDITOR` exportieren. Etwa so :

```
$ EDITOR=/usr/bin/vi
$ export EDITOR
$
```

Bildschirmausschnitt 27.1.1: Einstellen des Editors

Normalerweise tut der Benutzer dieses in seiner $HOME/.profile um nicht ständig den Editor einzustellen. Fehlt diese Einstellung wird der Editor *ex* verwendet. Mit dem jedoch kenn ich mich auch nicht so aus und weiss nur daß man den mit 'q' wieder verlassen kann.

Nachdem der Editor eingestellt wurde, muß die Cron-Steuerdatei, die sogenannte crontab, editiert werden. Dazu ist das Commando `crontab`[1] von nöten. Das Programm öffnet danach die crontab des Benutzers.

[1] `crontab` - Siehe Kommandoreferenz Seite 265

27.1.2 Format der crontab

Eine crontab[2] ist eine gewöhnliche ASCII Textdatei und erlaubt **KEINE** Leerzeilen. Jedoch können Kommentare mit Hashs eingeleitet werden. Jede einzelne Zeile entspricht einer Aktion. Jede Zeile wiederum besteht aus 6 Spalten. In der ersten Spalte werden die Minuten, in der zweiten die Stunden, in der dritten die Tage, in der vierten die Monate und in der 5 Spalte wird der Wochentag definiert. Ab der 6. Spalte wird das Programm angegeben welches man ausführen will.

Jedes Zeitfeld kann eine Liste von Zeiten enthalten. Wird z.B. `0,15,30,45` in das Minutenfeld eingetragen wird die Aktion alle 15 Minuten ausgeführt, sofern der Rest auch passt. Als Wildcard kann der Stern verwendet werden.

Folgende Beispieleinträge der Datei erläutern wird unten :

Quelltext 27.1.1 Beispieleinträge einer crontab

```
10 3 2 1 *   /usr/bin/p1
15 * * * *   /usr/bin/p2 >/var/adm/log/cronjobs/p2log 2>&1
1  0 * * 1,4 /usr/bin/p3 | mail whurst
```

1. Jeden 2. Januar um 3.10 Uhr wird das Program p1 gestartet
2. Wenn der große Zeiger der Uhr die 15 erreicht wird das Programm p2 gestartet. Egal an welchem Tag und egal zu welcher Stunde. Die Ausgabe wird umgelenkt
3. Jeden Montag und Donnerstag um 0.01 Uhr wird Programm p3 gestartet. Die Ausgabe wird zuge-mail-t

27.1.3 Die Ausgabe der Prozesse

Wenn ein Cronprozeß eine Ausgabe macht und diese wird nicht in eine Datei oder sonstwohin umgelenkt, dann landet diese Ausgabe nicht auf dem Bildschirm sondert wird dem Benutzer dem die crontab gehört zuge-mail-t. Von daher muß'beim Einsatz von `cron` auch `sendmail` funktionieren, sonst landt die Ausgabe einfach nirgenz.

27.1.4 Berechtigungen

Der Administrator sollte dafür sorgetragen, daß nicht jeder Benutzer Cron-Jobs loslassen kann. Zum einen behindert das eventuell die Arbeit an wirklich wichtigen Prozessen (Backup). Es könnten jedoch aber auch komische Aktionen jede Nacht ausgeführt werden die jeden Morgen für einen Systemabsturz sorgen (hihi). Ein Benutzer hat in einem Cron-Job genausoviele Rechte wie sonst auch.

Im Normalfall sollten nur die Administratoren die Berechtigung bekommen Cronjobs zu implementieren. Man kann dem `cron` mittels zweier Dateien mitteilen wer Cronjobs haben darf und wer nicht. Existiert keine der beiden Dateien, dann darf keiner.

Hier ist jedoch Vorsicht angesagt. Werden beide Dateien benutzt, kann es zu sehr merkwürdigen Zuständen kommen. Die eine Datei führt die Benutzer die dürfen (/etc/cron.d/cron.allow), die andere die nicht dürfen (/etc/cron.d/cron.deny). Um, sofern beide bestehen, ein Chaos zu verhindern folgt `cron` folgender Strategie :

[2] crontab - Siehe Konfigurationsdateien Referenz Seite 415

- Ein Benutzer hat die Berechtigung, wenn
  - sein Benutzername in der /etc/cron.d/cron.allow steht
  - die /etc/cron.d/cron.allow nicht existiert und sein Benutzername nicht in der /etc/cron.d/cron.deny steht
- Ein Benutzer hat KEINE Berechtigung, wenn
  - die /etc/cron.d/cron.allow existiert und sein Loginname nicht drinn steht
  - die /etc/cron.d/cron.allow nicht existiert und der Benutzername in der /etc/cron.d/cron.deny drinn steht
  - beide Dateien nicht existieren

Man sollte sich für ein Primärmechanismus entscheiden.

Entweder allen die Berechtigung verwehren und nur einigen die Berechtigung erteilen. Berechtigte Benutzer in die /etc/cron.d/cron.allow notieren und die /etc/cron.d/cron.deny löschen.

Oder den anderen Weg: Allen die Berechtigung erteilen nur eigen nicht. Dazu die /etc/cron.d/cron.allow löschen und die die nicht dürfen in die /etc/cron.d/cron.deny notieren.

Das Solaris 8 hat standardmässig keine /etc/cron.d/cron.allow definiert jedoch einige Systembenutzer in das /etc/cron.d/cron.deny

27.2 Der At Mechanismus

27.3 Zusammenfassung

Tabelle 27.1: cron und at

Programm / Datei	**Bedeutung**
/usr/sbin/cron	Daemon der die cron-Aktivitäten überwacht
/etc/init.d/cron	Startmodul für cron
/etc/rc2.d/S75cron	Startmodul für cron (Runlevel 2)
EDITOR=	Variable zur Einstellung des Editors
crontab	Benutzerprogramm zur Wartung von Cron-Aktionen
/var/spool/cron/crontabs	Ablage aller crontab Dateien
/etc/cron.d/cron.allow	Berechtigungsdatei für Cron
/etc/cron.d/cron.deny	Verweigerungsdatei für Cron

Kapitel 28

SOFTWAREMANAGEMENT

Unter Solaris werden Softwarepakete mit den *pkg* Tools installiert und deinstalliert.

28.1 Allgemeine Informationen

Jedes SystemV Betriebssystem hat eine besondere Schnittstelle die sich ABI[1] nennt. Diese Schnittstelle definiert den Aufbau einer PKG Datei. Durch ABI wird gewährleistet das Software nicht doppelt installiert wird, und wenn diese entfernt wird, auch wirklich weg ist.

Wenn Software nicht im pkg Format vorliegt, z.B. tar oder cpio, dann wird das Software Management des Betriebssystems umgangen und der Administrator muß diese Software manuell installieren und auch manuell deinstallieren. Dazu muß der Administrator die Software genau kennen.

Eine Installation einer Software führt zu einer erweiterten Prüfung und zur erweiterten Sicherheit. Geprüft wird u.a. ob die Software bereits installiert wurde, oder ob sich Programme mit gesetztem SUID oder SGID einnisten wollen. Der Administrator wird in diesen Fällen gefragt. Andere Softwaremanagementtools bieten dieses nicht.

28.2 Softwarepaketnamen

Ein Softwarepaket besitzt immer einen Namen. In den meisten Fällen fägt der Name mit grossen Buchstaben an und bezeichnen meist den Hersteller. Gefolgt vom eigendlichen kurznamen der Software. Die Pakete von Sun fangen alle mit `SUNW` an. Aber auch einige Geräte fangen so an. Pakete aus der Freewareszene die Solaris anbei liefert fangen mit `SFW` (Solaris Free Ware) an. Pakete von HTTP://WWW.SUNFREEWARE.COM fangen mit `SMC` an.

28.3 Informationen über installierte Software

Alle Informationen werden in `/var/sadm` abgelegt und sollten nicht geändert werden. Um Informationen über eine spezielle Software zu bekommen muß man wissen wie das Paket heisst. Mit dem Programm `pkginfo`[2] können weitere Informationen von Paketen abgerufen werden. Gibt man dem Programm kein Paketnamen mit auf den Weg druckt es alle aus. Das kann man dann z.B. durch `grep`[3] jaben und dort nach speziellen Informationen suchen :

[1] `ABI`-Application Binary Interface

[2] `pkginfo` - Siehe Kommandoreferenz Seite 353

[3] `grep` - Siehe Kommandoreferenz Seite 281

```
raven:whurst $ pkginfo -l SMCgv
   PKGINST:  SMCgv
      NAME:  gv
 KATEGORIE:  application
      ARCH:  sparc
   VERSION:  3.5.8
   BASEDIR:  /usr/local
    VENDOR:  Johannes Plass
    PSTAMP:  Steve Christensen
  INSTDATE:  Mai 16 2000 14:47
     EMAIL:  steve@smc.vnet.net
    STATUS:  vollstndig installiert
   DATEIEN:     69 installierte Pfadnamen
                 5 gemeinsam genutzte Pfadnamen
                10 Verzeichnisse
                 2 ausfhrbare Dateien
              1778 Blcke verwendet (ca.)

raven:whurst $ pkginfo | grep printer
raven:whurst $ pkginfo | grep print
system      SFWmpage       mpage - print multiple pages per sheet
raven:whurst $ pkginfo | grep lp
system      SUNWd8he       German UTF-8 CDE runtime Help
system      SUNWdehe       German CDE runtime Help
system      SUNWdehed      German Help Developer Environment
system      SUNWdepcz      PC File Viewer German help and icons
system      SUNWdodcv      German OPEN LOOK (R) document and help ....
system      SUNWdthez      Desktop Power Pack Help Volumes
system      SUNWeudhr      UTF-8 L10N For CDE Help Runtime
system      SUNWeudhs      UTF-8 L10N For CDE Help Runtime
system      SUNWlpmsg      LP Alerts
system      SUNWoldcv      OPEN LOOK document and help viewer applications
system      SUNWscplp      SunSoft Print - Source Compatibility, (Usr)
system      SUNWslpr       SLP, (Root)
system      SUNWslpu       SLP, (Usr)
system      SUNWslpx       SLP (64-bit)
application SUNWsmtvh      ShowMe TV Online Help
raven:whurst $
```

Bildschirmausschnitt 28.3.1: Auskünfte von Paketen einholen

Das Beispiel zeigt den verzweifelten Versuch den Druckermechanismus Pakete (lp) zu finden.

Eins jedoch geht überhaupt nicht. Man kann nicht feststellen zu welchem Paket eine einzelnen Datei gehört. Das muß man leider zu Fuß machen. Es existiert eine Datei die alle Dateien mit Zugriffsberechtigungen aller Pakete speichert. In dieser Datei muß man jetzt nur noch den `grep` darauf ansetzen :

```
aven:whurst $ grep /etc/hosts /var/sadm/install/contents
/etc/hosts=./inet/hosts s none SUNWcsr
raven:whurst $ grep /usr/bin/ls /var/sadm/install/contents
/usr/bin/ls f none 0555 root bin 18844 34850 947116674 SUNWcsu
raven:whurst $ grep /usr/bin/lp /var/sadm/install/contents
/usr/bin/lp f none 4511 root lp 22456 12700 947116840 SUNWpcu
/usr/bin/lp_1251 f none 0755 bin bin 172 14147 939822640 SUNW1251f
/usr/bin/lpget f none 0511 root lp 5656 30384 947116821 SUNWpcu
/usr/bin/lpset f none 4511 root lp 7116 54809 947116827 SUNWpcu
/usr/bin/lpstat f none 4511 root lp 21592 1317 947116867 SUNWpcu
raven:whurst $
```

Bildschirmausschnitt 28.3.2: Suche nach Paketen

Ganz zum Schluß steht das verwendete Paket zu dieser Datei.

28.4 Prüfen von Paketen

Jeder Hersteller von pkg's liefert eine Datei mit in der ganz genau steht, welche Datei wie gross mit welchen Rechten installiert wird. Wird nun am System rumgefummelt kann man einzelne Pakete, oder alle, mittels `pkgchk`[4] prüfen lassen. Das Programm wirft einem dann alle Unstimmigkeiten raus :

```
raven:whurst $ su -
Sun Microsystems Inc.   SunOS 5.8       Generic February 2000
# pkgchk SMCgv
# pkgchk SUNWatfsr
ERROR: /etc/auto_master
    file size <113> expected <114> actual
    file cksum <9773> expected <9808> actual
# exit
raven:whurst $
```

Bildschirmausschnitt 28.4.1: Überprüfen von Paketen

Man könnte diesen Mechanismus mit einem Backup z.B. koppeln. Man lässt sich von allen Pakenten nur den Dateinamen ausgeben, und kopiert die so gewonnenen Dateien auf ein Band. Leider gibt es im System Konfigurationsdateien die man selbst erst erstellen muß und somit werden diese mit dieser Methode nicht erfasst.

28.5 Deinstallation von Software

Wenn man Software deinstallieren will, sollte man zuvor ersteinmal feststellen ob die Software noch irgendwo läuft oder wirklich nicht mehr benötigt wird. Sofern man sich nicht sicher ist muß man ersteimal alle Dateien finden die zu diesem Paket gehören. Auch daß geht leider nicht so einfach. Man muß wieder in der `/var/sadm/install/contents` nach suchen

[4] `pkgchk` - Siehe Kommandoreferenz Seite 351

```
raven:whurst $ grep 'SUNWpsdpr$' /var/sadm/install/contents
/kernel/drv/pcata f none 0755 root sys 34016 14381 947382698 SUNWpsdpr
/kernel/drv/sparcv9/pcata f none 0755 root sys 45824 52506 947382703 SUNWpsdpr
raven:whurst $
```

Bildschirmausschnitt 28.5.1: Suchen von Dateien zu einem Paket

Aber wie gesagt, das benötigt man nur zur Kontrolle. Man könnte jetzt in der Prozessliste nach den laufenden Prozessen suchen und diese mit den Dateien des Paketes vergleichen. ... Gut das ist erledigt ...

Wird nun ein solches Paket entfernt muß man das Programm `pkgrm`[5] benutzen :

```
raven:whurst $ su -
Sun Microsystems Inc.   SunOS 5.8       Generic February 2000
# pkgrm SUNWpsdpr

The following package is currently installed:
   SUNWpsdpr       PCMCIA ATA card driver
                   (sparc) 11.8.0,REV=2000.01.08.18.12

Do you want to remove this package? y

## Removing installed package instance <SUNWpsdpr>

This package contains scripts which will be executed with super-user
permission during the process of removing this package.

Do you want to continue with the removal of this package [y,n,?,q] y
## Verifying package dependencies.
## Processing package information.
## Executing preremove script.
## Removing pathnames in class <none>
/kernel/drv/sparcv9/pcata
/kernel/drv/sparcv9 <shared pathname not removed>
/kernel/drv/pcata
/kernel/drv <shared pathname not removed>
/kernel <shared pathname not removed>
## Updating system information.

Removal of <SUNWpsdpr> was successful.
# exit
raven:whurst $
```

Bildschirmausschnitt 28.5.2: Entfernen von Paketen

Und schon isses wech !

[5] `pkgrm` - Siehe Kommandoreferenz Seite 355

28.6 Software hinzufügen

`pkgadd`[6] .... usw ....

28.7 Patches hinzufügen

28.8 Patches entfernen

[6] `pkgadd` - Siehe Kommandoreferenz Seite 349

28.9 Praktikum

28.9.1 Paketmanagement

- Schreiben Sie ein Script mit dem Namen `pkgfind` welches die Funktionalität 'Suche nach Paketen' 28.3.2 implementiert. Das Script soll mit mindestens einem Argument aufgerufen werden. Das Script soll nun nach den übergebenen Suchbegriffen suchen und nur den Paketnamen ausgeben.

- Dummerweise sucht Ihr Script (wie im Beispiel auch) den Suchbegriff in der gesammten Zeile. Das ist eigendlich nicht gewünscht und muß geändert werden. Schreiben Sie Ihr Script so um das nur noch im ersten Feld nach dem Suchbegrif gesucht wird.

- Testen Sie Ihr Script in dem Sie das `pkgfind` mit `pkginfo` kombinieren. Indem folgendes funktioniert :
  `pkginfo ‘pkgfind uname‘` (Das ‘ steht für das Substitute)

- Ein Fehler wird auftreten, weil Ihr Script `pkgfind` alle doppelten Paketnamen mit raus wirft. Das müssen Sie Fixen.

- Wenn man nach bestimmten Paketen sucht kann `pkginfo | grep` verwendet werden. Will man jetzt jedoch auch noch rauszufinden welche Dateien mit diesem Paket installiert werden, dann ist man etwas aufgeschmissen. Solaris bietet uns kein Programm an. Das ist aber nicht schlimm. Schreiben Sie ein Script mit dem Namen `pkgfiles` das genau diese Aufgabe erledigt. Dem Script werden Paketnamen übergeben.

28.10 Lösungsansätze

Das `pkgfind` könnte z.B. mit der Bourne Shell wie folgt aussehen. Wenn Ihr Script das gleiche Ausgibt aber anderst aufgebaut ist, ist das egal :

Quelltext 28.10.1 pkgfind

```
#!/bin/sh
#
# Dieses Script findet Softwarpakete. Als Suchkriterium werden
# Suchmuster fuer Dateien uebergeben.
#
# (C) 2000 by Wolfgang Hurst
#

contentfile=/var/sadm/install/contents

# argumente pruefen
if test $# -eq 0; then
        echo "usage: $0 <filename-regex>"
        exit 1
fi

# eine schleife die alle argumente durchgeht und alle zeilen
# auf den das suchmuster passt ausgibt (nach stdout)
# (nach dem done folgt eine PIPE)
while test $# -ne 0; do

        # nach dem suchmuster suchen und zeile ausgeben
        grep '^[^ ]*'"$1" $contentfile

        # naechstes argument (suchmuster)
        shift
# schleife beenden und pipen
done | cut -d" " -f10 | sort | uniq | grep -v '^$'

# ausgabe der pipe einfach auf stdout und ende
exit 0
```

Das Script `pkgfiles` könnte so aussehen :

Quelltext 28.10.2 pkgfiles

```
#!/bin/sh
#
# Dieses Script gibt alle Dateinamen eines Paketes aus
#
# (C) 2000 by Wolfgang Hurst
#

contentfile=/var/sadm/install/contents

# argumente pruefen
if test $# -eq 0; then
        echo "usage: $0 <packetname>..."
        exit 1
fi

# eine schleife die alle argumente durchgeht und alle packete
# sucht und dessen Dateinamen ausgibt
while test $# -ne 0; do

        # nach dem suchmuster suchen und zeile ausgeben
        # wobei der Name des Paketes vor der Zeile nochmal
        # geschrieben wird.
        grep " $1"'$' $contentfile | cut -d" " -f1 | while read a; do
                echo "$1:$a"
        done

        # naechstes argument (suchmuster)
        shift
# schleife beenden und pipen
done

# ausgabe der pipe einfach auf stdout und ende
exit 0
```

TEIL IV

NETZWERKDIENSTE

- Konfiguration der Netzwerkumgebung
- Netzwerkstandard Dienste
- Druckumgebung im Netzwerk
- R-Tools
- Remote Procedure Call
- Network File System
- Network Information Service
- Network Information Service Plus
- Automounter
- Cache File System
- Network Time Protocol
- Autoinstallationssystem

Kapitel 29

KONFIGURATION DER NETZWERKUMGEBUNG

29.1 Manuelle Konfiguration der NIC

Unter Solaris kann das Programm `ifconfig`[1] zur Konfiguration von Devices benutzt werden. Das Programm kann sich die nötigen Informationen aus der /etc/inet/netmasks[2] und /etc/inet/hosts[3] selbst besorgen, sofern man es ihm sagt.

Um ein Device überhaupt konfigurieren zu können, muß sichergestellt werden das die Schnittstelle bereits vorhanden und Aufnahmefähig ist. Dazu sollte man erst mit `ifconfig -a` sich die Liste anschauen.

```
raven # ifconfig -a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
raven #
```

Bildschirmausschnitt 29.1.1: Liste der netzfähigen Devices

Wird ein Device nicht aufgelistet, es jedoch vorhanden ist, muß es erst netzfähig gemacht werden. Der Parameter *plumb* erledigt dieses :

```
raven # ifconfig hme0 plumb
raven # ifconfig hme0
hme0: flags=1000842<BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 0.0.0.0 netmask 0
raven #
```

Bildschirmausschnitt 29.1.2: Deviceaktivierung für Netz

Nun kann man das Device nach herzenslust Konfigurieren

29.1.1 Konfiguration der IP Addresse

Mit dem Parameter *inet* $< ip >$ wird nur die IP Addresse konfiguriert. Die Netzmaske und Broadcastaddresse wird jedesmal auf die Standardwerte oder auf die definierten Werte der /etc/netmasks berechnet.

[1] `ifconfig` - Siehe Kommandoreferenz Seite **??**

[2] /etc/inet/netmasks - Siehe Konfigurationsdateien Referenz Seite 421

[3] /etc/inet/hosts - Siehe Konfigurationsdateien Referenz Seite 419

```
raven # ifconfig hme0 inet 10.10.10.10
raven # ifconfig hme0
hme0: flags=1000842<BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 10.10.10.10 netmask ff000000 broadcast 10.255.255.255
raven #
```

Bildschirmausschnitt 29.1.3: Temporäre IP Konfiguration

Wird anstelle der IP Addresse ein Hostnamen eingegeben dann wird das Programm `ifconfig` sich versucht sehen den Hostnamen entsprechend aufzulösen und die IP einzusetzen.

```
raven # ifconfig hme0 inet raven2
raven # ifconfig hme0
hme0: flags=1000842<BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 10.65.7.10 netmask ff000000 broadcast 10.255.255.255
raven #
```

Bildschirmausschnitt 29.1.4: Temporäre IP Konfiguration mit Hostnamen

29.1.2 Konfiguration der Netzmaske

Mit dem Parameter *netmask* $< nm >$ wird die Maske entsprechend eingerichtet. Die Broadcastaddresse ändert sich jedoch nicht.

```
raven # ifconfig hme0 netmask 255.255.255.0
raven # ifconfig hme0
hme0: flags=1000842<BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 10.65.7.10 netmask ffffff00 broadcast 10.255.255.255
raven #
```

Bildschirmausschnitt 29.1.5: Temporäre Netzmasken Konfiguration

29.1.3 Konfiguration der Broadcastaddresse

Mit dem Parameter *broadcast* $< br >$ kann die Broadcastaddresse eingestellt werden. Es können jedoch alle Arten des Broadcastes probiert werden, selbst ungültige Werte, wie das folgende Beispiel zeigt :

```
raven # ifconfig hme0 broadcast 10.65.7.255
raven # ifconfig hme0
hme0: flags=1000842<BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 10.65.7.10 netmask ffffff00 broadcast 10.65.7.255
raven # ifconfig hme0 broadcast 20.30.40.50
raven # ifconfig hme0
hme0: flags=1000842<BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 10.65.7.10 netmask ffffff00 broadcast 20.30.40.50
raven # ifconfig hme0 broadcast 10.65.7.255
raven #
```

Bildschirmausschnitt 29.1.6: Temporäre Broadcastaddressen

Da wir das jedoch nicht wollen, habe ich sie wieder normalisiert.

29.1.4 Aktivierung der Konfiguration

Die Konfiguration kann mit *up* aktiviert und mit *down* deaktiviert werden. Bevor man das Interface nutzen will muß es aktiviert werden

```
raven # ifconfig hme0 up
raven # ifconfig hme0
hme0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 10.65.7.10 netmask ffffff00 broadcast 10.65.7.255
raven #
```

Bildschirmausschnitt 29.1.7: Temporäre Aktivierung des Interfaces

Auf das **UP** kommt es an.

29.2 Manuelle Routingeinträge

Damit nicht genug. Damit man auch mit diesem Interface arbeiten kann müssen die Routen gesetzt werden. Dazu dient das Programm `route`[4] und `netstat`[5].

29.2.1 Temporäres setzen von Routen

Das Programm `route` erwartet bein Hinzufügen einer Route das Kommando *add* gefolgt der Parameterisierung.

[4] `route` - Siehe Kommandoreferenz Seite **??**
[5] `netstat` - Siehe Kommandoreferenz Seite 333

```
raven # route add -net 10.65.8.0 10.65.7.80
add net 10.65.8.0: gateway 10.65.7.80
raven # route add -host 10.65.13.1 raven2
add host 10.65.13.1: gateway raven2
raven #
```

Bildschirmausschnitt 29.2.1: Temporäres hinzufügen von Routen

Das zweite Beispiel zeigt eine verbotene Route. Es ist eine Hostroute für ein Rechner der normalerweise nicht in diesem Netzwerk sich befinden sollte.

29.2.2 Anzeigen von Routingeinträgen

Zur Anzeige muß das Programm `netstat`[6] verwendet werden. In Verbindung mit der Option *-r* zeigt es die Routingeinträge an

```
raven # netstat -r

Routing Table: IPv4
  Destination           Gateway           Flags  Ref   Use   Interface
-------------------- -------------------- ----- ----- ------ ---------
10.65.8.0            10.65.7.80            UG        1      0
10.65.7.0            raven2                U         1      2  hme0
localhost            localhost             UH       10     54  lo0
10.65.13.1           raven2                U         1      2  hme0
raven #
```

Bildschirmausschnitt 29.2.2: Anzeigen von Routingeinträgen

Wie man sieht wurde die Lokale Route bereits erstellt.

29.2.3 Löschen von Routen

Um Routen wieder zu entfernen muß der Parameter *delete* gefolgt der Parameterisierung angegeben werden :

[6] `netstat` - Siehe Kommandoreferenz Seite 333

```
raven # route delete -net 10.65.8.0 10.65.7.80
delete net 10.65.8.0: gateway 10.65.7.80
raven # route delete -net 10.65.7.0 raven2
delete net 10.65.7.0: gateway raven2
raven # ifconfig hme0 down
raven # ifconfig hme0 unplumb
raven # ifconfig -a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
raven # netstat -r

Routing Table: IPv4
  Destination           Gateway           Flags  Ref   Use   Interface
-------------------- -------------------- ----- ----- ------ ---------
localhost            localhost             UH      10     54  lo0
raven #
```

Bildschirmausschnitt 29.2.3: Löschen von Routingeinträgen

Das Beispiel zeigte gleichzeitig noch das Runterfahren des Interfaces.

Kapitel 30

NETZWERKSTANDARD DIENSTE

30.1 Funktionsweise

30.2 Praktikum

1. Schreiben Sie ein Script mit dem Namen `rdfd`. Das Schript soll auf `stdout` alle gemounteten Dateiensystem mit dessen Belegung ausgeben. Erstellen Sie danach in der `/etc/protocols` ein Eintrag für den Port `4711` für das TCP Protocol mit dem Namen `rdf`. Fügen Sie danach Ihr Script in die Konfiguration des `inetd`s ein. Starten Sie danach den `inetd` neu. Testen Sie Ihre Konfiguration in dem Sie mittels `telnet localhost rdf` die Konfiguration testen

2. Schreiben Sie Ihr Script `rdfd` so um das es vor jeder Zeile den Hostnamen notiert

3. Schreiben Sie ein Script mit dem Namen `rdf`. Das Script soll durch alle Hosts gehen und dort mittels `telnet` die entsprechenden Daten holen

4. Schreiben Sie ein Script mit dem Namen `rdf_adv`. Das Script soll die Ausgabe von `rdf` sortieren. Die Sortierung soll absteigend des größt möglichsten Platzes sein

5. Schreiben Sie Ihr Script `rdf_adv` so um das es keine Einträge mehr anzeigt, dessen freier Platz unterhalb von 200 MB liegt

Kapitel 31

DRUCKUMGEBUNG IM NETZWERK

31.1 Funktionsweise

31.2 Konfiguration des Druckservers

31.3 Konfiguration des Druckclients

Zur Konfiguration eines Druckclient muß der Hostname und Druckername des Servers bekannt sein. Die Konfiguration erfolgt wieder mittels `lpadmin`[1]. Der folgende Ausschnit zeigt die Konfiguration auf dem Client LX mit Zugriff auf den Netzwerkdruckers der lokal auf dem RAVEN freigegeben wurde :

```
lx # lpadmin -p nethpmonotext -s raven!hpmonotext
lx # lpadmin -d nethpmonotext
lx # lp /etc/hosts
request id is nethpmonotext-1 (1 file)
lx #
```

Bildschirmausschnitt 31.3.1: Einrichting des Druckclients LX

1. Der Drucker wird definiert. Mit der Option *-s* wird ein Remoteprinter eingerichtet. WARNUNG: Bei der Verwendung der `bash` muß das Ausrufezeichen Mit einem Backslash entwertet werden !

2. Der Drucker wird als Default Printer eingetragen

3. Es wird was gedruckt

Ein `enable` oder `accept` ist bei Remoteprintern nur auf dem Server notwendig.

31.3.1 Konfiguration mit dem `admintool`

Mit dem `admintool` ist es nicht möglich auf dem Clienten ein anderen Druckernamen zu definieren wie auf dem Server. Aber genaugenommen ist das auch vollkommener blödsinn. Wenn man das `admintool` gestartet hat muß man über *Edit - Add - Access to Printer* bzw. über *Bearbeiten - Hinzufügen - Zugriff auf Drucker* gehen. Es macht sich danach ein Dialogauf in dem Die werte entsprechend eingetragen werden. Das folgende Beispiel zeigt die Konfiguration zum Postscriptdrucker auf dem RAVEN :

[1] `lpadmin` - Siehe Kommandoreferenz Seite 311

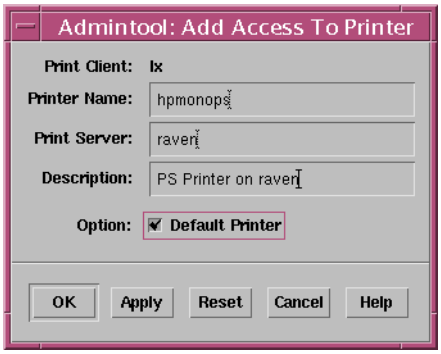


Abbildung 31.1: Hinzufügen eines Remoteprinters mit dem `admintool`

Kapitel 32

R-TOOLS

Kapitel 33

REMOTE PROCEDURE CALL

Kapitel 34

NETWORK FILE SYSTEM

Das NFS[1] dient zur Verteilung von Dateien und Verzeichnissen im Netzwerk. Es gibt ein Server, der NFS-Server genannt wird, der auf seiner eigenen lokalen Festplatte ein Verzeichnis für die Benutzung im Netzwerk freigibt. Ein NFS-Client kann diese Resource dann benutzen.

34.1 Allgemein

Das NFS war nicht immer bei UNIX dabei. Als erstes kam SystemV mit dem DFS[2] auf den Markt. Das DFS kam mit zwei variationen auf den Markt zum einen das RFS[3], welches sich jedoch nicht durchgesetzt hat, und zum zweitem das NFS. Warum ich das hier schreibe ? Nun im Solarissystem fangen die Konfigurationsdateien alle mit *dfs* an. Jetzt wissen Sie warum *dfs* und nicht *nfs*.

Das Solaris NFS System, wechlech von SystemV abstammt, unterscheidet sich in der Konfiguration ganz erheblich vom BSD Standard. Den BSD Standard, den auch Linux nutzt, habe ich in der TCP/IP Guide beschrieben. Achtung: Ich wollte den Teil mal auslagern, er könnte sich jetzt vielleicht wo anderst befinden, und ich habe vergessen diesen Text hier anzupassen.

34.2 Terminologie

34.2.1 Versionen

Von NFS gibt es 3 Versionen.

Version 1 Die Version 1 wird nicht mehr eingesetzt. Da ich diese auch nicht kenne, sag ich nicht viel dazu.

Version 2 Wurde von Sun Microsystems in Verbindung mit RPC neu implementiert und ist unter vielen UNIX Derivaten immer noch Standard. Es kann zur Übertragung nur das UDP verwenden und besitzt auch keine Verschlüsselungsmechanismen wie Kerberos. Linux kann zur Zeit nur Version 2.

Version 3 Die Version 3 kann sowohl über UDP als auch TCP arbeiten. Es bietet Verschlüsselungsmechanismen mit Kerberos. Und kann SecureRPC benutzen. Nur mit Version 3 kann auch WebNFS implementiert werden. Die Version 3 kann auch mit großen bzw. sehr großen Datenblöcken umgehen. Das Limit liegt bei 4 GB (Version 2 nur 8 KB).

34.2.2 NFS-Server

Ein NFS-Server ist ein normaler UNIX Host. Der NFS-Server hat zusätzlich die Aufgabe lokale Dateien und Verzeichnisse im Netzwerk freizugeben. Zum Betrieb eines NFS-Servers muß RPC installiert sein.

[1] NFS-Network File System
[2] DFS-Distributed File System
[3] RFS-Remote File System

34.2.3 Share

Um einem NFS-Server zu sagen was er im Netzwerk zur Verfügungstellen soll, muß man ein Verzeichnis kennzeichnen. Es kann nur ein Verzeichnis zur Freigabe gestellt werden, einzelnen Dateien sind zwar auch möglich, werden jedoch nicht direkt benutzt. Wird ein solches verzeichnis freigegeben nennt man dieses eine Share.

34.2.4 NFS-Client

Der NFS-Client kann sich eine Share vom NFS-Server mounten. Dazu wird der gewohnte Mechanismus via Mount verwendet. Um ein Share zumounten muß auch hier ein Mointpoint existieren, etc.pp. Die gleichen Bedingungen wie das Mounten einer Slice.

34.2.5 Operationen

Eine Operation ist ein Befehl den der NFS-Client an den NFS-Server sendet, damit der das tut was der NFS-Client will. Diese Operationen werden im Protokoll entsprechend gesetzt. Die Operationen haben jedoch auch ein menschlichen Namen :

null Diese Operation macht nix.

root Besorgt sich das Fileaccesshandle vom Rootdevice des NFS-Servers. Diese Funktion wurde von Version 1 auf Kompatiblitätsgründen mit in Version 2 übernommen. Diese Operation wird nicht mehr genutzt da es sie in Version 3 auch nicht mehr gibt.

lookup Durchsucht ein Verzeichnis nach einem Dateinamen und gibt dessen Fileaccesshandle und dessen Rechte zurück.

access Seit der Version 3 gibt es ihn ! Der Server prüft ob ein Benutzer Rechte an einer Datei besitzt oder nicht. Diese Operation ist nötig wenn das Dateiensystem vom NFS-Server ACL's unterstützt. Da Version 2 keine ACL's untersützt kann es dort zu erheblichen Problemen kommen, wenn man beides vermischt.

readlink Liest einen verweisenden Dateinamen aus einem symbolischen Link. Greift man auf ein symbolischen Link über NFS zu, dann wird der Link erst auf dem Server aufgelöst. Würde der NFS-Client dieses nicht beachten würde es bei symbolischen Links ständig zu Fehlermeldungen kommen, wenn das Root des NFS-Server sich in einem anderen Verzeichnis befinden würde als der Mountpoint auf dem NFS-Client. Der NFS-Client würde versuchen eine Datei zu öffnen die auf dem Server nicht freigegeben wurde, oder nicht existiert.

getattr Liest einige INode Daten des Fileaccesshandles aus. Das betrifft die Rechte, die Zeitstempel, Groesse und den Typ. Nicht die ACLs.

setattr Schreibt die Daten wieder in die INode zurück. Achtung: ACLs nicht.

read Liest ein Datenblock aus dem Fileaccesshandle. Die Version 2 unterstützt nur die größe von maximal 8 KB pro Operation. Die Version 3 hingegen kann mit bis zu 4 GB großen Datenblöcken umgehen.

write Schreibt ein Datenblock in den Fileaccesshandle. Die Version 2 unterstützt nur die größe von maximal 8 KB pro Operation. Die Version 3 hingegen kann mit bis zu 4 GB großen Datenblöcken umgehen.

wrcache Diese Operation veranlasst den NFS-Server die Daten im Schreibbuffer sofort und jetzt wegzuschreiben. Diese Funktionalität wurde von der Version 1 übernommen und sollte nicht mehr verwendet werden. Weil :

commit macht das gleiche wie *wrcache* jedoch für Version 3.

create Erstellt eine Datei

mknod Erst ab der Version 3 ist es möglich über NFS eine Gerätedatei wie Pipes, FIFO oder sonstigen zu erstellen.

remove Löschen einer Datei

rename Umbenennen einer Datei

link Erstellt ein Hardlink

symlink Erstellt ein symbolischen Link, der jedoch immer relativ zum Client ist

mkdir Erstellt ein Verzeichnis

rmdir Löscht ein Verzeichnis, aber nur wenn es leer ist

readdir Gibt die Liste eines Verzeichnisses zurück. Der Befehl verursacht sehr viel Traffic wenn man ein `ls -l` absetzt. Da es für jede einzelne Datei ein *lookup* und ein *getattr* machen muß.

readdir+ Ab Version 3 wurde die *readdir* Operation verbessert. *readdir+* gibt nicht nur das Verzeichnisslisting zurück sondern auch die Attribute (*getatttr*) aller Einträge. Das macht sich sehr gut beim Einsatz von `ls -l`

pathconf Hat irgendwas mit POSIX Dateiensystemen zu tun. Mehr weiß ich auch nicht. Jedoch erst ab Version 3 verfügbar

statfs Gibt Auskunft über die maximale Dateigröße. Ist in Version 3 nicht mehr definiert.

fsstat Macht das gleiche wie *statfs* jedoch nur ab Version 3

fsinfo Geht noch ein Schritt weiter als *fsstat* und gibt noch den freinen Platz und die freien INodes aus. Nur ab Version 3.

Die nachfolgende Tabelle notiert die Verfügbarkeit im Detail :

Tabelle 34.1: NFS Operationen Version 2 und Version 3

Name	V 2	V 3		Name	V 2	V 3		Name	V 2	V 3
null	✓	✓		root	✓	✗		lookup	✓	✓
access	-	✓		readlink	✓	✓		getattr	✓	✓
setattr	✓	✓		read	✓	✓		write	✓	✓

Name	V 2	V 3		Name	V 2	V 3		Name	V 2	V 3
commit	-	✓		wrcache	✓	-		create	✓	✓
mknod	-	✓		remove	✓	✓		rename	✓	✓
link	✓	✓		symlink	✓	✓		mkdir	✓	✓
rmdir	✓	✓		readdir	✓	✓		readdir+	-	✓
pathconf	-	✓		statfs	✓	✓		fsstat	-	✓
fsinfo	-	✓								

34.2.6 Operation Cache Management

Einige von den oben genannten Operationen müssen wiederholbar sein, ohne das es zu Fehlern kommt. Man schaue sich folgenden Fall an :

Der Benutzer will eine Datei löschen. Der NFS-Client sendet nun ein *remove* zum NFS-Server. Der NFS-Server führt die Operation aus und sendet dem NFS-Client die Bestätigung. Diese Bestätigung jedoch bleibt irgendwo im Netzwerk hängen. Der NFS-Client bekommt nun leichte Schwierigkeiten und sendet die Operation einfach nochmal. Der NFS-Server bekommt nun nochmal ein *remove*. Der NFS-Server kann die Operation nicht mehr ausführen weil die Datei schon gelöscht wurde. Normalerweise müsste der NFS-Server nun eine Fehlermeldung an den NFS-Client senden. Da jedoch die *remove* Operation noch in seinem Cache liegt, weiss der NFS-Server daß die zweite *remove* Operation eigentlich die gleiche ist wie die erste. Das sieht er an seinem Cache. Also sendet der NFS-Server dem NFS-Client jetzt eine positive Bestätigung zurück. Sollte die Bestätigung zum NFS-Client durchkommen, reagiert der NFS-Client ganz normal und bekommt nicht mit, daß der NFS-Server richtig was zu tun hatte.

Daher gibt es wiederholbare und nicht Wiederholbare Befehle. Der *remove* ist nicht wiederholbar ! Da bei der ersten Opertation die Datei weg ist. Auch *mkdir* ist nicht wiederholbar, da wenn ein Verzeichnis bei der ersten Operation angelegt wird, gibt die zweite Operation ein Fehler: *directory already exists* zurück.

Die Operation *read* ist beliebig oft wiederholbar ohne das es bei den Wiederholungen zu Fehlern kommen könnte. Dann wird ebend der Datenblock achtzich mal übers Netz geschaufelt.

Die folgende Tabelle ordnet die wiederholbaren und nicht wiederholbarewn Operationen :

Tabelle 34.2: Wiederholbare und nicht wiederholbare NFS Operationen

wiederholbare Operationen	nicht wiederholbare Operationen
null, root, lookup, access, readlink, getattr, read, write, commit, wrcache, readdir, readdir+, pathconf, statfs, fsstat, fsinfo	setattr, create, rename, mknod, remove, mkdir, rmdir, link, symlink

34.2.7 Fileaccesshandle

Für jedes *lookup* oder *mount* Kommando vom NFS-Client bekommt der NFS-Client ein Fileaccesshandle vom NFS-Server zugewiesen. Greift der NFS-Client nun auf eine Datei zu, benötigt er zuerst ein *lookup* auf die Datei um dessen Fileaccesshandle zu bekommen. Will der NFS-Client ein Verzeich-

nis lesen benötigt er auch ein Fileaccesshandle. Die Operation *mount* liefert das Fileaccesshandle des Rootverzeichnisses zurück. Und somit kann der NFS-Client das Root durchsuchen.

34.3 Konfiguration des NFS-Servers

Der NFS-Server ist in den Softwarepaketen `SUNWcsr` und `SUNWcsu`. Diese Pakete gehören zum Coresystem und müssen nicht nach installiert werden. Alle nötigen Daemons befinden sich im `/usr/lib/nfs` Verzeichnis. Als Startmodul wird das `/etc/init.d/nfs.server` verwendet und im Runlevel 3 wird es mittels `/etc/rc3.d/S15nfs.server` gestartet.

Wirft man jedoch ein Blick in die `/etc/init.d/nfs.server` stellt man fest, daß der NFS-Server erst dann gestartet wird wenn er wirklich erst benötigt wird. Wird ein System gestartet ohne das er vorher ein NFS-Server war und man will schnell mal ein Share erstellen, dann wird das nicht funktionieren, weil die benötigten Daemons nicht gestartet wurden.

34.3.1 Einrichten von Shares

Um ein Share hinzuzufügen wird die Datei `/etc/dfs/dfstab` vom Startmodul analysiert. Befinden sich in dieser Datei keine Shares dann wird der NFS-Server nicht gestartet. Die `/etc/dfs/dfstab` kann als Shellscript verwendet werden, d.h. man kann Shares direkt in diese Datei notieren und diese mittels `sh /etc/dfs/dfstab` starten. Das erspart den Stopp und Startlauf des NFS-Servers mittels Startmodul. Da die Datei nur einmal beim Hochfahren ausgelesen wird.

Ein Share kann jederzeit mit `share`[4] erstellt werden. Wenn der NFS-Server bereits läuft kann das Share sofort benutzt werden. Dem `share` muß das Filesystem bekannt gegeben werden, wie es freigegeben werden soll. Zu fast 100% ist dies NFS. Wird die Angabe nicht gemacht wird diese aus der `/etc/dfs/fstypes` geholt.

Der folgende Ausschnitt zeigt ein Beispiel:

Quelltext 34.3.1 Beispiel einer `/etc/dfs/dfstab`

```
#
share -F nfs -o ro,root=eagle -d "SPARC:/usr/local" /usr/local
share -F nfs -o rw=crow,ro,root=crow /usr/local/bin.sparcv9
```

34.3.2 Hinzufügen von Shares

Wie bereits angedeutet kann ein Share kurz mal so gemacht werden in dem man das Programm `share` verwendet :

```
raven # share -o ro /opt
raven #
```

Bildschirmausschnitt 34.3.1: Beispiele von `share`

[4] `share` - Siehe Kommandoreferenz Seite **??**

34.3.3 Anzeige von Shares

Um eine Liste von allen Shares zubekommen die zur Zeit verfügbar sind muß `share` einfach nur ohne Argumente aufgerufen werden :

```
raven # share
- /usr/local              ro,root=eagle        "SPARC:/usr/local"
- /usr/local/bin.sparcv9 rw=crow,ro,root=crow ""
- /opt                    ro                   ""
raven #
```

Bildschirmausschnitt 34.3.2: Beispiele von `share`

Alternativ kann die `/etc/dfs/sharetab` verwendet werden um alle Shares ausfindig zu machen.

34.3.4 Entfernen eines Shares

Zum entfernen eines Shares kann `unshare`[5] verwendet werden. Man sollte jedoch vorsichtig sein. Wenn ein NFS-Client das Share noch gemountet hat verliet der NFS-Client nach dem `unshare` den kompletten Baum ab dem Mountpoint. Es erfolgt auf dem Server keine Warnung oder Fehlermeldung !

```
raven # unshare /opt
raven #
```

Bildschirmausschnitt 34.3.3: Beispiele von `unshare`

34.3.5 Übersicht der gemounteten Shares

Unter einem SystemV System, wie Solaris nun mal eins ist, kann `dfshares`[6] verwendet werden um sich die freigegebenen Reources anzuschauen. Oder man kann das BSD übliche `showmount`[7] verwenden. Beide Programme können auch die Shares von anderen NFS-Servern anzeigen, jedoch nur `showmount` ist in der Lage die mountings von NFS-Clients sich anzeigen zulassen. Bei SystemV muß `dfmounts`[8] verwendet werden :

[5] `unshare` - Siehe Kommandoreferenz Seite **??**
[6] `dfshares` - Siehe Kommandoreferenz Seite **??**
[7] `showmount` - Siehe Kommandoreferenz Seite **??**
[8] `dfmounts` - Siehe Kommandoreferenz Seite **??**

```
raven # dfshares cheetah
RESOURCE                                  SERVER ACCESS    TRANSPORT
   cheetah:/home                         cheetah  -         -
   cheetah:/                             cheetah  -         -
raven # dfmounts cheetah
RESOURCE SERVER PATHNAME                   CLIENTS
  -      cheetah /home                     eagle....,puma....,raven....
  -      cheetah /home/.SPARC_USR_LOCAL   crow.hurst.pnet
  -      cheetah /home/whurst              falcon.hurst.pnet
raven # showmount -a
eagle.hurst.pnet:/opt
raven # showmount -e falcon
export list for falcon:
/var/spool/mail *.hurst.pnet
/               *.hurst.pnet
raven # showmount -a falcon
eagle.hurst.pnet:/
eagle.hurst.pnet:/var/spool/mail
raven.hurst.pnet:/var/spool/mail
tiger.hurst.pnet:/var/spool/mail
raven #
```

Bildschirmausschnitt 34.3.4: Beispiele von `unshare`

Das `showmount -a` und `dfmounts` zeigt die `/etc/rmtab` an in der alle von NFS-Client gemounteten Mountings eingetragen sind.

34.3.6 Und nun alles auf einmal

Mit dem Befehl `shareall`[9] und `unshareall`[10] werden entweder alle Resourcen geshared oder alle Sharings ungeshared. Oh man, was für ein Satz ...

34.4 Konfiguration des NFS-Client

Der NFS-Client ist in den Softwarepaketen `SUNWcsr` und `SUNWcsu`. Diese Pakete gehören zum Coresystem und müssen nicht nach installiert werden. Alle nötigen Daemons befinden sich im `/usr/lib/nfs` Verzeichnis. Als Startmodul wird das `/etc/init.d/nfs.client` verwendet und im Runlevel 2 wird es mittels `/etc/rc2.d/S73nfs.client` gestartet.

Der NFS-Client benötigt auch das RPC.

34.4.1 Mounten von Shares

Um ein Share zu mounten kann `mount`[11] verwendet werden. Das Programm erwartet den Namen oder die IP Addresse des NFS-Servers, das Share und den Mointpoint.

[9] `shareall` - Siehe Kommandoreferenz Seite **??**
[10] `unshareall` - Siehe Kommandoreferenz Seite **??**
[11] `mount` - Siehe Kommandoreferenz Seite **??**

```
mount <servername oder ip>:<share> <mointpoint>
```

Es gelten hier die gleichen Bestimmungen wie beim Mounten einer Slice. Der Mointpoint muß existieren, er darf nicht zweimal verwendet werden etc.pp.

```
eagle # mount raven:/opt /mnt
eagle # mount | grep remote
/home on cheetah:/home remote/read/write/setuid/rsize=8192.........
/var/mail on falcon:/var/spool/mail remote/read/write/setuid.......
/mnt on raven:/opt remote/read/write/setuid/dev=2e40007 on Sun.....
eagle #
```

Bildschirmausschnitt 34.4.1: Mounten von Shares

Praktikum

1. Geben Sie auf Ihrem Host das /usr Verzeichnis als Readonly frei. Beachten Sie das es das erste Share ist und entsprechend vorgehen müssen. Eintrag in der /etc/dfs/dfstab und NFS-Server Manuell starten.
2. Prüfen Sie mit dem geeigneten Tool ob Ihr Nachbar schon fertig ist.
3. Mounten Sie nun das Share Ihres Nachbarn nach /mnt. Hat dieses geklappt sollte ein `ls /mnt` die Liste des /usr Verzeichnisses des Servers zeigen.
4. Unmounten und Unsharen Sie alles wieder. Lassen jedoch den Eintrag in der /etc/dfs/dfstab so wie der ist.
5. Jeder von Ihnen shared nun irgendwelche Verzeichnisse als Readonly.
6. Jeder von Ihnen mountet nun quer durch die Klasse irgendwelche Resourcen. Legen Sie einfach unter /mnt weitere Mountpoints an.
7. Versuchen Sie nun festzustellen wer alles auf Ihrem Host sich ein Share gemountet hat. Prüfen Sie ob es stimmt.
8. Beseitigen Sie das Chaos indem Sie wieder alles Unmounten und Unsharen

Lösungsansätze

1. Die /etc/dfs/dfstab sollte folgenden Eintrag besitzen : `share -o ro /usr` und der NFS-Server wird mittels `/etc/init.d/nfs.server start` gestartet.
2. Geeignet wären `showmount -e` oder `dfmounts`
3. `umount /....` und `unshareall`
4. `mount` $<rechnername>$`:/usr /mnt`
5. `share -o ro /something-else`
6. `mount` $<somewhere>$`:`$<somewhat>$ `/....`
7. Man muß `dfshares` benutzen um festzustellen wer was gemountet hat.
8. `umount /....` und `unshareall`

34.5 Rootzugriffe über NFS

34.6 User-ID Puzzle

34.7 Zusammenfassung

Tabelle 34.3: Programme und Konfigurationsdateien für NFS

Programm / Datei	**Bedeutung**
share[12]	Programm zur Definierung von Shares
unshare[13]	Löschen einer Share
showmount[14]	Anzeige von Shares und benutzen Shares von irgendeinem NFS-Server
dfshares[15]	Anzeige aller Shares eines Servers
dfmounts[16]	Anzeige aller vom NFS-Client gemounteten Shares
mount[17]	Mounten einer Share
umount[18]	UnMounten einer Share
/etc/dfs/dfstab	Standardshares die beim Hochfahren installiert werden
/etc/dfs/sharetab	Liste aller Shares. Besser ist jedoch direkt `share` aufzurufen
/etc/dfs/fstypes	Liste der unterstützten Dateiensysteme die man als Share definieren kann
/etc/rmtab	Liste der gemounteten Shares von NFS-Clients. Man sollte `showmount -a` verwenden
/usr/lib/nfs/	Daemons für NFS
/etc/init.d/nfs.server	Startmodul für den NFS-Server
/etc/init.d/nfs.client	Startmodul für den NFS-Client. Der NFS-Server benötigt das Modul auch.
[19]	

[12] `share` - Siehe Kommandoreferenz Seite **??**
[13] `unshare` - Siehe Kommandoreferenz Seite **??**
[14] `showmount` - Siehe Kommandoreferenz Seite **??**
[15] `dfshares` - Siehe Kommandoreferenz Seite **??**
[16] `dfmounts` - Siehe Kommandoreferenz Seite **??**
[17] `mount` - Siehe Kommandoreferenz Seite **??**
[18] `umount` - Siehe Kommandoreferenz Seite **??**
[19] - Siehe Kommandoreferenz Seite **??**

Praktikum

1. a

Lösungsansätze

1. a

Kapitel 35

NETWORK INFORMATION SERVICE

35.1 Funktionsweise

35.2 Konfiguration des Servers

35.3 Konfiguration des Clients

NETWORK INFORMATION SERVICE PLUS

36.1 Konfiguration des Servers

```
PATH=$PATH:/usr/lib/nis

eagle # nisserver -r -d hurst.nis.
This script sets up this machine "eagle" as an NIS+
root master server for domain hurst.nis..

Domain name             : hurst.nis.
NIS+ group              : admin.hurst.nis.
NIS (YP) compatibility  : OFF
Security level          : 2=DES

Is this information correct? (type ’y’ to accept, ’n’ to change) y



This script will set up your machine as a root master server for
domain hurst.nis. without NIS compatibility at security level 2.

Use "nisclient -r" to restore your current network service environment.

Do you want to continue? (type ’y’ to continue, ’n’ to exit this script) y



setting up domain information "hurst.nis." ...

setting up switch information ...

running nisinit ...
This machine is in the "hurst.nis." NIS+ domain.
Setting up root server ...
All done.

starting root server at security level 0 to create credentials...
```

```
running nissetup to create standard directories and tables ...
org_dir.hurst.nis. created
groups_dir.hurst.nis. created
passwd.org_dir.hurst.nis. created
group.org_dir.hurst.nis. created
auto_master.org_dir.hurst.nis. created
auto_home.org_dir.hurst.nis. created
bootparams.org_dir.hurst.nis. created
cred.org_dir.hurst.nis. created
ethers.org_dir.hurst.nis. created
hosts.org_dir.hurst.nis. created
ipnodes.org_dir.hurst.nis. created
mail_aliases.org_dir.hurst.nis. created
sendmailvars.org_dir.hurst.nis. created
netmasks.org_dir.hurst.nis. created
netgroup.org_dir.hurst.nis. created
networks.org_dir.hurst.nis. created
protocols.org_dir.hurst.nis. created
rpc.org_dir.hurst.nis. created
services.org_dir.hurst.nis. created
timezone.org_dir.hurst.nis. created
client_info.org_dir.hurst.nis. created
auth_attr.org_dir.hurst.nis. created
exec_attr.org_dir.hurst.nis. created
prof_attr.org_dir.hurst.nis. created
user_attr.org_dir.hurst.nis. created
audit_user.org_dir.hurst.nis. created

adding credential for eagle.hurst.nis...
Enter login password:




Retype password:

creating NIS+ administration group: admin.hurst.nis. ...
adding principal eagle.hurst.nis. to admin.hurst.nis. ...

restarting NIS+ root master server at security level 2 ...
starting NIS+ password daemon ...
starting NIS+ cache manager ...

This system is now configured as a root server for domain hurst.nis.
You can now populate the standard NIS+ tables by using the
```

```
nispopulate script or /usr/lib/nis/nisaddent command.
eagle #
```

```
master1# nispopulate -F -p /nis+files -d doc.com.
          NIS+ domain name : doc.com.
          Directory Path : /nis+files
          Is this information correct? (type 'y' to accept, 'n' to change)
```

```
eagle # nispopulate -Y -d hurst.nis. -h puma -a 10.65.13.202

NIS+ domain name                : hurst.nis.
NIS (YP) domain                 : hurst.nis
NIS (YP) server hostname        : puma

Is this information correct? (type 'y' to accept, 'n' to change)


This script will populate the standard NIS+ tables for domain
hurst.nis. from the NIS (YP) maps in domain hurst.nis:
auto_master auto_home ethers group hosts ipnodes networks passwd protocols se

**WARNING:  Interrupting this script after choosing to continue
may leave the tables only partially populated.  This script does
not do any automatic recovery or cleanup.

Do you want to continue? (type 'y' to continue, 'n' to exit this script)
```

```
eagle # mkdir /nis+files
eagle # cd /nis+files/
eagle # cp /net/10.65.13.202/etc/yp/hosts .
eagle # ls
hosts



eagle # nispopulate -F -p /nis+files -d hurst.nis.

NIS+ domain name                : hurst.nis.
Directory Path                  : /nis+files

Is this information correct? (type 'y' to accept, 'n' to change) y


This script will populate the standard NIS+ tables for domain
hurst.nis. from the files in /nis+files:
auto_master auto_home ethers group hosts ipnodes networks passwd protocols services

**WARNING:  Interrupting this script after choosing to continue
may leave the tables only partially populated.  This script does
not do any automatic recovery or cleanup.

Do you want to continue? (type 'y' to continue, 'n' to exit this script)









**WARNING: file /nis+files/auto_master does not exist!
        auto_master table will not be loaded.

**WARNING: file /nis+files/auto_home does not exist!
        auto_home table will not be loaded.
```

```
**WARNING: file /nis+files/ethers does not exist!
        ethers table will not be loaded.

**WARNING: file /nis+files/group does not exist!
        group table will not be loaded.

populating hosts table from file /nis+files/hosts...
hosts table done.

Populating the NIS+ credential table for domain hurst.nis.
from hosts table.

dumping hosts table...
loading credential table...
nisaddcred: domain of principal 'cheetah.hurst.pnet.hurst.nis.' does not matc
Should only add DES credential of principal in its home domain
nisaddcred: unable to create credential.
nisaddcred: domain of principal 'crow.hurst.pnet.hurst.nis.' does not match d
Should only add DES credential of principal in its home domain
nisaddcred: unable to create credential.
nisaddcred: domain of principal 'eagle.hurst.pnet.hurst.nis.' does not match
Should only add DES credential of principal in its home domain
nisaddcred: unable to create credential.
nisaddcred: domain of principal 'falcon.hurst.pnet.hurst.nis.' does not match
Should only add DES credential of principal in its home domain
nisaddcred: unable to create credential.
nisaddcred: domain of principal 'lx.hurst.pnet.hurst.nis.' does not match des
Should only add DES credential of principal in its home domain
nisaddcred: unable to create credential.
nisaddcred: domain of principal 'panther.hurst.pnet.hurst.nis.' does not matc
Should only add DES credential of principal in its home domain
nisaddcred: unable to create credential.
nisaddcred: domain of principal 'puma.hurst.pnet.hurst.nis.' does not match d
Should only add DES credential of principal in its home domain
nisaddcred: unable to create credential.
nisaddcred: domain of principal 'raven.hurst.pnet.hurst.nis.' does not match
Should only add DES credential of principal in its home domain
nisaddcred: unable to create credential.
nisaddcred: domain of principal 'shark.hurst.pnet.hurst.nis.' does not match
Should only add DES credential of principal in its home domain
nisaddcred: unable to create credential.
nisaddcred: domain of principal 'tiger.hurst.pnet.hurst.nis.' does not match
Should only add DES credential of principal in its home domain
nisaddcred: unable to create credential.
```

```
The credential table for domain hurst.nis. has been populated.

The password used will be nisplus.


**WARNING: file /nis+files/ipnodes does not exist!
        ipnodes table will not be loaded.

**WARNING: file /nis+files/networks does not exist!
        networks table will not be loaded.

**WARNING: file /nis+files/passwd does not exist!
        passwd table will not be loaded.

**WARNING: file /nis+files/protocols does not exist!
        protocols table will not be loaded.

**WARNING: file /nis+files/services does not exist!
        services table will not be loaded.

**WARNING: file /nis+files/rpc does not exist!
        rpc table will not be loaded.

**WARNING: file /nis+files/netmasks does not exist!
        netmasks table will not be loaded.

**WARNING: file /nis+files/bootparams does not exist!
        bootparams table will not be loaded.

**WARNING: file /nis+files/netgroup does not exist!
        netgroup table will not be loaded.

**WARNING: file /nis+files/aliases does not exist!
        mail_aliases table will not be loaded.

**WARNING: file /nis+files/timezone does not exist!
        timezone table will not be loaded.

**WARNING: file /nis+files/auth_attr does not exist!
        auth_attr table will not be loaded.

**WARNING: file /nis+files/exec_attr does not exist!
        exec_attr table will not be loaded.

**WARNING: file /nis+files/prof_attr does not exist!
        prof_attr table will not be loaded.
```

```
**WARNING: file /nis+files/user_attr does not exist!
        user_attr table will not be loaded.

**WARNING: file /nis+files/audit_user does not exist!
        audit_user table will not be loaded.

**WARNING: file /nis+files/shadow does not exist!
        passwd table will not be loaded.



Credentials have been added for the entries in the
hosts table(s).  Each entry was given a default
network password (also known as a Secure-RPC password).
This password is:

                  nisplus

Use this password when the nisclient script requests the
network password.


nispopulate failed to populate the following tables:
 auto_master auto_home ethers group ipnodes networks passwd protocols service
eagle #
```

```
eagle # niscat hosts.org_dir
localhost localhost 127.0.0.1
localhost loghost 127.0.0.1
falcon.hurst.pnet falcon.hurst.pnet 10.65.13.1
falcon.hurst.pnet falcon 10.65.13.1
falcon.hurst.pnet mail 10.65.13.1
falcon.hurst.pnet pop3 10.65.13.1
falcon.hurst.pnet www 10.65.13.1
eagle.hurst.pnet eagle.hurst.pnet 10.65.13.4
eagle.hurst.pnet eagle 10.65.13.4
tiger.hurst.pnet tiger.hurst.pnet 10.65.13.7
tiger.hurst.pnet tiger 10.65.13.7
cheetah.hurst.pnet cheetah.hurst.pnet 10.65.13.8
cheetah.hurst.pnet cheetah 10.65.13.8
raven.hurst.pnet raven.hurst.pnet 10.65.13.10
```

```
raven.hurst.pnet raven 10.65.13.10
shark.hurst.pnet shark.hurst.pnet 10.65.13.11
shark.hurst.pnet shark 10.65.13.11
lx.hurst.pnet lx.hurst.pnet 10.65.13.13
lx.hurst.pnet lx 10.65.13.13
crow.hurst.pnet crow.hurst.pnet 10.65.13.72
crow.hurst.pnet crow 10.65.13.72
panther.hurst.pnet panther.hurst.pnet 10.65.13.81
panther.hurst.pnet panther 10.65.13.81
puma.hurst.pnet puma.hurst.pnet 10.65.13.202
puma.hurst.pnet puma 10.65.13.202
puma.hurst.pnet nis 10.65.13.202
puma.hurst.pnet home 10.65.13.202
eagle #
```

```
eagle # nisls
hurst.nis.:
org_dir
groups_dir
eagle # nisls org_dir
org_dir.hurst.nis.:
passwd
group
auto_master
auto_home
bootparams
cred
ethers
hosts
ipnodes
mail_aliases
sendmailvars
netmasks
netgroup
networks
protocols
rpc
services
timezone
client_info
auth_attr
```

```
exec_attr
prof_attr
user_attr
audit_user
eagle # nisls -l org_dir
org_dir.hurst.nis.:
T ----rmcdrmcdr--- eagle.hurst.nis. Thu Jun 22 10:35:40 2000 passwd
T ----rmcdrmcdr--- eagle.hurst.nis. Thu Jun 22 10:35:41 2000 group
T r---rmcdrmcdr--- eagle.hurst.nis. Thu Jun 22 10:35:41 2000 auto_master
T r---rmcdrmcdr--- eagle.hurst.nis. Thu Jun 22 10:35:41 2000 auto_home
T r---rmcdrmcdr--- eagle.hurst.nis. Thu Jun 22 10:35:42 2000 bootparams
T r---rmcdrmcdr--- eagle.hurst.nis. Thu Jun 22 10:35:42 2000 cred
T r---rmcdrmcdr--- eagle.hurst.nis. Thu Jun 22 10:35:43 2000 ethers
T r---rmcdrmcdr--- eagle.hurst.nis. Thu Jun 22 10:35:43 2000 hosts
T r---rmcdrmcdr--- eagle.hurst.nis. Thu Jun 22 10:35:43 2000 ipnodes
T r---rmcdrmcdr--- eagle.hurst.nis. Thu Jun 22 10:35:44 2000 mail_aliases
T r---rmcdrmcdr--- eagle.hurst.nis. Thu Jun 22 10:35:44 2000 sendmailvars
T r---rmcdrmcdr--- eagle.hurst.nis. Thu Jun 22 10:35:45 2000 netmasks
T r---rmcdrmcdr--- eagle.hurst.nis. Thu Jun 22 10:35:45 2000 netgroup
T r---rmcdrmcdr--- eagle.hurst.nis. Thu Jun 22 10:35:45 2000 networks
T r---rmcdrmcdr--- eagle.hurst.nis. Thu Jun 22 10:35:46 2000 protocols
T r---rmcdrmcdr--- eagle.hurst.nis. Thu Jun 22 10:35:46 2000 rpc
T r---rmcdrmcdr--- eagle.hurst.nis. Thu Jun 22 10:35:47 2000 services
T r---rmcdrmcdr--- eagle.hurst.nis. Thu Jun 22 10:35:47 2000 timezone
T r---rmcdrmcdr--- eagle.hurst.nis. Thu Jun 22 10:35:47 2000 client_info
T r---rmcdrmcdr--- eagle.hurst.nis. Thu Jun 22 10:35:48 2000 auth_attr
T r---rmcdrmcdr--- eagle.hurst.nis. Thu Jun 22 10:35:48 2000 exec_attr
T r---rmcdrmcdr--- eagle.hurst.nis. Thu Jun 22 10:35:48 2000 prof_attr
T r---rmcdrmcdr--- eagle.hurst.nis. Thu Jun 22 10:35:49 2000 user_attr
T ----rmcdrmcd---- eagle.hurst.nis. Thu Jun 22 10:35:49 2000 audit_user
eagle #
```

```
\section{Konfiguration des Clients}

\section{Migration von NIS zu NIS+}
```

Kapitel 37

AUTOMOUNTER

Praktikum

1. a

Lösungsansätze

1. a

Kapitel 38

CACHE FILE SYSTEM

38.1 Allgemein

Das CacheFS ist ein Solaris eigenes Dateiensystem. Es kann nicht für reguläre Dateien benutzt werden. Ein CacheFS wird benutzt um auf einem NFS Client die Zugriffe auf ein NFS Server zwischenzuspeichern. Man kann das CacheFS jedoch auch für die Zugriffe auf CDROMs verwenden, heute jedoch ist das ehr ein Nachteil.

Allgemein kann man sagen : Ein CacheFS wird dann verwendet, wenn die Quelle der Daten gegenüber der lokalen Festplatte langsamer ist.

Im Bereich des NFS kann somit eine sehr hohe Performance rausgekitzelt werden und das Netzwerk wird extrem entlastet. Jedoch gibt es auch hier wieder mal eine Kehrseite :

Das CacheFS sollte in Verbindung mit NFS nur dann eingesetzt werden, wenn die Daten auf dem NFS entweder Readonly oder nur sehr selten geändert werden. Sprich für /var oder /home sehr ungeeignet.

Werden ständig veränderbare Daten via CacheFS zwischengebuffert, kommt es zu sehr hohen Lasten, weil der Client ständig die Daten vom CacheFS und die des NFS Servers abgleichen muß. Wenn mehrere User auf ein Share zugreifen kann die Last hier sehr hoch ausfallen. Insgesamt hoher als wenn man es ohne CacheFS machen würde.

38.1.1 Back und Front

Es gibt beim CacheFS zwei neue Begriffe. Zum einen das BackFS und zum zweiten das FrontFS.

BackFS Als BackFS wird das Dateiensystem bezeichnet, welches das CacheFS benutzt. Das kann entweder ein NFS oder HSFS (CDROM) sein. Sprich: Da kommt es her.

FrontFS Als FrontFS wird das Filesystem bezeichnet auf dem der Cache abgelegt wird. Meist ist es eine lokale Festplatte

38.1.2 Funktionsweise im Detail

*** PLEASE PLACE krapfick HERE ***

1. Das Betriebssystem oder der Benutzer führt das Programm `cat datei` aus. Wobei angenommen wird das `cat` nicht im CacheFS und datei im CacheFS ist.

2. Die Shell sucht als erstes die Datei `cat` und versucht diese Auszuführen. Sie findet `cat` in /usr/bin und sucht `/usr/bin/cat` auf dem CacheFS. Ist die Datei nicht auf dem CacheFS vorhanden, dann wird die Anfrage zum NFS Server weitergeleitet

3. Die Datei `cat` wird vom NFS Server gesucht und übertragen

4. Die übertragende Datei wird in das CacheFS eingetragen. Der Shell wird nun das Programm übermittels. Die Shell führt es nun aus.

5. Das Programm `cat` öffnet nun die Datei und das Betriebssystem sucht wieder `datei` im CacheFS. Da es im CacheFS liegt, findet nun die Übertragung aus dem CacheFS statt, ohne den NFS Server zu fragen

38.2 Einrichten eines CacheFS

Als erstes muß ein CacheFS gemountet werden. Dazu ist ein Mountpoint von nöten. Der Mountpoint jedoch kann liegen wo sie wollen. In den meisten Büchern findet man `/cache` oder `/.cache`. Der Mointpoint darf jedoch nicht angelegt werden, das macht `cfsadmin` automatisch.

Das Programm zur Einrichtung nennt sich `cfsadmin`[1]. Mit dem Programm können die Einstellungen angezeigt und geändert werden.

```
# cfsadmin -c /cacheme
#
```

Bildschirmausschnitt 38.2.1: Erstellen eines CacheFS

Mit der Option *-l* können die Daten angezeigt werden

```
# cfsadmin -l /cacheme
cfsadmin: list cache FS information
 maxblocks    90%
 minblocks     0%
 threshblocks 85%
 maxfiles     90%
 minfiles      0%
 threshfiles  85%
 maxfilesize   3MB
#
```

Bildschirmausschnitt 38.2.2: Anzeigen der CacheFS Einstellungen

Es sind alles die Defaulteinstellungen.

38.3 Benutzen eines CacheFS

Soll ein CacheFS mit einem BackFS benutzt werden, dann muß es mit dem Dateientyp *cachefs* mit `mount`[2] gemountet werden. Dem Programm `mount` muß man dann die speziellen Optionen zum BackFS als Optionen *-o* mit angeben.

Das folgende Beispiel mountet z.B. vom EAGLE das `/usr/X11R6` und benutzt das ebend eingerichtete CacheFS :

[1] `cfsadmin` - Siehe Kommandoreferenz Seite 255
[2] `mount` - Siehe Kommandoreferenz Seite **??**

```
raven # mount -F cachefs -o backfstype=nfs,cachedir=/cacheme,
        demandconst eagle:/usr/X11R6 /usr/X11R6
#
```

Bildschirmausschnitt 38.3.1: Erstellen eines CacheFS

Praktikum

1. a

Lösungsansätze

1. a

Kapitel 39

NETWORK TIME PROTOCOL

Praktikum

1. a

Lösungsansätze

1. a

Kapitel 40

AUTOINSTALLATIONSSYSTEM

40.1 Funktionsweise

40.1.1 Regelwerke

40.2 Konfiguration des Boot-Servers

40.3 Konfiguration des Install-Servers

40.4 Konfiguration des Regelwerk-Servers

40.5 Konfiguration des cfg-Servers

Praktikum

1. a

Lösungsansätze

1. a

TEIL V

ANHÄNGE UND REFERENZEN

Kapitel 41

KOMMANDOREFERENZ

arch

Aufruf

```
arch
```

Beschreibung

Das Programm `arch` zeigt einem die Architektur der verwendeten Hardward an. Dieses Programm eignet sich in Shellscripts dazu herauszufinden um welchen Rechnertyp es sich handelt.

Beispiele

```
crow: whurst $ arch
sun4
crow: whurst $ rsh tiger arch
i586
crow: whurst $
```

Bildschirmausschnitt 1: Beispiele zu `arch`

Exitcodes

Exitcodes von `arch`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

bg

Aufruf

```
bg [<jobid>]
```

Beschreibung

Das Programm bg bringt ein Job der Shell in den Hintergrund. Ist der Job in den Hintergrund gebracht worder, verhält er sich so, als wenn der Prozess via & in den Hintergrund gebracht würde.

Ein mit bg gestellten Hintergrundprozess wird beim Verlassen der Shell mit beendet.

Optionen

$< jobid >$ Angabe der Job-ID. Wird keine JobID angegeben dann wird der letzte Job in den Hintergrund gebracht.

Beispiele

```
$ sleep 1000
^Z
$ jobs
[1]+  Stopped                 sleep 1000
$ bg 1
[1]- sleep 1000 &
$ ps
   PID TTY      TIME CMD
   863 pts/7    0:00 sleep
   543 pts/7    0:00 sh
$ jobs
[1]-  Running                 sleep 1000 &
$
```

Bildschirmausschnitt 2: Beispiele zu bg

Exitstatus

Tabelle 41.2: Exitcodes von bg

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Siehe auch

jobs, fg

cat

Aufruf

```
cat
```

Beschreibung

Das Programm `cat`

Optionen

-v

Beispiele

```
cmd-cat-samples
```

Bildschirmausschnitt 3: Beispiele zu `cat`

Exitstatus

Exitcodes von `cat`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

cd

Aufruf

```
cd
```

Beschreibung

Das Programm `cd`

Optionen

-v

Beispiele

```
cmd-cd-samples
```

Bildschirmausschnitt 4: Beispiele zu `cd`

Exitstatus

Exitcodes von `cd`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

cfsadmin

Aufruf

```
cfsadmin [options] <cache-directory>
```

Beschreibung

Das Programm `cfsadmin` dient zur Einrichtung und Verwaltung eines CacheFS Verzeichnisses.

Optionen

-c Dient zur Erstellung von CacheFS Verzeichnissen. Werden keine weiteren Optionen mit *-o ...* angegeben gelten die Standardeinstellungen

-d $< cache-id >$ Mit der Option *-d* wird die Bufferung von Daten für die entsprechende ID zurückgenommen. Die $< cache-id >$ kann mittels *-l* festgestellt werden. Wird *all* angegeben werden alle Bufferungen aufgehoben. Achtung wird eine Bufferung gelöscht muß unbegingt `fsck` ausgeführt werden.

-l Gibt Ausgkung über ein CacheFS und zeigt auch die entsprechenden ID's an

-s Wird mit der Option *demandconst* gemountet, werden die Daten erst nach der manuellen Synchonisation abgeglichen. Diese Sync kann mit *-s* entsprechend gestartet werden.

-u Mit *-u* können die Einstellungen vom CacheFS geändert werden. Dazu ist die Option *-o* mit den entsprechenden Werten anzugeben. Wird ein Wert jedoch nicht angegeben wird dafür der Standardwert genommen

-o $< option >=< value >, < option >=< value > ...$ Mit *-o* können diverse Optionen mit angegeben werden. Werden mehrere Optionen angegeben werden diese durch Kommas getrennt angegeben. Folgende Optionen sind möglich :

$maxblocks =< zahl >$ Mit *maxblocks* wird die maximale Größe angegeben, die CacheFS auch dem Frontend (Festplatte des Clients) verwenden soll. Die Angabe wird in Prozent angegeben, relativ zur gesamt Frontendgröße. Standardwert ist 90%. Wird die Festplatte jedoch noch für andere Zwecke eingesetzt sollte man den Wert eventuell verkeinern.

$minblocks =< zahl >$ Gibt die minimale Größe relativ zur gesamtgröße des Frontends an. Standardwert ist 0%.

$threshblocks =< zahl >$ Ist der Wert erreicht und das CacheFS die Größe von *minblocks* überschritten hat, wird das Wachsen des Buffers verhindert. Standardeinstellung ist 85%

$maxfiles =< zahl >$ siehe *maxblocks* jedoch für Dateien (INodes). Standardwert hier sind 90%

$minfiles =< zahl >$ siehe *minblocks* jedoch für Dateien (INodes). Standardwert hier sind 0%

$threshfiles =< zahl >$ siehe *threshblocks* jedoch für Dateien (INodes). Standardwert hier sind 85%

$maxfilesize =< zahl >$ Angabe der größtmöglichsten Datei im CacheFS. Standardeinstellung ist 3MB.

Beispiele

```
# cfsadmin -c -o maxblocks=50,maxfilesize=10 /.cache
# cfsadmin -l /.cachefs
 maxblocks    50%
 minblocks     0%
 threshblocks 85%
 maxfiles     90%
 minfiles      0%
 threshfiles  85%
 maxfilesize  10MB
#
```

Bildschirmausschnitt 5: Beispiele zu `cfsadmin`

Exitstatus

Tabelle 41.5: Exitcodes von `cfsadmin`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

cachefspack, cachefslog, cachefsstat, cachefswssize, mount, fsck

chgrp

Aufruf

```
chgrp [optionen] gruppe filename...
```

Beschreibung

Das Programm `chgrp` verändert die Gruppenzugehörigkeit einer oder mehrerer Dateien. Als Gruppe kann entweder der Name oder die GruppenID angegeben werden

Optionen

-R Wird die Option `-R` angegeben, dann verändert `chgrp` die Gruppenangehörigkeit der Dateien in Unterverzeichnissen. (Recursiv)

Beispiele

```
cmd-chgrp-samples
$ ls -lR
.:
-rw-r--r--   1 whurst   whurst       651 Apr 30 23:34 backup.tex
drwxr-xr-x   2 whurst   whurst      1024 Apr 16 17:21 Figures

./Figures:
-rw-r--r--   1 whurst   whurst      4359 Apr 16 16:11 modeword-filetyp.fig
$ chgrp users *
$ ls -lR
.:
-rw-r--r--   1 whurst   users        651 Apr 30 23:34 backup.tex
drwxr-xr-x   2 whurst   users       1024 Apr 16 17:21 Figures

./Figures:
-rw-r--r--   1 whurst   whurst      4359 Apr 16 16:11 modeword-filetyp.fig
$ chgrp -R lager *
$ ls -lR
.:
-rw-r--r--   1 whurst   lager        651 Apr 30 23:34 backup.tex
drwxr-xr-x   2 whurst   lager       1024 Apr 16 17:21 Figures

./Figures:
-rw-r--r--   1 whurst   lager      4359 Apr 16 16:11 modeword-filetyp.fig
$
```

Bildschirmausschnitt 6: Beispiele zu `chgrp`

Exitstatus

Exitcodes von `chgrp`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

/etc/group

Siehe auch

chown, chmod

chmod

Aufruf

```
chmod
```

Beschreibung

Das Programm `chmod`

Optionen

-v

Beispiele

```
cmd-chmod-samples
```

Bildschirmausschnitt 7: Beispiele zu `chmod`

Exitstatus

Exitcodes von `chmod`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

chown

Aufruf

```
chown
```

Beschreibung

Das Programm chown

Optionen

-v

Beispiele

```
cmd-chown-samples
```

Bildschirmausschnitt 8: Beispiele zu chown

Exitstatus

Exitcodes von chown

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

cp

Aufruf

```
cp [optionen] <quelldatei> <zieldatei>
cp [optionen] <quelldatei> <zielverzeichnis>
cp [optionen] <quelldateien>... <zielverzeichnis>
```

Beschreibung

Das Programm `cp` kopiert Dateien. Ist das letzte Argument ein Verzeichnis dann werden die angegebenen Quelldateien mit dem Originalnamen in das Verzeichnis hinein kopiert. Existiert das Ziel nicht dann wird es angelegt.

Optionen

-p Die Dateien werden kopiert und die Zieldatei(en) bekommen die gleichen Rechte wie die Originaldateien. Es werden auch ACLs kopiert. Unterstützt das Zielsystem keine ACLs dann kommt es nicht zu einem Fehler ! In der Kopie wird immer das SUID und SGID Bit gelöscht. Der Eigentümer wird nur gesetzt wenn der Supervisor diese Operation ausführt.

-r Damit kann `cp` auch Verzeichnisstrukturen kopieren

-R Wie -r jedoch werden Pipe-Dateien dupliziert und nicht aus denen gelesen.

-i Wenn `cp` eine Zieldatei überschreiben soll, fragt es vorher den Benutzer

Beispiele

```
$ cp /etc/group mygroup
$ cp /etc/passwd .
$ ls
mygroup   passwd
$ cp -R /etc .
$ ls -l
Gesamt 64
drwxr-xr-x  41 whurst   whurst     11316 Jun  1 18:46 etc/
-rw-r--r--   1 whurst   whurst       290 Jun  1 18:45 mygroup
-r--r--r--   1 whurst   whurst       576 Jun  1 18:46 passwd
$
```

Bildschirmausschnitt 9: Beispiele zu `cp`

Exitstatus

Exitcodes von cp

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

crontab

Aufruf

```
crontab -{elr} [username]
```

Beschreibung

Das Programm `crontab` verwalten die crontabs von Benutzern. bzw. die eigenen. Nur ein Administrator kann die crontab's eines anderen Benutzers editieren.

Optionen

-e Mit der Option *-e* wird die crontab editiert, oder neuerstellt wenn es noch keine gab. Es wird dazu der Editor aufgerufen der in der Environmentvariable EDITOR vereinbart ist. Ist EDITOR nicht definiert dann wird der Editor `ex` verwendet.

-l Die crontab wird aufgelistet

-r Die crontab wird entfernt

Beispiele

```
$ crontab -l
.....
$
```

Bildschirmausschnitt 10: Beispiele zu `crontab`

Exitstatus

Tabelle 41.10: Exitcodes von `crontab`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

EDITOR

Dateien

Siehe auch

cron

df

Aufruf

```
df [optionen] [device | directory]
```

Beschreibung

Das Programm `df` zeigt die gemounteten Devices und deren Benutzung an. Die Ausgabe ist jenach eingesetzter Option unterschiedlich.

Optionen

-a Anzeige aller Dateiensysteme und Mountpoints

-b Zeigt nur das Device und die noch verfügbaren Kilobytes an.

-e Zeigt nur das Device und die noch freien INodes an

-F $< FSType >$ Zeigt nur die Devices an, die mit den entsprechenden $FSTyp$ gemountet wurden.

-g Zeigt erweiterte Informationen

-k Zeigt Informationen von gesamte, benutzte Kapazität und Device und Mountpoint an. (Die geeigneste Anzeige)

-l Zeigt nur lokale Datenträger an. Netzwerklaufwerke werden nicht angezeigt

-n Zeigt nur den Mointpoint und dessen verwendetes Dateiensystem an

-t Druckt eine erweiterte Liste

Beispiele

```
$ df -F nfs
/home     (cheetah:/home         ): 9687278 Blcke -1 Dateien
/var/mail (falcon:/var/spool/mail): 1489542 Blcke -1 Dateien
$ df -nlF ufs
/                  : ufs
/usr               : ufs
/var               : ufs
/opt               : ufs
/usr/local         : ufs
$ df -k /usr
Dateisystem            kByte  belegt verfgbar Kapazitt Eingehngt auf
/dev/dsk/c0t2d0s3    5159318  894662 4213063      18%    /usr
$
```

Bildschirmausschnitt 11: Beispiele zu `df`

Exitstatus

Exitcodes von `df`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

edquota

Aufruf

```
edquota
```

Beschreibung

Das Programm `edquota`

Optionen

-v

Beispiele

```
cmd-edquota-samples
```

Bildschirmausschnitt 12: Beispiele zu `edquota`

Exitstatus

Exitcodes von `edquota`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

export

Aufruf

```
export
```

Beschreibung

Das Programm `export`

Optionen

-v

Beispiele

```
cmd-export-samples
```

Bildschirmausschnitt 13: Beispiele zu `export`

Exitstatus

Exitcodes von `export`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

fg

Aufruf

```
fg
```

Beschreibung

Das Programm `fg`

Optionen

-v

Beispiele

```
cmd-fg-samples
```

Bildschirmausschnitt 14: Beispiele zu `fg`

Exitstatus

Tabelle 41.14: Exitcodes von `fg`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

file

Aufruf

```
file
```

Beschreibung

Das Programm `file`

Optionen

-v

Beispiele

```
cmd-file-samples
```

Bildschirmausschnitt 15: Beispiele zu `file`

Exitstatus

Exitcodes von `file`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

find

Aufruf

```
find
```

Beschreibung

Das Programm `find`

Optionen

-v

Beispiele

```
cmd-find-samples
```

Bildschirmausschnitt 16: Beispiele zu `find`

Exitstatus

Tabelle 41.16: Exitcodes von `find`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

getfacl

Aufruf

```
getfacl
```

Beschreibung

Das Programm `getfacl`

Optionen

-v

Beispiele

```
cmd-getfacl-samples
```

Bildschirmausschnitt 17: Beispiele zu `getfacl`

Exitstatus

Exitcodes von `getfacl`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

grep

Aufruf

```
grep
```

Beschreibung

Das Programm `grep`

Optionen

-v

Beispiele

```
cmd-grep-samples
```

Bildschirmausschnitt 18: Beispiele zu `grep`

Exitstatus

Tabelle 41.18: Exitcodes von `grep`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

groups

Aufruf

```
groups
```

Beschreibung

Das Programm `groups`

Optionen

-v

Beispiele

```
cmd-groups-samples
```

Bildschirmausschnitt 19: Beispiele zu `groups`

Exitstatus

Exitcodes von `groups`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

halt

Aufruf

```
halt
```

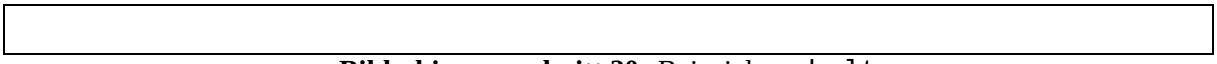
Beschreibung

Das Programm `halt`

Optionen

-v

Beispiele



Bildschirmausschnitt 20: Beispiele zu `halt`

Exitstatus

Tabelle 41.20: Exitcodes von `halt`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

head

Aufruf

```
head [-<Anzahl Zeilen>] [Dateinamen]...
```

Beschreibung

Das Programm `head` gibt die ersten 10 Zeilen der Datei bzw. von `stdin` aus. Wird jedoch eine Anzahl angegeben, werden die ersten $< AnzahlZeilen >$ ausgegeben.

Optionen

-$< AnzahlZeilen >$ Die Anzahl der auszugebenen Zeilen

Beispiele

```
$ head /etc/passwd
...
$ grep ":/home" /etc/passwd | head -20
...
$
```

Bildschirmausschnitt 21: Beispiele zu `head`

Exitstatus

Exitcodes von `head`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

tail, more, cat,

hostname

Aufruf

```
Solaris: hostname [name-of-host]
Linux: hostname [-d|-f|-s|-a|-i|-y|-n]
```

Beschreibung

Das Programm `hostname` zeigt den Hostnamen des Hosts an. Wird jedoch ein Hostname als Parameter angegeben wird dieser Hostname gesetzt. Das jedoch darf nur der Superuser.

Der Hostname wird normalerweise beim Systemstart gesetzt. Er kann jedoch wärend der Laufzeit des Systems geändert werden ohne das man das System neu starten muß.

Optionen

Die Version von Solaris erlaubt dem normalen Benutzern nur die Verwendung von Hostname ohne Parameter. Die Linuxversion unterstützt jedoch einige mehr.

-d Gibt Auskunft über die verwendete DNS Domain

-f Gibt den kompletten Hostnamen mit Domainnamen bekannt. (FQDN)

-s Gibt nur den kurzen Hostnamen aus

-i Gibt Auskunft über die verwendete IP-Adresse

-y Gibt den NIS Domainnamen in dem sich der Host befindet bekannt

Beispiele

```
cmd-hostname-samples
tiger: whurst $ hostname
tiger
tiger: whurst $ hostname -f
tiger.hurst.pnet
tiger: whurst $ hostname -i
10.65.13.7
tiger: whurst $
```

Bildschirmausschnitt 22: Beispiele zu `hostname`

Exitstatus

Exitcodes von `hostname`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Siehe auch

uname

id

Aufruf

```
id
```

Beschreibung

Das Programm `id`

Optionen

-v

Beispiele

```
cmd-id-samples
```

Bildschirmausschnitt 23: Beispiele zu `id`

Exitstatus

Exitcodes von `id`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

infocmp

Aufruf

```
infocmp
```

Beschreibung

Das Programm `infocmp`

Optionen

-v

Beispiele

```
cmd-infocmp-samples
```

Bildschirmausschnitt 24: Beispiele zu `infocmp`

Exitstatus

Exitcodes von `infocmp`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

init

Aufruf

```
init
```

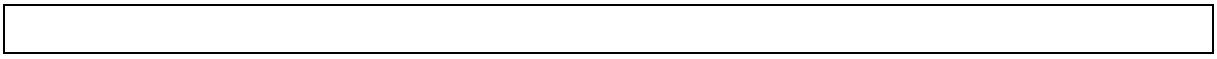
Beschreibung

Das Programm `init`

Optionen

-v

Beispiele



Bildschirmausschnitt 25: Beispiele zu `init`

Exitstatus

Tabelle 41.25: Exitcodes von `init`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

installboot

Aufruf

```
installboot
```

Beschreibung

Das Programm `installboot`

Optionen

-v

Beispiele

```
cmd-installboot-samples
```

Bildschirmausschnitt 26: Beispiele zu `installboot`

Exitstatus

Tabelle 41.26: Exitcodes von `installboot`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

jobs

Aufruf

```
jobs
```

Beschreibung

Das Programm `jobs`

Optionen

-v

Beispiele

```
cmd-jobs-samples
```

Bildschirmausschnitt 27: Beispiele zu `jobs`

Exitstatus

Tabelle 41.27: Exitcodes von `jobs`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

kill

Aufruf

```
kill
```

Beschreibung

Das Programm `kill`

Optionen

-v

Beispiele

```
cmd-kill-samples
```

Bildschirmausschnitt 28: Beispiele zu `kill`

Exitstatus

Exitcodes von `kill`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

ln

Aufruf

```
ln
```

Beschreibung

Das Programm `ln`

Optionen

-v

Beispiele

```
cmd-ln-samples
```

Bildschirmausschnitt 29: Beispiele zu `ln`

Exitstatus

Tabelle 41.29: Exitcodes von `ln`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

logname

Aufruf

```
logname
```

Beschreibung

Das Programm `logname` gibt nur den Loginnamen des Benutzers aus. Das Programm in in Scripts und in Programmen sehr wertvoll. Auf der Kommandozeile jedoch ist es etwas sinnlos.

Optionen

Keine

Beispiele

```
cmd-logname-samples
crow: whurst $ logname
whurst
crow: whurst $
```

Bildschirmausschnitt 30: Beispiele zu `logname`

Quelltext 41.0.1 Beispiel von `logname` in Shellscripts

```
#!/bin/sh

echo "Hello ‘logname‘"

if test "‘logname‘" != "otto"; then
echo "Hallo ‘logname‘, du bist nicht otto"
fi
```

Exitstatus

Exitcodes von `logname`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Siehe auch

env(1), login(1), getlogin(3C), utmp(4), attributes(5), environ(5)

lp

Aufruf

```
lp [optionen] [Dateien...]
```

Beschreibung

Das Programm `lp` überträgt einen Datenstrom in das Spoolverzeichnis eines Druckers. Und informiert den Druckserver über die neuen Druckaufträge.

Optionen

-c Mit der Option *-c* wird `lp` dazu veranlasst die zu druckende Datei zuerst zu kopieren (meist nach `/var/tmp`) und dann erst den Druckauftrag aufzugeben. Wenn `lp` am Ende einer Pipe steht oder bei anderen Programmen kann dieses extrem helfen ! (Das Programm `gv` benötigt diese Einstellungen dringend)

-d $< printer >$ Angabe des Druckernamens oder der Druckerklasse zu dem der Druckauftrag gesendet werden soll. Auch Netzwerkbezeichner in der Form von $< server >:< printer >$ sind möglich. Aber auch FNS bezeichner sind erlaubt wzb. `.../service/printer`
`lp -d prnserver:hplaserjet /etc/irgendwas` z.B.
Wird die Option nicht angegeben wird der Standarddrucker verwendet.

-f $< form >$ Benutzt zur Ausgabe ein bestimmtes Formular.

-H $< special >$ Mit *-H* kann ein spezielles Verhalten für diesen Druckauftrag eingestellt werden. Als Bezeichner kommen die folgenden in Frage :

hold Der Druckauftrag wird bis zu einer Bestätigung nicht gedruckt.

resume Startet ein auf *hold* gesetzten Druckauftrag

immediate Schiebt den Druckauftrag an den Anfang der Druckerwarteschlage. Dieses Special ist nur für LP Administratoren anwendbar.

-m Sendet dem Auftraggeber eine EMail, wenn der Druckauftrag beendet wurde

-n $< number >$ Gibt die Anzahl der zu druckenden Kopien an. Der Vorteil bei Umwandlungen von z.B. ASCII in Postscript ist, daß der Druckauftrag nur einmal Konvertiert wird und nicht etwa 10 mal.

-o $< option >$ Definiert zusätzliche Optionen zu diesem Druckauftrag. Diese Optionen werden mit den Optionen die mittels `lpadmin` konfiguriert wurden vermischt. Mögliche Optionen wären :

nobanner Druckt kein Banner aus

nofilebreak Druckt mehrere Dateien ohne ein Seitenumbruch dazwischen

length=$< number >$ Angabe der Zeilenanzahl der Seite. Es können auch Inch oder Zentimeterangegeben werden. Dann muß jedoch entweder ein `i` oder `c` an die Zahl angehängt werden.
`lp -o length=11i`

width=$< number >$ Das gleiche wie *length* nur jetzt für die Seitenbreite.

lpi=$< number >$ Zeilen pro Inch. Angabe der Zeilen pro Inch. Meist sind es 6 oder 8.

cpi=n—pica—elite—compressed Jenach Wahl wird mit der entsprechenden cpi[1] gedruckt. Die Palette reicht von 10cpi bis 20 cpi.

stty=$< stty - options >$ Schnittstellenparameter zur Initialisierung der Schnittstelle mittels `stty`[2]

Die Optionen dürfen durch Kommas getrennt angegeben werden :
`lp -o cpi=pica,lpi=8,length=66 /etc/jaja`

-P $< pagelist >$ Wenn das Interface des Druckers in der Lage ist einzelne Seiten auszudrucken dann können die hier angegeben werden. Ist das Interface dazu nicht in der Lage wird der Druckauftrag *rejected*

-p Aktiviert die Benachrichtigung nach einem Drckauftrag. Wie das jedoch passiert liegt am Interface. Das kann via EMail oder `write` passieren.

-q $< prior >$ Mit der Option *-q* kann die numerische Priorität eines Druckauftrages festgelegt werden. Die Null (0) bezeichnet dabei die höchste Priorität aus und kommt mit *-H immediate* gleich. Und 39 (neununddreizich) ist die kleinste Priorität.

-s Unterdrückt die Nachricht die `lp` normalerweise an den Benutzer sendet, wenn er ein Druckauftrag aufgibt.

-S $< wheel >$ Wird ein Typenraddrucker eingesetzt mit der Möglichkeit mehrere Typenräder zu benutzen. Dann kann das gewünschte Typenrad mit *-S* angegeben werden.

-t $< title >$ Druckt den angegebenen Text als Titel aus. Man sollte diesen in Double-Quotes einschliessen

-T $< content >$ Sucht ein Drucker der den angegebenen Content ausdrucken kann. Gibt es kein Drucker, dann wird versucht die Datei entsprechend zu wandeln. Wird zusätzlich die Option *-r* angegeben wird kein Converter gesucht und der Druckjob wird *rejected*

-w Schreibt eine Nachricht auf das Terminal wenn der Druckauftrag beendet wurde. Ist der Benutzer nicht mehr eingeloggt kommt eine EMail.

-y $< mode >$ Angabe des Druckmodus. Jaja, keiner weiss was genaues.

Beispiele

```
$ lp -d eptxt -o cpi=compressed,lpi=8 -w -s /daten/text
```

Bildschirmausschnitt 31: Beispiele zu `lp`

Exitstatus

[1] `cpi`-Character Per Inches
[2] `stty` - Siehe Kommandoreferenz Seite 391

Exitcodes von `lp`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

`LPDEST` - Defaultprinter oder auch `PRINTER` ...

Dateien

/var/spool/* - Spooldateien
$HOME/.printers - Standard Drucker des Users
/etc/printer.conf - Systemweite Drucker
printers.conf.byname - NIS Map
fns.context.domain - NIS+ Map

Siehe auch

lpstat, lpadmin

lpadmin

Aufruf

```
lpadmin optionen...
```

Beschreibung

Das Programm `lpadmin` dient zur Einrichtung von Druckern in einer SystemV Umgebung. Alle relevanten Einstellungen zu einem Drucker werden mit `lpadmin` gemacht. Man kann entweder alle Optionen in einem Befehl absetzen oder es einzelnd tun.

Optionen

-p $< druckername >$ Mit dieser Option muß der logische Druckername angegeben werden. Damit `lpadmin` auch weiss welchen Drucker Sie meinen.

-A $< key >$ **-W** $< zahl >$ Mit der Option *-A* wird angegeben wie der Druckerdienst den Systemadministrator über Fehler im Drucksystem informieren soll. Es stehen dazu einige Keys zurverfügung :

mail Der Administrator bekommt eine Mail. Optional kann der Administrator dort ein Usernamen angeben an den die Mail gesendet werden soll. z.B:
`lpadmin -A 'mail otto' ...`

write Der Administrator bekommt eine Meldung auf sein Terminal geschrieben.

quite Eine Wiederholung der Nachrichten wird unterdrückt

none Nix wird gemacht. Jeder steht dann jedoch voll im Dunkel

$< programname >$ Es kann auch ein Programm angegeben werden, welches ausgeführt wird wenn es zu einem Fehler kommt. In diesem Programm kann man dann machen was man will

list Die Benachrichtigungen werden aufgelistet

Mit der Option-*W* wird die Anzahl der Minuten angegeben die zur nächsten Benachrichtigung führt. Voreingestellt ist die 0 (NULL) was zu einer einmaligen Benachrichtigung führt.

-c $< klasse >$ Mit der Option *-c* wird der Klassenname des Druckers angegeben

-r $< klasse >$ Der angegebene Drucker wird aus der angegebenen Klasse wirder entfernt

-D $< kommantar >$ Ein Kommentar für diesen Drucker kann man mittels *-D* einstellen. Wichtig ist jedoch das der Kommentar in Anführungszeichen steht, sofern dieser Leerzeichen enthält

-d $< druckername >$ Definiert den angegebenen Drucker zum Standarddrucker

-e $< druckername >$ Kopiert das Interfaceprogramm des angegebenen Druckers

-F $< key >$ Gibt das Ferhalten des Druckerdienstes im Fehlerfall an. Als Key können folgende Einstellungen verwendet werden :

continue Der Druckauftrag wird dort weiter geführt, wo er zum Abbruch kam

beginning Der Druckauftrag wird komplett wiederholt

wait Der Druckauftrag wird erst nach der Aufforderung des Administrators neu gestartet

-I $< dateityp >$ Mit der Option *-I* wird dem Drucksystem mitgeteilt welche Dateitypen dieser Drucker zu verarbeiten mag. Als Dateityp kommen folgende in Frage :

postscript Drucker kann Postscript Dateien verarbeiten

simple Drucker kann nur ASCII-Text Dateien drucken

any Drucker kann alles

Die Angaben können durch kommas getrennt aufgelistet werden. z.B.
`lpadmin -I postscript,simple ....`

-i $< interface >$ Wird ein nicht standard Interface für ein Drucker benutzt, damm muß man das Interface angeben. Das Interface wird danach automatisch kopiert.

-T $< druckertyp >$ Wird das Standarddrucksystem verendet muß man hier den Druckertyp eintragen. Der Druckertyp ist aus der *terminfo* Datenbank zu entnehmen. Für Postscript Drucker kann PS oder für HP PCL Drucker kann hplaser verwendet werden. Wird ein eigenes Interface benutzt sollte man unknown angeben

-u $< rights >:< userlist >$ Mit der Option *-u* werden Berechtigungen für Benutzer angegeben die diesen Drucker Benutzen dürfen. Es gibt zum einen eine Liste die dürfen und eine Liste die nicht dürfen. Man sollte sich nur für eine Entscheiden, weil sonst kommt es zu erheblichen Problemen. Als $< rights >$ können folgendE schlüssel dienen :

allow Liste der Benutzer die Drucken dürfen. Die werden in /etc/lp/printers/.../users.allow abgelegt.

deny Liste der Benutzer die nicht Drucken dürfen. Die werden in /etc/lp/printers/.../users.deny abgelegt.

Die Benutzer werden einfach durch kommas getrennt aufgelistet. Zum Beispiel :
`lpadmin -u allow:root,otto,karl,hostname!peter ...` erlaubt den Benutzern root, otto und karl das Drucken. Sowie auch den Benutzer Peter vom Druckclient HOSTNAME.

-o $< option >$ Mit *-o* können Optionen angegeben werden. Die vom Druckinterface ausgewertet werden. Als Optionen können folgende Optionen optional verwendet werden :

timeout=$< sekunden >$ Wird bei Netzwerkdruckern verwendet um die Wartezeit in Sekunden anzugeben bevor es zum Fehler kommt. Standardeinstellungen sind 10 sekunden.

protocol=$< typ >$ Angabe des Protokolls wie der Druckclient sich an den Druckserver meldet. Voreinstellung ist *bsd*. Es kann aber auch *raw* verwendet werden.

dest=$< ziel >$ Angabe des Zieldruckers im Netzwerk. Entweder IP oder Hostnamen

nobanner Ausgabe des Banners wird unterdrückt

banner Ausgabe des Banners

-s ... Obsolete. Setup network printer ...

-v $< device >$ Angabe der Gerätedatei des Druckers

-x $< druckername >$ Löscht ein Drucker komplett aus der Konfiguration raus.

Beispiele

```
cmd-lpadmin-samples
# lpadmin -p std -d std -v /dev/ecpp0 -T hplaser -I postscript
  -o nobanner -A "mail otto" -W 5 -u deny:otto,karl
#
```

Bildschirmausschnitt 32: Beispiele zu `lpadmin`

Exitstatus

Exitcodes von `lpadmin`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

/var/spool/lp/* Spooldateien

/etc/lp Konfigurationsdateien

/etc/lp/alert/printer Default Fehlermeldungs Manager

Siehe auch

enable, disable, accept, reject, lpstat, lpforms, lpsched, lp, /etc/init.d/lp

lpsched

Aufruf

```
lpsched
```

Beschreibung

Das Programm `lpsched`

Optionen

-v

Beispiele

```
cmd-lpsched-samples
```

Bildschirmausschnitt 33: Beispiele zu `lpsched`

Exitstatus

Exitcodes von `lpsched`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

lpstat

Aufruf

```
lpstat [optionen]
```

Beschreibung

Das Programm `lpstat` ist das Tool überhaupt für den Druckeradministrator. Mit `lpstat` können alle möglichen Staten der Druckerkonfiguration ausgegeben werden.

Mit $< list >$ ist eine Liste von Druckernamen oder Klassennamen gemeint und ist überall Optional

Optionen

-d Gibt den Default Drucker aus

-o $< list >$ Gibt den Ausgabe Status aus. Werden keine Drucker oder Klassen angegeben, dann werden alle ausgegeben.

-r Gibt Informationen über den Scheduler aus ob dieser aktiv ist oder nicht.

-R Gibt die Nummer der Druckerauftrage in dessen Druckerwarteschlange aus.

-s Druckt eine Übersicht der Drucker aus

-t Druckt alle Statusmeldungen aller Drucker aus

-u $< login - id - list >$ Druckt eine Liste aller noch zu druckenden Druckaufträge von speziellen Benutzern an. Wobei $< login - list >$ eine normale Liste mit User-Namen sein kann oder direkt mit Systemname. Folgende Kombinationen sind möglich :

$< login >$ Ein Benutzer vom System

$< system >! < login >$ Ein Benutzer eines speziellen Systems

$< system >! all$ Alle Benutzer eines speziellen Systems

$all! < login >$ Ein Benutzer von allen Systemen

z.B. `lpstat -u whurst,eagle\!dozent`

Man muß bei der Verwendung der `/bin/bash` das Ausrufezeichens flüchten !

-v $< list >$ Gibt die Drucker mit dessen Devices aus

-a $< list >$ Gibt den Status (accept/reject) aus

-c $< list >$ Gibt die Drucker und Klassennamen aus

-f $< list >$ Druckt eine Übersicht der Forms

-p $< list >$ Druckt noch eine Übersicht

-S $< list >$ Druckt eine Übersicht der eingestellten Typenräder

Beispiele

```
raven:whurst $ lpstat
raven:whurst $ lpstat -d
system default destination: hpmonops
raven:whurst $ lpstat -v
system for _default: raven (as printer hpmonops)
device for hpmonops: /dev/ecpp0
device for hpmonotext: /dev/ecpp0
raven:whurst $ lpstat -s
scheduler is running
system default destination: hpmonops
system for _default: raven (as printer hpmonops)
device for hpmonops: /dev/ecpp0
device for hpmonotext: /dev/ecpp0
raven:whurst $
```

Bildschirmausschnitt 34: Beispiele zu `lpstat`

Es ist wohl alles selbsterklärent

Exitstatus

Exitcodes von `lpstat`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

lp, lpsched

ls

Aufruf

```
ls
```

Beschreibung

Das Programm `ls`

Optionen

-v

Beispiele

```
cmd-ls-samples
```

Bildschirmausschnitt 35: Beispiele zu `ls`

Exitstatus

Tabelle 41.35: Exitcodes von `ls`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

mail

Aufruf

```
mail
```

Beschreibung

Das Programm `mail`

Optionen

-v

Beispiele

```
cmd-mail-samples
```

Bildschirmausschnitt 36: Beispiele zu `mail`

Exitstatus

Exitcodes von `mail`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

man

Aufruf

```
man
```

Beschreibung

Das Programm `man`

Optionen

-v

Beispiele

```
cmd-man-samples
```

Bildschirmausschnitt 37: Beispiele zu `man`

Exitstatus

Exitcodes von `man`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

- Kommandoreferenz

mkdir

Aufruf

```
mkdir
```

Beschreibung

Das Programm `mkdir`

Optionen

-v

Beispiele

```
cmd-mkdir-samples
```

Bildschirmausschnitt 38: Beispiele zu `mkdir`

Exitstatus

Exitcodes von `mkdir`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

more

Aufruf

```
more
```

Beschreibung

Das Programm `more`

Optionen

-v

Beispiele

```
cmd-more-samples
```

Bildschirmausschnitt 39: Beispiele zu `more`

Exitstatus

Exitcodes von `more`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

mv

Aufruf

```
mv
```

Beschreibung

Das Programm mv

Optionen

-v

Beispiele

```
cmd-mv-samples
```

Bildschirmausschnitt 40: Beispiele zu mv

Exitstatus

Exitcodes von mv

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

netstat

Aufruf

```
netstat [options]
```

Beschreibung

Das Programm `netstat` liefert Informationen über den Netzwerkstatus. Es hat ca. 1 Millionen mögliche Optionen.

Optionen

OHNE OPTIONEN Ohne die Angabe von Optionen zeigt `netstat` alle Sockets an die zur Zeit in Benutzung sind. In Kombination mit *-a* werden alle Socket angezeigt.

```
$ netstat

TCP: IPv4
Local Address Remote Address Swind Send-Q Rwind Recv-Q State
------------- -------------- ----- ------ ----- ------ -------
raven.1023    cheetah.nfsd   64240      0 24820    100 ESTABLISHED
raven.1022    falcon.nfsd    64240      0 24820      0 ESTABLISHED
raven.32815   raven.32774    32768      0 32768      0 ESTABLISHED
raven.32774   raven.32815    32768      0 32768      0 ESTABLISHED
raven.32909   raven.32808    32768      0 32768      0 TIME_WAIT

Active UNIX domain sockets
Address  Type          Vnode       Conn      Local Addr       Remote Addr
30000dc5ba8 stream-ord 3000094c1b8 00000000 /tmp/.X11-unix/X0
30000dc5d48 stream-ord 00000000    00000000
$
```

Bildschirmausschnitt 41.0.1: Beispiel: `netstat`

Local Address Angabe der lokalen IP Addresse oder des Namen mit Portnummer (Portnummer wird leider durch ein Punkt getrennt)

Remote Address Angabe des Hostnamens oder der IP Addresse mit Portnummer zu wem die Verbindung besteht

Swind Grösse des TCP-Window beim Senden

Send-Q Anzahl der Bytes die noch in der Sendequeue liegen

Rwind Größe des TCP-Window beim Empfang

Recv-Q Anzahl der noch nicht abgearbeiteten Bytes

State Gibt den Tatus der Verbindung an. Der Status kann folgendes sein :

BOUND Socket ist bereit zur Benutzung. Er wurde mit der Anwendung gebunden.

CLOSED Dieser Socket wurde geschlossen und wird nicht länger benutzt

CLOSING Socket wird geschlossen. Es wird noch auf die Bestätigung des Remotehosts gewartet

CLOSE_WAIT Remotehost hat den Socket geschlossen. Die Anwendunng muß den Socket noch schliessen

ESTABLISHED Die verbindung besteht und wird auch benutzt

FIN_WAIT_1 Der Socket wurde geschlossen. Er muß nur noch das *FIN* senden

FIN_WAIT_2 Das *FIN* wurde gesendet und man wartet auf die Bestätigung des Remotehosts

IDLE Socket ist verfügbar jedoch nicht gebunden

LAST_ACK Wartet auf die letzte Bestätigung nach dem Schliessen des Sockets

LISTEN Wartet auf ankommende Verbindungen

SYN_RECEIVED *SYN* ist unterwegs

SYN_SENT Remotehost hat *SYN* gesendet

TIME_WAIT Erwartet nach dem Schliessen die Bestätigung vom Remotehost

-g Zeigt die eingestellten Multicastaddressen aller Interfaces an.

```
$ netstat -g

Group Memberships: IPv4
Interface Group                RefCnt
--------- -------------------- ------
lo0       224.0.0.1                 1
hme0      224.0.0.1                 1
$
```

Bildschirmausschnitt 41.0.2: Beispiel: `netstat -g`

Man erkennt dass die Multicastaddresse 224.0.0.1 auf den Interfaces lo0 (loopback) und hme (100Mb/s) zur Verfuegung steht. Die Angabe **RefCnt** gibt die Anzahl der z.Zt benutzen Verbindungen an.

-i Zeigt die Interfaces mit den eingestellten IP Addressen und einer Statistik an. Man kann die Option *-a* verwenden um sich alle anzeigen zu lassen.

```
$ netstat -i
Name    Mtu  Net/Dest   Address   Ipkts Ierrs Opkts Oerrs Collis Queue
lo0     8232 localhost  localhost   542     0   542     0     0      0
hme0    1500 hurstnet   raven     10043     0  9924     0     0      0
ipdptp0 8232 0.0.0.0    0.0.0.0       0     0     0     0     0      0
$
```

Bildschirmausschnitt 41.0.3: Beispiel: `netstat -i`

Name Name des Interfaces

MTU Maximum Transfer Unit pro Datenpaket

Net/Dest Die lokale Netzwerkaddresse

Address Die IP Addresse des Interfaces

Ipkts Anzahl der empfangenden Pakete

Ierrs Anzahl der fehlerhaften empfangenden Pakete

Opkts Anzahl der gesendeten Pakete

Oerrs Anzahl der fehlerhaft gesendeten Pakete

Collis Anzahl der Collisionen

Queue Anzahl der Pakete in der Übertragungswarteschlange

-p Anzeige des ARP-Caches.

```
$ netstat -p

Net to Media Table: IPv4
Device   IP Address               Mask      Flags   Phys Addr
------ -------------------- --------------- ----- ---------------
hme0   tiger.hurst.pnet     255.255.255.255       00:00:b4:a6:00:86
hme0   falcon.hurst.pnet    255.255.255.255       00:00:b4:a6:00:2f
hme0   cheetah.hurst.pnet   255.255.255.255       00:00:b4:a6:7c:ee
hme0   puma.hurst.pnet      255.255.255.255       00:00:b4:a6:00:9b
hme0   raven.hurst.pnet     255.255.255.255 SP    08:00:20:b1:17:5e
hme0   224.0.0.0            240.0.0.0       SM    01:00:5e:00:00:00
$
```

Bildschirmausschnitt 41.0.4: Beispiel: `netstat -p`

Sollte die Ausgabe hier hängen bleiben muß die zusätzlich die Option *-n* verwendet werden um die Namensauflösung zu umgehen.

Device Name des Interfaces wo der folgende Rechner mit der MAC Addresse zu erreichen ist

IP Address Der Name oder die IP Addresse des Zielrechners

Mask Die Maske des Zieles. Bei Hosts ist es immer eine /32. Bei Multicastaddressen ist es immer eine /4

Flags Die Flags geben Auskunft uber den Status eines Eintrages. Dabei haben die folgenden Buchstaben eine besondere Bedeutung :

S Statischer Eintrag der mit `arp`[3] gesetzt wurde

P Publish. Diese MAC Addresse liefert der TCP/IP Stack bei Anfragen über das Netzwerk dem anfragenden Host. Im Normalfall hat jedes Interface immer nur eine Publishing Address

U Unresolved. Dieses Flag kennzeichnet ein noch nicht aufgelöste MAC Addresse. Das kommt immer dann vor wenn das System auf den ARP-Reply wartet

[3] `arp` - Siehe Kommandoreferenz Seite **??**

M Mapping. Für Multicastaddressen wird ein Spezielles Mapping zu einer speziellen MAC Addresse erforderlich

Phy. Addr Die MAC Addresse des Hosts

-r Zeigt die Routingtabelle des Kernels an. Wird die Option *-a* zusätzlich verwendet werden auch die Routen angezeigt die normalerweise ehr im Hintergrund Ihre Arbeit tun, wzB: Routen für den Broadcast und dynamische Hostrouten und so weiter :

```
$ netstat -ra

Routing Table: IPv4
  Destination           Gateway           Flags  Ref   Use   Interface
-------------------- -------------------- ----- ----- ------ ---------
network.hurst.pnet   raven.hurst.pnet      U         1     10  hme0
224.0.0.0            raven.hurst.pnet      U         1      0  hme0
default              falcon.hurst.pnet     UG        1      0
255.255.255.255      raven.hurst.pnet      UHB       1      0  hme0
255.255.255.255      raven.hurst.pnet      UHB       1      0  hme0
0.0.0.0              raven.hurst.pnet      UHB       1      0  hme0
0.0.0.0              raven.hurst.pnet      UHB       1      0  hme0
10.0.0.0             raven.hurst.pnet      UHB       1      0  hme0
10.0.0.0             raven.hurst.pnet      UHB       1      0  hme0
tiger.hurst.pnet         --                UHA       1      1  hme0
network.hurst.pnet   raven.hurst.pnet      UHB       1      0  hme0
network.hurst.pnet   raven.hurst.pnet      UHB       1      0  hme0
falcon.hurst.pnet        --                UHA       2    126  hme0
raven.hurst.pnet         --                UHL       6  11082  hme0
cheetah.hurst.pnet       --                UHA       2    488  hme0
localhost            localhost             UH       19    211  lo0
puma.hurst.pnet          --                UHA       3     53  hme0
broadcast.hurst.pnet raven.hurst.pnet      UHB       1    104  hme0
broadcast.hurst.pnet raven.hurst.pnet      UHB       1      0  hme0
10.255.255.255       raven.hurst.pnet      UHB       1      0  hme0
10.255.255.255       raven.hurst.pnet      UHB       1      0  hme0
$
```

Bildschirmausschnitt 41.0.5: Beispiel: `netstat -r`

Destination Das Zielnetzwerk oder der Zielhost

Gateway Das Gateway über den das Ziel erreicht wird. Entweder ist es ein Router oder das lokale Interface wenn das Ziel im lokalem Netzwerk sich befindet

Flags Status des Routingeintrages. Dabei haben die Buchstaben folgende Bedeutung :

U Up. Diese Route ist Aktiv

G Gateway. Dieses Ziel ist nur durch ein Gateway erreichbar und kommt nicht bei lokalen Zielen vor

H Host. Diese Route kennzeichnet eine Hostroute

B Broadcast. Diese Route kennzeichnet eine Route für die Benutzung einer Broadcastmessage

A Addressresolution. Eine Dynamischeroute durch eine Addressauflösung wie ARP

L Local. Eine Lokaleroute zum Interface

D Dynamic. Eine dynamische Route durch ein Routingprotokoll wie RIP.

Ref Anzahl der zur Zeit benutzen Verbindungen zu diesem Routingeintrag

Use Anzahl der bisher dort übertragenden Pakete

Interface Das Interface an dem die Route angehängt wird

-s Statistik der Protokolle

```
$ netstat -s
RAWIP
        rawipInDatagrams    =     1 rawipInErrors       =     0
        rawipInCksumErrs    =     0 rawipOutDatagrams   =     1
        rawipOutErrors      =     0
```

Bildschirmausschnitt 41.0.6: Beispiel: `netstat -s` Teil 1

```
UDP
        udpInDatagrams      =   762 udpInErrors         =     0
        udpOutDatagrams     =   769 udpOutErrors        =     0
```

Bildschirmausschnitt 41.0.7: Beispiel: `netstat -s` Teil 2

```
TCP
        tcpRtoAlgorithm     =     4 tcpRtoMin           =   400
        tcpRtoMax           = 60000 tcpMaxConn          =    -1
        tcpActiveOpens      =    87 tcpPassiveOpens     =    55
        tcpAttemptFails     =     1 tcpEstabResets      =     1
        tcpCurrEstab        =    22 tcpOutSegs          = 12461
        tcpOutDataSegs      =  9785 tcpOutDataBytes     =5693794
        tcpRetransSegs      =     2 tcpRetransBytes     =     0
        tcpOutAck           =  2672 tcpOutAckDelayed    =  1125
        tcpOutUrg           =     0 tcpOutWinUpdate     =     0
        tcpOutWinProbe      =     0 tcpOutControl       =   270
        tcpOutRsts          =     8 tcpOutFastRetrans   =     0
        tcpInSegs           = 12131
        tcpInAckSegs        =  7850 tcpInAckBytes       =5692940
        tcpInDupAck         =   173 tcpInAckUnsent      =     0
        tcpInInorderSegs    = 10032 tcpInInorderBytes   =5775708
        tcpInUnorderSegs    =     0 tcpInUnorderBytes   =     0
        tcpInDupSegs        =     0 tcpInDupBytes       =     0
        tcpInPartDupSegs    =     0 tcpInPartDupBytes   =     0
        tcpInPastWinSegs    =     0 tcpInPastWinBytes   =     0
        tcpInWinProbe       =     0 tcpInWinUpdate      =     0
        tcpInClosed         =     2 tcpRttNoUpdate      =     0
        tcpRttUpdate        =  7749 tcpTimRetrans       =     0
        tcpTimRetransDrop   =     0 tcpTimKeepalive     =     0
        tcpTimKeepaliveProbe=     0 tcpTimKeepaliveDrop =     0
        tcpListenDrop       =     0 tcpListenDropQ0     =     0
        tcpHalfOpenDrop     =     0 tcpOutSackRetrans   =     0
```

Bildschirmausschnitt 41.0.8: Beispiel: `netstat -s` Teil 3

```
IPv4
        ipForwarding        =     2 ipDefaultTTL        =   255
        ipInReceives        = 11669 ipInHdrErrors       =     0
        ipInAddrErrors      =     0 ipInCksumErrs       =     0
        ipForwDatagrams     =     0 ipForwProhibits     =     0
        ipInUnknownProtos   =     0 ipInDiscards        =     0
        ipInDelivers        = 12849 ipOutRequests       = 12187
        ipOutDiscards       =     0 ipOutNoRoutes       =     0
        ipReasmTimeout      =    60 ipReasmReqds        =     0
        ipReasmOKs          =     0 ipReasmFails        =     0
        ipReasmDuplicates   =     0 ipReasmPartDups     =     0
        ipFragOKs           =     0 ipFragFails         =     0
        ipFragCreates       =     0 ipRoutingDiscards   =     0
        tcpInErrs           =     0 udpNoPorts          =   133
        udpInCksumErrs      =     0 udpInOverflows      =     0
        rawipInOverflows    =     0 ipsecInSucceeded    =     0
        ipsecInFailed       =     0 ipInIPv6            =     0
        ipOutIPv6           =     0 ipOutSwitchIPv6     =     1
```

Bildschirmausschnitt 41.0.9: Beispiel: `netstat -s` Teil 4

```
IPv6
        ipv6Forwarding      =     2 ipv6DefaultHopLimit =   255
        ipv6InReceives      =     0 ipv6InHdrErrors     =     0
        ipv6InTooBigErrors  =     0 ipv6InNoRoutes      =     0
        ipv6InAddrErrors    =     0 ipv6InUnknownProtos =     0
        ipv6InTruncatedPkts =     0 ipv6InDiscards      =     0
        ipv6InDelivers      =     0 ipv6OutForwDatagrams=     0
        ipv6OutRequests     =     0 ipv6OutDiscards     =     0
        ipv6OutNoRoutes     =     0 ipv6OutFragOKs      =     0
        ipv6OutFragFails    =     0 ipv6OutFragCreates  =     0
        ipv6ReasmReqds      =     0 ipv6ReasmOKs        =     0
        ipv6ReasmFails      =     0 ipv6InMcastPkts     =     0
        ipv6OutMcastPkts    =     0 ipv6ReasmDuplicates =     0
        ipv6ReasmPartDups   =     0 ipv6ForwProhibits   =     0
        udpInCksumErrs      =     0 udpInOverflows      =     0
        rawipInOverflows    =     0 ipv6InIPv4          =     0
        ipv6OutIPv4         =     0 ipv6OutSwitchIPv4   =     0
```

Bildschirmausschnitt 41.0.10: Beispiel: `netstat -s` Teil 5

```
ICMPv4
        icmpInMsgs          =     5 icmpInErrors        =     0
        icmpInCksumErrs     =     0 icmpInUnknowns      =     0
        icmpInDestUnreachs  =     4 icmpInTimeExcds     =     0
        icmpInParmProbs     =     0 icmpInSrcQuenchs    =     0
        icmpInRedirects     =     0 icmpInBadRedirects  =     0
        icmpInEchos         =     0 icmpInEchoReps      =     1
        icmpInTimestamps    =     0 icmpInTimestampReps =     0
        icmpInAddrMasks     =     0 icmpInAddrMaskReps  =     0
        icmpInFragNeeded    =     0 icmpOutMsgs         =     5
        icmpOutDrops        =   128 icmpOutErrors       =     0
        icmpOutDestUnreachs =     5 icmpOutTimeExcds    =     0
        icmpOutParmProbs    =     0 icmpOutSrcQuenchs   =     0
        icmpOutRedirects    =     0 icmpOutEchos        =     0
        icmpOutEchoReps     =     0 icmpOutTimestamps   =     0
        icmpOutTimestampReps=     0 icmpOutAddrMasks    =     0
        icmpOutAddrMaskReps =     0 icmpOutFragNeeded   =     0
        icmpInOverflows     =     0
```

Bildschirmausschnitt 41.0.11: Beispiel: `netstat -s` Teil 6

```
ICMPv6
        icmp6InMsgs         =     0 icmp6InErrors       =     0
        icmp6InDestUnreachs =     0 icmp6InAdminProhibs =     0
        icmp6InTimeExcds    =     0 icmp6InParmProblems =     0
        icmp6InPktTooBigs   =     0 icmp6InEchos        =     0
        icmp6InEchoReplies  =     0 icmp6InRouterSols   =     0
        icmp6InRouterAds    =     0 icmp6InNeighborSols =     0
        icmp6InNeighborAds  =     0 icmp6InRedirects    =     0
        icmp6InBadRedirects =     0 icmp6InGroupQueries =     0
        icmp6InGroupResps   =     0 icmp6InGroupReds    =     0
        icmp6InOverflows    =     0
        icmp6OutMsgs        =     0 icmp6OutErrors      =     0
        icmp6OutDestUnreachs=     0 icmp6OutAdminProhibs=     0
        icmp6OutTimeExcds   =     0 icmp6OutParmProblems=     0
        icmp6OutPktTooBigs  =     0 icmp6OutEchos       =     0
        icmp6OutEchoReplies =     0 icmp6OutRouterSols  =     0
        icmp6OutRouterAds   =     0 icmp6OutNeighborSols=     0
        icmp6OutNeighborAds =     0 icmp6OutRedirects   =     0
        icmp6OutGroupQueries=     0 icmp6OutGroupResps  =     0
        icmp6OutGroupReds   =     0
```

Bildschirmausschnitt 41.0.12: Beispiel: `netstat -s` Teil 7

```
IGMP:
          0 messages received
          0 messages received with too few bytes
          0 messages received with bad checksum
          0 membership queries received
          0 membership queries received with invalid field(s)
          0 membership reports received
          0 membership reports received with invalid field(s)
          0 membership reports received for groups to which we belong
          0 membership reports sent
$
```

Bildschirmausschnitt 41.0.13: Beispiel: `netstat -s` Teil 8

Ich glaube nicht das Sie wollen daß ich jetzt alles erkläre ...

-M Zeigt die Multicast Routingtabelle an

-D Zeigt die Konfigurierten DHCP Interfaces an

Zusatzoptionen

-a Zeigt alles an (in conjunct with -a, -i -r)

-n Zeigt IP Addressen und Netzwerkaddressen als Nummern an. `netstat` versucht ständig diie IP Addressen in Namen zuwandelt. Hat man ein Problem mit der Namensauflösung (DNS) dann bleibt das Programm `netstat` scheinbar hängen, wenn kein DNS der Welt einen Namen auflösen kann. Man muß dann mit der Option *-n* arbeiten und die Namensauflösung zu umgehen.

-v Verbose. Zeigt noch mehr details an

-P $< protocol >$ Beschränkt die Ausgabe auf ein bestimmtes Protocol. z.B. `netstat -sP tcp`

Beispiele

siehe oben !

Exitstatus

Tabelle 41.41: Exitcodes von `netstat`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

/etc/default/inet_type Standardeinstellungen für IP

Siehe auch

arp, iostat, ifconfig, ...

od

Aufruf

```
od
```

Beschreibung

Das Programm od

Optionen

-v

Beispiele

```
cmd-od-samples
```

Bildschirmausschnitt 41: Beispiele zu od

Exitstatus

Tabelle 41.42: Exitcodes von od

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

patchadd

Aufruf

```
patchadd
```

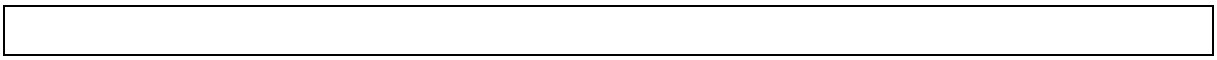
Beschreibung

Das Programm `patchadd`

Optionen

-v

Beispiele



Bildschirmausschnitt 42: Beispiele zu `patchadd`

Exitstatus

Tabelle 41.43: Exitcodes von `patchadd`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

patchrm

Aufruf

```
patchrm
```

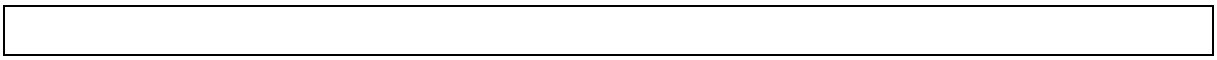
Beschreibung

Das Programm `patchrm`

Optionen

-v

Beispiele



Bildschirmausschnitt 43: Beispiele zu `patchrm`

Exitstatus

Tabelle 41.44: Exitcodes von `patchrm`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

pkgadd

Aufruf

```
pkgadd
```

Beschreibung

Das Programm `pkgadd`

Optionen

-v

Beispiele

```
cmd-pkgadd-samples
```

Bildschirmausschnitt 44: Beispiele zu `pkgadd`

Exitstatus

Tabelle 41.45: Exitcodes von `pkgadd`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

pkgchk

Aufruf

```
pkgchk
```

Beschreibung

Das Programm `pkgchk`

Optionen

-v

Beispiele

```
cmd-pkgchk-samples
```

Bildschirmausschnitt 45: Beispiele zu `pkgchk`

Exitstatus

Tabelle 41.46: Exitcodes von `pkgchk`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

pkginfo

Aufruf

```
pkginfo
```

Beschreibung

Das Programm `pkginfo`

Optionen

-v

Beispiele

```
cmd-pkginfo-samples
```

Bildschirmausschnitt 46: Beispiele zu `pkginfo`

Exitstatus

Tabelle 41.47: Exitcodes von `pkginfo`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

pkgrm

Aufruf

```
pkgrm
```

Beschreibung

Das Programm `pkgrm`

Optionen

-v

Beispiele

```
cmd-pkgrm-samples
```

Bildschirmausschnitt 47: Beispiele zu `pkgrm`

Exitstatus

Tabelle 41.48: Exitcodes von `pkgrm`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

pmadm

Aufruf

```
pmadm -a [-p <pmtag> | -t <type>] -s <svctag> -i <id>
          -m <pmspecific> -v <ver> [-f <xu>] [-y <comment>]
          [-z <script>]

pmadm -{r|e|d} -p <pmtag> -s <svctag>

pmadm -{l|L} [-t <type> | -p <pmtag>] [-s <svctag>]

pmadm -g -p <pmtag> -s <svctag> [-z <script>]
```

Beschreibung

Das Programm `pmadm` wird zur Konfiguration von Portmonitoren eingesetzt. Die Portmonitore selbst werden mit `sacadm`[4] Konfiguriert. Um den Portmonitoren zu sagen an welchen Schnittstellen und welche Dienste er ausführen soll ist `pmadm` nötig.

Optionen

-a Mit der Option *-a* wird einem Portmonitor ein neuer Service zugewiesen. Dazu werden folgende Optionen benötigt :

- **-p** $< pmtag >$ Angabe des Namens des Portmonitors. Der Name wurde bei der Konfiguration mit `sacadm` entsprechend angegeben
- **-t** $< type >$ Alternativ kann der Typ des Portmonitors angegeben werden. Entweder *ttymon* oder *listen*
- **-s** $< svctag >$ Der Name des Services unter dem man den Eintrag später identifizieren kann. Der Name kann freigewählt werden. Im Normalfall legt man die Namen *tty¡x¿* an und die Port zu kennzeichnen
- **-i** $< id >$ Angabe der Benutzer-ID unter dem der Service laufen soll. Im Normalfall ist das *root*
- **-m** $< pmspecific >$ Mit *-m* werden Portmonitor zu Service Daten gespeichert. Das Format ist im Normalfall egal, da man `ttyadm`[5] als Substitute verwendet um die geforderten Daten in das richtige Format zu bekommen. Wichtig ist das die Ausgabe von `ttyadm` als ein Argument übergeben wird. Von daher muß man Hochkommas " dumdrumherum machen
- **-v** $< ver >$ Angabe der Version. Auch hier wird ein Substitute auf `ttyadm -V` bzw. `nlsadmin -V` gemacht
- **-f** $< xu >$ Mit *-f* wird der Status des Portservices festgelegt. Mit dem *u* wird festgelegt, daß wenn sich jemand einloggt ein Eintrag in der `/var/adm/utmp` stattfindet, somit ist der Benutzer von `who` erfassbar. Ein *x* besagt das der Service nicht verfügbar ist

[4] `sacadm` - Siehe Kommandoreferenz Seite 381
[5] `ttyadm` - Siehe Kommandoreferenz Seite 397

-y < *comment* > Ein Kommentar

-z < *script* > Konfigurationsscript

Um ein Portmonitor mit einem Service entsprechend zu Konfigurieren kann wie folgt vorgegangen werden :

```
eagle # pmadm -a -p mytty -s ttyb -i root -v ‘ttyadm -V‘
                -f u -m "‘ttyadm -l 9600 -s /usr/bin/login -d /dev/term/b‘"
```

Bildschirmausschnitt 48: Beispiele zu `pmadm` - Konfiguration hinzufügen

Hier wird der Loginservice hinzugefügt.

-r Entfernt ein Eintrag. Wobei mit *-p* und mit *-s* der Eintrag spezifiziert wird

-e Schaltet die Konfiguration ein. (enable)

-d Schaltet die Konfiguration aus. (disable)

-l Mit der Option *-l* wird die Konfiguration angezeigt. Die Ausgabe sieht wie folgt aus :

```
eagle # pmadm -l
PMTAG   PMTYPE  SVCTAG  FLGS ID       <PMSPECIFIC>
mytty   ttymon  ttyb    u    root     /dev/term/b .......
mytty   ttymon  ttya    ux   root     /dev/term/a .......
```

Bildschirmausschnitt 49: Beispiele zu `pmadm` - Konfiguration hinzufügen

Die Tabelle hat folgende Felder

PMTAG Der Name der Portmonitors. Der wird mit `sacadm` entsprechend Konfiguriert

PMTYPE Der Typ des Portmonitors. Entweder *ttymon* oder *listen*. Wird auch mit `sacadm` konfiguriert

SVCTAG Der Name des Services, den man freiwählen und mit *-s* angegeben hat

FLGS Die eingestellten Flags. Siehe Option *-f*.

ID Die Benutzer ID unter dem der Service läuft

PMSPECIFIC Die Konfiguration des Services. Siehe dazu `ttyadm`

-L Die Option *-L* zeigt die Konfiguration im weiterverarbeitungsgerechten Format an. Das Format ist durch Doppelpunkte getrennt und kann wunderbar in Scripts ausgewertet werden. Die Reihenfolge entspricht dem der Tabelle

Beispiele

siehe Beschreibung

Exitstatus

Tabelle 41.49: Exitcodes von `pmadm`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

/etc/saf/*, /var/saf/*

Siehe auch

sacadm, ttyadm, nlsadmin

poweroff

Aufruf

```
poweroff
```

Beschreibung

Das Programm `poweroff`

Optionen

-v

Beispiele

Bildschirmausschnitt 50: Beispiele zu `poweroff`

Exitstatus

Tabelle 41.50: Exitcodes von `poweroff`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

prtvtoc

Aufruf

```
prtvtoc
```

Beschreibung

Das Programm `prtvtoc`

Optionen

-v

Beispiele

```
cmd-prtvtoc-samples
```

Bildschirmausschnitt 51: Beispiele zu `prtvtoc`

Exitstatus

Tabelle 41.51: Exitcodes von `prtvtoc`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

pwd

Aufruf

```
pwd
```

Beschreibung

Das Programm pwd

Optionen

-v

Beispiele

```
cmd-pwd-samples
```

Bildschirmausschnitt 52: Beispiele zu pwd

Exitstatus

Exitcodes von pwd

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

quotacheck

Aufruf

```
quotacheck
```

Beschreibung

Das Programm `quotacheck`

Optionen

-v

Beispiele

```
cmd-quotacheck-samples
```

Bildschirmausschnitt 53: Beispiele zu `quotacheck`

Exitstatus

Exitcodes von `quotacheck`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

quotaoff

Aufruf

```
quotaoff
```

Beschreibung

Das Programm `quotaoff`

Optionen

-v

Beispiele

```
cmd-quotaoff-samples
```

Bildschirmausschnitt 54: Beispiele zu `quotaoff`

Exitstatus

Exitcodes von `quotaoff`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

quotaon

Aufruf

```
quotaon
```

Beschreibung

Das Programm `quotaon`

Optionen

-v

Beispiele

```
cmd-quotaon-samples
```

Bildschirmausschnitt 55: Beispiele zu `quotaon`

Exitstatus

Exitcodes von `quotaon`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

reboot

Aufruf

```
reboot
```

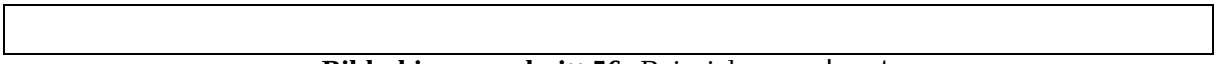
Beschreibung

Das Programm `reboot`

Optionen

-v

Beispiele



Bildschirmausschnitt 56: Beispiele zu `reboot`

Exitstatus

Tabelle 41.56: Exitcodes von `reboot`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

rm

Aufruf

```
rm
```

Beschreibung

Das Programm `rm`

Optionen

-v

Beispiele

```
cmd-rm-samples
```

Bildschirmausschnitt 57: Beispiele zu `rm`

Exitstatus

Exitcodes von `rm`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

rmdir

Aufruf

```
rmdir
```

Beschreibung

Das Programm `rmdir`

Optionen

-v

Beispiele

```
cmd-rmdir-samples
```

Bildschirmausschnitt 58: Beispiele zu `rmdir`

Exitstatus

Exitcodes von `rmdir`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

sac

Aufruf

```
sac -t <interval>
```

Beschreibung

Das Programm `sac` ist der Controller des SAF Systems. Er kontrolliert die anderen Portmonitore wie `ttymon` oder `listen`. Er wird benötigt wenn man die Seriellen Schnittstellen zum Login freigeben will, oder um Netzwerkdienste anzubieten.

Der `sac` wird grundsätzlich von `init` gestartet.

Optionen

-t $<$ *interval* $>$ Angabe in Sekunden, wie oft `sac` seine Prozesse prüfen soll und diese gegebenfalls neu startet.

Beispiele

```
$ grep sac /etc/inittab
sc:234:respawn:/usr/lib/saf/sac -t 300
$
```

Bildschirmausschnitt 59: Eintrag in der /etc/inittab

Exitstatus

Tabelle 41.59: Exitcodes von `sac`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

/etc/saf/_sysconfig, /etc/saf/_sactab, /var/saf/_log

Siehe auch

`sacadm`[6], `pmadm`[7], `ttyadm`[8]

[6] `sacadm` - Siehe Kommandoreferenz Seite 381
[7] `pmadm` - Siehe Kommandoreferenz Seite 357
[8] `ttyadm` - Siehe Kommandoreferenz Seite 397

sacadm

Aufruf

```
sacadm -a -p <pmtag> -t <type> -c <cmd> -v <ver> [ -f <dx> ]
            [-n <count>]  [-y <comment>]  [-z <script>]

sacadm -{r|s|k|e|d} -p <pmtag>

sacadm -{l|L} [-p <pmtag> | -t <type>]

sacadm -g -p <pmtag> [-z <script>]

sacadm -G [-z <script>]

sacadm -x [-p <pmtag>]
```

Beschreibung

Das Programm `sacadm` konfiguritert das SAF System. Mit dem Programm werden Portmonitore hinzugefügt/geändert oder gelöscht.

Optionen

-a Die Option *-a* fügt der Konfiguration ein Portmonitor hinzu. Dazu sind die folgenden Optionen erlaubt

-p $< pmtag >$ Angabe des Portmonitornamens. Dieser kann freigewählt werden und dient zur unterscheidung zu anderen Portmonitoren

-t $< type >$ Legt den Typ des Portmonitors fest. Es kann entweder *ttymon* für serielle Schnittstellen oder *listen* für Netzwerkmonitore angegeben werden

-c $< cmd >$ Gibt das Programm mit absolutem Path an. Jenach Typeinstellung kann es `/usr/lib/saf/ttymon` oder `/usr/lib/saf/listen` sein

-v $< ver >$ Angabe der Versionsnummer. Die sollte als Substitute vom Programm `ttyadm -V` kommen, sofern es sich um den Typ *ttymon* handelt. Bei dem Netzwerkmonitor *listen* sollte man `nlsadmin -V` verwenden

-f $< dx >$ Das Optionale Argument *-f* legt den Status des Portmonitors fest. Ein *-f d* wird der Portmonitor nicht freigegeben (disable). Durch *-f x* wird der Portmonitor nicht gestartet. Standard ist das der Portmonitor gestartet und aktiviert wird, wenn *-f* fehlt.

-n $< count >$ Angabe der Startwiederholungen eines Portmonitors. Die Angabe von 0 (NULL) oder das fehlende *-n* bedeuten endlose Versuche den Portmonitor zu starten

-y $< comment >$ Angabe eines Kommentars. Der jedoch sollte unbedingt in Hochkommas " gesetzt werden, wegen der Leerzeichen

-z $< script >$ Angabe eines Konfigurationsscriptes für den Portmonitor

```
eagle # sacadm -a -p mytty -t ttymon -c /usr/lib/saf/ttymon
         -v 'ttyadm -V' -y "Mein TTY Monitor"
eagle #
```

Bildschirmausschnitt 60: Beispiele zu `sacadm` - Hinzufügen eines Portmonitors

Fügt ein Portmonitor *ttymon* mit dem Namen *mytty* der Konfiguration hinzu.

-r -p $< pmtag >$ Entfernt eine Portmonitorkonfiguration

-s -p $< pmtag >$ Startet ein Portmonitor (start)

-k -p $< pmtag >$ Stoppt ein Portmonitor (kill)

-e -p $< pmtag >$ Erlaubt den Zugriff auf ein Portmonitor (enable)

-d -p $< pmtag >$ Verweigert den Zugriff auf den Portmonitor (disable)

-l Ausgabe der Konfiguration in einer übersichtlichen Tabelle. Wird zusätzlich noch *-p* oder *-t* angegeben, werden nur die Treffer ausgegeben. Die Liste hat folgendes Format :

```
eagle # sacadm -l
PMTAG      PMTYPE     FLGS RCNT STATUS      COMMAND
mytty      ttymon     -    0    ENABLED     /usr/lib/saf/ttymon #Mein ...
eagle #
```

Bildschirmausschnitt 61: Beispiele zu `sacadm` - Ausgabe der Konfiguration

Wobei die folgenden Felde folgendes Bedeuten :

PMTAG Der Name des Portmonitors der mit *-p* angegeben werden muß

PMTYPE Der Typ des Portmonitors. Entweder *ttymon* oder *listen*

FLGS In Flags wird der Status des Portmonitors eingetragen. Steht dort ein *d* dann ist er disabled, ein *x* kenzeichnet das der Portmonitor nicht gestartet ist. Ein - steht für alles klar. (vgl. Option *-f*)

RCNT Anzahl der maximalen Startversuche.

STATUS Zeigt den Status des Portmonitors an. Die folgende Liste zeigt alle Staten : STARTING, ENABLED, DISABLED, STOPPING, NOTRUNNING, FAILED.

COMMAND Das Programm welches sac ausführen soll. (vgl. Option *-c*). Und der Kommentar

-L Soll die Ausgabe der Konfiguration in einem Script weiterverarbeitet werden eignet sich *-L* am Besten :

```
eagle # sacadm -L
mytty:ttymon::0:ENABLED:/usr/lib/saf/ttymon#Mein TTY Monitor
eagle #
```

Bildschirmausschnitt 62: Beispiele zu `sacadm` - Ausgabe der Konfiguration (Kurzform)

Wie man sieht werden die Felder durch Doppelpunkte voneinander getrennt. Die Reihenfolge entspricht deer wie sie auch mit *-l* erzeugt wird

-G Wird die Option *-z* nich angegeben, dann wird die Configurationsdatei vom `sac` (/etc/saf/_sysconfig angezeigt. Mit der Option *-z* kann man diese Konfigurationsdatei entsprechend neu setzen.

-g Mach das gleiche wie *-G* jedoch zeigt und verändert es die Konfigurationsscripte eines Portmonitors

-x Zwingt den `sac` seine Konfiguration neu einzulesen

Beispiele

siehe in der Beschreibung

Exitstatus

Tabelle 41.60: Exitcodes von `sacadm`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

/etc/saf/$< pmtag >$/_config, /etc/saf/_sactab, /etc/saf/_sysconfig

Siehe auch

pmadm, sac

setfacl

Aufruf

```
setfacl
```

Beschreibung

Das Programm `setfacl`

Optionen

-v

Beispiele

```
cmd-setfacl-samples
```

Bildschirmausschnitt 63: Beispiele zu `setfacl`

Exitstatus

Exitcodes von `setfacl`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

shutdown

Aufruf

```
shutdown
```

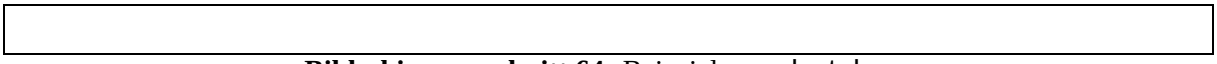
Beschreibung

Das Programm `shutdown`

Optionen

-v

Beispiele



Bildschirmausschnitt 64: Beispiele zu `shutdown`

Exitstatus

Tabelle 41.62: Exitcodes von `shutdown`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

sleep

Aufruf

```
sleep sekunden
```

Beschreibung

Das Programm `sleep` legt ein Prozess für *sekunden* schlafen.

Beispiele

```
cmd-sleep-samples
$ date; sleep 5; date
Freitag, 28. April 2000, 10:50:24 Uhr MEST
Freitag, 28. April 2000, 10:50:29 Uhr MEST
$
```

Bildschirmausschnitt 65: Beispiele zu `sleep`

Exitstatus

Exitcodes von `sleep`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

stty

Aufruf

```
stty
```

Beschreibung

Das Programm `stty`

Optionen

-v

Beispiele

Bildschirmausschnitt 66: Beispiele zu `stty`

Exitstatus

Tabelle 41.64: Exitcodes von `stty`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

tip

Aufruf

```
tip
```

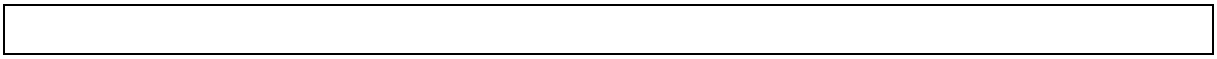
Beschreibung

Das Programm `tip`

Optionen

-v

Beispiele



Bildschirmausschnitt 67: Beispiele zu `tip`

Exitstatus

Tabelle 41.65: Exitcodes von `tip`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

touch

Aufruf

```
touch [optionen] <dateiname>
```

Beschreibung

Das Programm `touch` setzt Zeitstempel von Dateien auf das aktuelle Datum. Wird meist bei der Programmierung verwendet um dem Compiler anzuzeigen das diese Datei neuer ist als alle anderen.

Das Programm `touch` hat jedoch auch noch ganz woanderst ungewollt Einzug erhalten. In der Erstellung von LOCK Dateien, um Modems z.B. bei der Benutzung zu Kennzeichnen erstellt der Prozeß irgendwo (meist im /var Verzeichnis) eine LOCK Datei. Der Vorteil von `touch` ist, daß nur ein Prozeß in der Lage ist ein `touch` auszuführen. Der andere Prozeß wird blockiert.

Optionen

-a Ändert den Zugriffszeitstempel

-c Verhindert die Erstellung der Datei

-m Ändert nur die Änderungszeit

Beispiele

```
$ ls
$ touch aFile
$ ls -l
Gesamt 0
-rw-r--r--   1 whurst   whurst         0 Jun  1 19:37 aFile
$ sleep 100
$ touch aFile
$ ls -l
Gesamt 0
-rw-r--r--   1 whurst   whurst         0 Jun  1 19:39 aFile
$
```

Bildschirmausschnitt 68: Beispiele zu `touch`

Exitstatus

Tabelle 41.66: Exitcodes von `touch`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

ttyadm

Aufruf

```
ttyadm [-b] [-c] [-h] [-I] [-r <count>] [-i <msg>]
            [-m <modules>] [-p <prompt>] [-t <timeout>]
            [-S {y|n}] [-T <termtype>]
            -d <device> -l <ttylabel> -s <service>

ttyadm -V
```

Beschreibung

Das Programm `ttyadm` formatiert die übergebenen Parameter für das Programm `pmadm`[9]. Es hat keine Auswirkungen auf die Konfiguration. Sondern formatiert einfach nur die Ausgabe

Optionen

-b Setzt den bidirektionalen Modus. Man kann die Leitung also in beiden Richtungen nutzen

-c Connect-On-Carrier. Ist die Option gegeben wird der Service sofort gestartet ohne vorher die Baudrate zu ermitteln. Der Prompt wird auch nicht dargestellt (*-p*)

-h Wird die Option *-h* nicht gesetzt, dann wird das Auflegen der Leitung durch das setzten der Leitungsgeschwindigkeit auf 0 (NULL) entsprechend geregelt

-I Initialisiert den Port nur einmal und wird meist zur Konfiguration eines Ports verwendet. Solaris definiert nach der Installation 2 Serielle Ports. Dort ist auch nur *-I* gegeben

-r $< count >$ Wartet genau $< count >$ Zeichen, bevor der Monitor reagiert. Wird die Angabe weggelassen wird 0 (NULL) voreingestellt, und dann muß man nach dem Connect einmal Return drücken

-i $< msg >$ Ist der Service nicht Aktiv (disable - `pmadm -d ...` dann kann man hier eine Nachricht für den Benutzer ablegen die dann erscheint. Wichtig ist das die Nachricht in Hochkommas eingeschlossen wird

-m $< modules >$ Liste der *pushable* Streammodules. Meist werden `ldterm,ttcompat` eingesetzt.

-p $< prompt >$ Angabe des Promptes. Normalerweise *login:*

-t $< timeout >$ Angabe der Sekunden wann ein Port geschlossen werden soll. Wird die Angabe weggelassen wird 0 (NULL) voreingestellt, was soviel heisst wie : Nie !

-S $< yn >$ Setzen der Software-Carrier. Mit *-S y* wird die Softwar aktiviert mit *-S n* nicht

[9] `pmadm` - Siehe Kommandoreferenz Seite 357

-T $< termtype >$ Voreinstellung für das verwendete Terminal. Jedes Terminal wird mit verschiedenen Steuersequenzen angesprochen. Das muß hier entsprechend Eingetragen sein. Alle Terminaltypen aus der /usr/share/lib/terminfo sind zugelassen. Weiterhin wird des Users TERM Variable auf den entsprechenden Wert gesetzt

-d $< device >$ Angabe des Devices. Zu verwenden sind die logischen Seriellenports, wie /dev/term/*

-l $< ttylabel >$ Angabe der Terminaleinstellungen. Der Begriff wird in der /etc/ttydefs[10] gesucht und der wird eingestellt

-s $< service >$ Angabe des Programms das gestartet werden soll

-V Ausgabe der Versionsnummer. Wichtig für `sacadm` und `pmadm`

Die Ausgabe des Befehls ist für `pmadm` wichtig. Das Format ist auch wieder durch Doppelpunkte voneinander getrennt und in Spalten aufgeteilt :

Tabelle 41.67: Spaltenbeschreibung: Ausgabe von `ttyadm`

Spalte	Bedeutung
1	Das Device welches mit *-d* konfiguriert wurde
2	Auflistung der Optionen die mit *-b, -c, -h, -I* gemacht wurden. Wurde ein *-r ¡count¿* angegeben wird ein *r* zusätzlich eingetragen
3	Wurde mittels *-r ¡count¿* die Anzahl der Zeichen eingestellt, dann wird die Anzahl hier notiert
4	Das Programm welches mittels *-s* eingestellt wurde
5	Wurde ein Timeout mittels *-t* angegeben, dann wird es hier notiert
6	Der ttylabel von *-l* wird hier notiert
7	Die Liste der Module von *-m* werden durch Kommas getrennt hier eingetragen
8	Der Prompt (Achtung: Doppelpunkte die von *-p* kommen werden automatisch entwertet)
9	Die Interaktive Nachricht von *-i*
10	Der Terminaltyp von *-T*
11	Software Carrier von *-S*

Das entspricht auch der Reihenfolge bei `pmadm -l` im Feld **PMSPECIFIC**

[10] /etc/ttydefs - Siehe Konfigurationsdateien Referenz Seite 435

Beispiele

```
eagle # ttyadm -d /dev/term/a -s /usr/bin/login -l 9600
        -b -T vt100 -m ldterm,ttcompat -i "Inactive"
        -p "Login please :" -c -h -I -r 2 -t 4 -S y
/dev/term/a:bchIr:2:/usr/bin/login:4:9600:ldterm,ttcompat:Login plea
se  :Inactive:vt100:y:
eagle #
```

Bildschirmausschnitt 69: Beispiele zu `ttyadm`

Exitstatus

Tabelle 41.68: Exitcodes von `ttyadm`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

/etc/ttydefs

Siehe auch

pmadm, sacadm

type

Aufruf

```
type
```

Beschreibung

Das Programm `type`

Optionen

-v

Beispiele

```
cmd-type-samples
```

Bildschirmausschnitt 70: Beispiele zu `type`

Exitstatus

Exitcodes von `type`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

umask

Aufruf

```
umask
```

Beschreibung

Das Programm `umask`

Optionen

-v

Beispiele

```
cmd-umask-samples
```

Bildschirmausschnitt 71: Beispiele zu `umask`

Exitstatus

Exitcodes von `umask`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

uname

Aufruf

```
uname [-aimnprsvX]
```

Beschreibung

Das Programm `uname` gibt Ihnen Auskunft über das verwendete Betriebsystem und der verwendeten Prozessoren.

Optionen

-a Gibt Ihnen alle Informationen aus.

-i Gibt Ihnen Auskunft über die verwendete Hardware aus. *-i* ist nicht auf Linuxsystemen verfügbar.

-m Gibt Auskunft über den Typ der Hardware.

-n Gibt den Hostnamen des Rechners aus. Mit diesem Namen wird der Rechner im Netzwerk identifiziert.

-p Gibt den verwendetet Prozessorbus aus.

-r Gibt die Releasenummer des Betriebsystems aus. Unter Linux wird die Kernelversion angezeigt,

-s Gibt den Namen des Betriebsystems aus

-v Gibt die Version des Betriebsystems aus. Unter Linux wird das Datum des Kernels angezeigt. Unter Solaris wird unteranderem die Patchnummer angezeigt.

-X Gibt alle Informationen aus. Unter Linux ist *-X* nicht vorhanden.

Beispiele

```
cmd-uname-samples
crow: whurst $ uname -a
SunOS crow 5.7 Generic_106541-04 sun4u sparc SUNW,Ultra-5_10
....
tiger: whurst $ uname -a
Linux tiger 2.2.10 #3 Wed Sep 8 19:57:14 CEST 1999 i586 unknown
```

Bildschirmausschnitt 72: Beispiele zu `uname`

Exitstatus

Exitcodes von `uname`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Siehe auch

arch(1), isalist(1), sysinfo(2), uname(2), attributes(5), environ(5)

which

Aufruf

```
which
```

Beschreibung

Das Programm `which`

Optionen

-v

Beispiele

```
cmd-which-samples
```

Bildschirmausschnitt 73: Beispiele zu `which`

Exitstatus

Tabelle 41.72: Exitcodes von `which`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

who

Aufruf

```
who [optionen]
```

Beschreibung

Das Programm `who` erstellt eine Liste mit allen Benutzern die sich zur Zeit am System angemeldet haben. Weiterhin werden auch Systemprozesse angezeigt und der Runlevel.

Wird eine Ausgabe mit Header (-H) gewählt dann besitzen die Werte folgende Beschreibungen :

NAME Des Benutzers Loginname, oder der Prozessname der auf ein Login wartet

STATE Status des Terminals. Wird bei -T angezeigt.

- Ein Pluszeichen (`+`) zeigt die Schreibfähigkeit an
- Ein Minuszeichen (`-`) zeigt die nicht Schreibfähigkeit an
- Ein Fragendeszeichen (`?`) zeigt an, daß der Status nicht ermittelt werden konnte

LINE Devicename des Terminals. Ein Punkt (`.`) steht für Systemprozesse

TIME Die Zeit der ersten Benutzung dieser Leitung. Bei Benutzern ist es das Logindatum

IDLE Die Zeit in Minuten oder Stunden seit der letzten Aktion auf dieser Leitung. Ein Punkt (`.`) zeigt an, daß das Terminal vor weniger als 5 oder 10 Sekunden benutzt wurde.

COMMENT Dort wird meist angezeigt von wo aus diese Leitung benutzt wird. Wenn die Benutzer sich via Telnet einloggen, wird der Hostname angezeigt. Bei Graphischer Oberfläche wird dort das Display angezeigt.

PID Die PID der Prozesses der die Leitung besitzt.

Optionen

-a Mit *-a* wird alles angezeigt

-b Mit der Option *-b* wird nur der letzte Systemboot angezeigt

-H Mit der Option *-H* wird eine Kopfzeile mit ausgegeben

-m Mit der Option *-m* wird nur die Information angezeigt, die mit dem aktuellen Terminal zusammen hängt

-q Mit der Option *-q* werden nur die eingeloggten Benutzer ausgegeben. Keine weiteren Informationen. Mit *-n xx* kann man die Anzahl der anzuzeigenen Benutzern pro Zeile limitiert werden. (Beim Drucken oder so wichtig)

-r Anzeige des aktuellen Runlevels

-T Zur normalen Anzeige wird noch der Status des Terminals angezeigt. Ein `+` zeigt die `write`-fähigkeit und ein `-` (Minus) zeigt an das Sie auf dieses Terminal nicht schreiben können

Beispiele

```
$ who
whurst     console      Jun  1 10:23    (:0)
whurst     pts/4        Jun  1 10:41    (:0.0)
whurst     pts/6        Jun  1 18:00    (:0.0)
whurst     pts/7        Jun  1 18:02    (:0.0)
$ who -m
whurst     pts/7        Jun  1 18:02    (:0.0)
$ tty
/dev/pts/7
$ who -HaT
NAME     LINE         TIME         IDLE   PID COMMENTS
   .     system boot  Jun 1 10:22
   .     run-level 3  Jun 1 10:22    3     0 S
rc2            .      Jun 1 10:22 7:50   69 id=  s2 term=0   exit=0
rc3            .      Jun 1 10:22 7:50  293 id=  s3 term=0   exit=0
sac            .      Jun 1 10:22 7:50  423 id=  sc
LOGIN    console      Jun 1 10:22 7:38  424
zsmon          .      Jun 1 10:22 7:50  434
whurst + console      Jun 1 10:23 7:38  459      (:0)
whurst + pts/4        Jun 1 10:41 0:12  742      (:0.0)
whurst + pts/6        Jun 1 18:00 0:01 3218      (:0.0)
whurst - pts/7        Jun 1 18:02  .   3228      (:0.0)
whurst   pts/5        Jun 1 18:00 0:12 3204 id=  5 term=0  exit=2 (:0.0)
whurst   pts/5        Jun 1 13:12 0:12 2679 id= /5 term=14 exit=2 (:0.0)
$
```

Bildschirmausschnitt 74: Beispiele zu `who`

Exitstatus

Exitcodes von `who`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

write

Aufruf

```
write
```

Beschreibung

Das Programm `write`

Optionen

-v

Beispiele

```
cmd-write-samples
```

Bildschirmausschnitt 75: Beispiele zu `write`

Exitstatus

Exitcodes von `write`

Exitcode	Beschreibung
0	Fehlerfreie Ausführung
> 0	Fehler trat auf

Environment

Dateien

Siehe auch

Kapitel 42

KONFIGURATIONSDATEIEN REFERENZ

crontab

Verwendungszweck

Die Datei crontab wird für folgende Dienste benotigt : cron.
Die Originaldateien der crontabs befinden sich im /var/spool/cron/crontabs Verzeichnis.

Aufbau und Format der Datei

- Die crontab ist eine ASCII Textdatei
- Als Kommentarzeichen wird das Hash # verwendet
- Leere Zeilen sind nicht erlaubt
- Jede Zeile entspricht
- Als Spaltentrenner koennen mehrere Leerzeichen oder Tabulatoren verwendet werden

Spaltenbedeutungen

Tabelle 42.1: Spaltenbeschreibung: crontab

Spalte	Bedeutung
1	Minute
2	Stunde
3	Tag
4	Monat
5	Wochentag
6	Program

Beispiele

Quelltext 42.0.2 Beispiele für crontab

Siehe auch

/etc/default/init

Verwendungszweck

Die Datei /etc/default/init wird von `init`[1] eingelesen. In dieser Datei befinden sich Systemweite Einstellungen für Sprache und Zeitzone.

Aufbau und Format der Datei

- Die /etc/default/init ist eine ASCII Textdatei
- Als Kommentarzeichen wird das Hash # verwendet
- Die Datei ist vom Aufbau her ähnlich eines Shellscriptes

Environmentvariablen

In der Datei sind meist die folgenden Variablen definiert. Diese werden gleich nach der Installation des Systems dort notiert.

TZ Legt die Zeitzone fest

CMASK Legt die Maskierung der Rechte für neue Dateien fest (siehe umask)

LC_* Legt die Sprachumgebung und die Ländereinstellungen für diverse Ausgaben fest.

Beispiele

Quelltext 42.0.3 Beispiele für /etc/default/init

```
TZ=CET
CMASK=022
LC_COLLATE=de_DE.ISO8859-1
LC_CTYPE=de_DE.ISO8859-1
LC_MESSAGES=de
LC_MONETARY=de_DE.ISO8859-1
LC_NUMERIC=de_DE.ISO8859-1
LC_TIME=de_DE.ISO8859-1
```

Siehe auch

init

[1] `init` - Siehe Kommandoreferenz Seite 295

/etc/inet/hosts

Verwendungszweck

Die Datei /etc/inet/hosts dient als Übersetzungstabelle für Hostnamen und IP Addressen.

Aufbau und Format der Datei

- Die /etc/inet/hosts ist eine ASCII Textdatei
- Als Kommentarzeichen wird das Hash # verwendet
- Leere Zeilen sind erlaubt
- Jede Zeile entspricht einem Eintrag
- Als Spaltentrenner koennen mehrere Leerzeichen oder Tabulatoren verwendet werden

Spaltenbedeutungen

Tabelle 42.2: Spaltenbeschreibung: /etc/inet/hosts

Spalte	Bedeutung
1	IP-Addresse
2	Offizieller Hostname
3...	Optionale Aliasnamen

Beispiele

Quelltext 42.0.4 Beispiele für /etc/inet/hosts

```
127.0.0.1       localhost
10.65.13.10     raven   loghost   # grinns ... no more comments
10.65.13.202    puma              # mein hoch wichtiger server
10.65.13.8      cheetah           # brutalschnell
10.65.13.1      falcon            # fliegt gern ins internet
10.65.13.13     lx                # klein aber super schnell
10.65.13.4      eagle eagle.hurst.pnet nfs2.ha.hurst.pnet
10.65.13.110    dianas-pc
```

Namendienstverfügbarkeit

NIS, NIS+, DNS, FNS

Siehe auch

[RFC 952][2] [RFC 921][3] [RFC 1123][4]

[2] `RFC-952` DoD Internet Host Table Specification
[3] `RFC-921` Domain Name System Implementation Schedule
[4] `RFC-1123` Requirements for Internet Hosts

/etc/inet/netmasks

Verwendungszweck

Die Datei /etc/inet/netmasks bietet dem Administrator eine Datenbank an. In dieser Datei befinden sich Netzmasken bekannter Netzwerke. Diese Datei kann auch via NIS und NIS+ entsprechend zentralisiert werden.

Aus Kompatiblitätsgründen zum BSD Standard befindet sich ein Symbolischer Link unter /etc/netmasks

Aufbau und Format der Datei

- Die /etc/inet/netmasks ist eine ASCII Textdatei
- Als Kommentarzeichen wird das Hash # verwendet
- Leere Zeilen sind, glaube ich, nicht erlaubt
- Jede Zeile entspricht einem Eintrag
- Als Spaltentrenner koennen mehrere Leerzeichen oder Tabulatoren verwendet werden

Spaltenbedeutungen

Tabelle 42.3: Spaltenbeschreibung: /etc/inet/netmasks

Spalte	Bedeutung
1	Netzwerkaddresse
2	Subnetzmaske

Beispiele

Quelltext 42.0.5 Beispiele für /etc/inet/netmasks

```
128.32.0.0      255.255.255.0
128.32.27.0     255.255.255.240
128.32.27.16    255.255.255.240
128.32.27.32    255.255.255.240
128.32.27.48    255.255.255.240
128.32.27.64    255.255.255.240
128.32.27.80    255.255.255.240
128.32.27.96    255.255.255.240
```

Namendienstverfügbarkeit

NIS und NIS+

Siehe auch

[RFC 950][5] und [RFC 1519][6]

[5] `RFC-950` Internet Standard Subnetting Procedure
[6] `RFC-1519` Classless Inter-Domain Routing (CIDR)

/etc/inet/services

Verwendungszweck

Die Datei /etc/inet/services dient zur Umwandlung eines Servicenamens in eine konkrete Portaddresse mit dessen Protokoll. Die meisten Programme öffnen eine Verbindung mit dessen Servicenamen. Damit auch ein echter Port benutzt werden kann ist diese Tabelle wichtig.

Aufbau und Format der Datei

- Die /etc/inet/services ist eine ASCII Textdatei
- Als Kommentarzeichen wird das Hash # verwendet
- Leere Zeilen sind erlaubt
- Jede Zeile entspricht einer Port - Service
- Als Spaltentrenner koennen mehrere Leerzeichen oder Tabulatoren verwendet werden

Spaltenbedeutungen

Tabelle 42.4: Spaltenbeschreibung: /etc/inet/services

Spalte	Bedeutung
1	Offizieller Servicename
2	Portnummer und Protocol
3...	Aliasnamen

Beispiele

Quelltext 42.0.6 Beispiele für /etc/inet/services

```
telnet          23/tcp
domain          53/udp
printer         515/tcp    spooler # line printer spooler
who             513/udp    whod
```

Namendienstverfügbarkeit

NIS, NIS+

Siehe auch

[RFC 1700][7]

[7] RFC-1700 Assigned Numbers

/etc/inittab

Verwendungszweck

Die Datei /etc/inittab wird für das Programm `init`[8] benötigt. Es steuert das Verhalten der einzelnen Runlevels.

Aufbau und Format der Datei

- Die /etc/inittab ist eine ASCII Textdatei
- Kommentare sind nicht erlaubt
- Leere Zeilen sind nicht erlaubt
- Jede Zeile entspricht einem Konfigurationseintrag
- Als Spaltentrenner kommt der Doppelpunkt zum Einsatz

Spaltenbedeutungen

Tabelle 42.5: Spaltenbeschreibung: /etc/inittab

Spalte	Bedeutung
1 (id)	Um ein Konfigurationseintrag zu Kennzeichnen muß hier ein Schlüssel eingegeben werden der nur einmal in der Datei vorkommt. Entweder ein Zeichen oder zwei. Mit `who -a`[9] wird die der noch laufende Prozess und dessen ID angezeigt
2 (rstate)	Liste der Runlevels. Hier kann angegeben werden wann der Prozess in welchen Runlevels gestartet werden soll. Die Liste darf keine Leerzeichen oder Kommas oder sonstiges enthalten.
3 (action)	Aktion die `init` machen soll. Es ist immer nur eine Aktion gültig.
4 (process)	Das Programm welches `init` starten soll. Dort sind wieder Doppelpunkte erlaubt. Man sollte beim Ausschneiden der Spalten darauf achten, nicht das man sich den halben Befehl abschneidet !

Aktionen

In der dritten Spalte darf folgendes stehen :

respawn Der Prozeß wird beim Eintritt in den Runlevel gestartet. Wird der Prozeß verlassen, dann erfolgt ein erneuter Start das Prozesses. Dieses Vorgehen verwendet man z.B. beim SAF oder bei Anrufbeantwortern/Faxservern. Ein Faxserver hat die Angewohnheit nach dem Empfang eines Faxes sich zu beenden, um dem Administrator die Gelegenheit zu bieten diverse Aktionen auszuführen. Beim Verlassen des Runlevels wird der Prozeß gekillt

[8] `init` - Siehe Kommandoreferenz Seite 295
[9] `who -a` - Siehe Kommandoreferenz Seite **??**

wait Der Prozeß wird beim Eintritt in den Runlevel gestartet. `init` wartet jedoch auf die Beendigung des Prozesses. Wird für die gesammten Runlevelscripts benötigt und verwendet

once Der Prozeß wird beim Eintritt in den Runlevel gestartet. `init` wartet nicht auf die Beendigung des Prozesses. Der Prozess selbst wird nach dem Ende nicht erneut gestartet. Wnn `init` in ein Runlevel wechselt und der Prozeß aus dem vorherigen Runlevel läuft noch wird dieser weder neu gestartet noch angehalten

boot Die Prozesse werden wärend des Bootens gestartet. `init` wartet nicht auf die Beendigung der Prozesse und startet sie auch nach dem Ende nicht noch einmal

bootwait Die Prozesse werden nach dem Wechsel vom Booten in den Runlevel 2 gestartet. `init` wartet auf dessen Beendigung

powerfail Die Prozesse werden gestartet, wenn `init` das Signal `SIGPWR` bekommt. Normalerweise hat ein Server eine UPS/USV die mit der Seriellenschnittstelle verbunden ist. Ein Prozess fragt den Status der UPS/USV ab, und wenn der Saft weg ist sendet der Prozeß ein `SIGPWR` an `init`. Normalerweise fährt er dann das System nach ein paar Minuten runter, je nach Leistung der UPS/USV. `init` wartet nicht auf das Ende der Prozesse

powerwait Die Prozesse werden gestartet wenn `init` ein `SIGPWR` bekommt. `init` wartet jedoch auf dessen Beendigung.

off Die Prozesse werden von `init` nicht gestartet. Sondern sollten die Prozesse bereits laufen dann dendet `init` ein `SIGTERM` an den Prozeß und wartet 5 Sekunden. Passiert nix dann wird der Prozess gekillt. Ist für das Runterfahren von Diensten sehr interessant, wird jedoch nicht genutzt.

ondemand siehe **respawn**

initdefault Legt den Default Runlevel fest. Nur ein Runlevel darf im *rstate* stehen. Man sollte vielleicht darauf achten daß man nicht etwa den Runlevel 6, 5 oder 0 wählt.

sysinit Die Prozesse werden direkt nach dem Booten gestartet. `init` wartet auf das Ende der Prozesse. Wird benötigt um z.B. Geräte zu konfigurieren, etc.pp.

Beispiele

Quelltext 42.0.7 Beispiele für /etc/inittab

Siehe auch

/etc/nsswitch.conf

Verwendungszweck

Die Datei /etc/nsswitch.conf wird zur Konfiguration der Suchdienste verwendet. In der Datei wird die Reihenfolge der abzufragenden Namensdienste konfiguriert.

Aufbau und Format der Datei

- Die /etc/nsswitch.conf ist eine ASCII Textdatei
- Als Kommentarzeichen wird das Hash # verwendet
- Leere Zeilen sind erlaubt
- Die erste Spalte definiert die Datenbank gefolgt unmittelbar von einem Doppelpunkt
- Die nachfolgenden durch Leerzeichen getrennten Werte geben die Namensdienstquelle an.

Spaltenbedeutungen

Tabelle 42.6: Spaltenbeschreibung: /etc/nsswitch.conf

Spalte	Bedeutung
1	Datenbank
2..	Namensdienstquelle

Als Datenbankschlüssel kommen die folgenden Werte in Frage :

Tabelle 42.7: Übersicht der Datenbankschlüssel in der /etc/nsswitch.conf

Schlüssel	Bedeutung
aliases	Mail Aliasnamen für `sendmail`[10]
automount	Datenbankdateien für den AutoMounter. Zum einen kann es die `auto_master` und die `auto_home`
bootparams	Für BOOTP und RARP Definierungen. Muß bei JumpStart immer zuerst von den lokalen Dateien kommen. Siehe auch /etc/bootparams[11]
ethers	Wird für RARP benötigt. In der Datenbank befinden sich die MAC Adressen und die dazugehörigen IP/Hostnamen. Siehe auch /etc/ethers[12]
group	Gruppendatenbank. Siehe auch /etc/group[13]
hosts	Hosttabelle. Siehe auch /etc/inet/hosts[14]
ipnodes	Hosttabelle. Siehe auch /etc/inet/ipnodes[15]

[10] `sendmail` - Siehe Kommandoreferenz Seite **??**
[11] /etc/bootparams - Siehe Konfigurationsdateien Referenz Seite **??**
[12] /etc/ethers - Siehe Konfigurationsdateien Referenz Seite **??**
[13] /etc/group - Siehe Konfigurationsdateien Referenz Seite **??**
[14] /etc/inet/hosts - Siehe Konfigurationsdateien Referenz Seite 419
[15] /etc/inet/ipnodes - Siehe Konfigurationsdateien Referenz Seite **??**

▲

Schlüssel	Bedeutung
netgroup	NIS Netzwerkgruppen. Siehe auch /etc/netgroup[16]
netmasks	Netzwerkmaskentabelle. Siehe auch /etc/inet/netmasks[17]
networks	Netzwerkaddressen. Siehe auch /etc/inet/networks[18]
passwd	Benutzerdatenbank. Siehe auch /etc/passwd[19]
printers	Druckerkonfigurationsdateien. Siehe auch /etc/printers.conf[20]
protocols	TCP/IP Protokolltabelle. Siehe auch /etc/inet/protocols[21]
publickey	Der öffentliche Schlüssel bei Übertragungen mittels Secure-RPC
rpc	Die RPC Tabelle. Siehe auch /etc/rpc[22]
sendmailvars	Wird für Sendmail benötigt.
services	Portnummerntabelle. Siehe auch /etc/inet/services[23]

Als Namensdienste kommen die folgenden Werte in Frage :

Tabelle 42.8: Übersicht der Namensdienstquellen in der /etc/nsswitch.conf

Schlüssel	Bedeutung
files	Die Daten werden in den lokalen Dateien des Hosts gesucht. Also entweder in /etc oder in /etc/inet
nis	Die Daten werden auf dem NIS Server gesucht. Ist der Host kein NIS Client wird diese Quelle nicht beachtet
nisplus	Die Daten werden auf dem NIS+ Server gesucht. Auch hier gilt. Ist der Host kein NIS+ Client dann wird diese Quelle ignoriert.
ldap	Verwendet zur Suche von Daten ein LDAP Server. Ist kein LDAP Server konfiguriert wird diese Quelle ignoriert
dns	Verwendet zur Auflösung von IP Addressen oder Hostnamen ein DNS Server. Die Quelle kann nur bei dem Schlüssel **hosts** angegeben werden. Ist der Host kein DNS Client wird die Quelle ignoriert. (Siehe /etc/resolv.conf[24])
compat	Darf nur im Schlüssel **passwd** und/oder **group** definiert sein. Es aktiviert die Verwendung der +/- Syntax für Netzgruppen in Verbindung mit NIS
user	Darf nur im Schlüssel **printers** definiert werden und gibt an das des Users eigene Druckerkonfiguration benutzt werden soll. Der Benutzer benötigt dazu die $HOME/.printers
xfn	Darf nur im Schlüssel **printers** definiert werden und gibt an das die Drucker über das FNS gesucht werden sollen

[16] /etc/netgroup - Siehe Konfigurationsdateien Referenz Seite **??**
[17] /etc/inet/netmasks - Siehe Konfigurationsdateien Referenz Seite 421
[18] /etc/inet/networks - Siehe Konfigurationsdateien Referenz Seite **??**
[19] /etc/passwd - Siehe Konfigurationsdateien Referenz Seite **??**
[20] /etc/printers.conf - Siehe Konfigurationsdateien Referenz Seite **??**
[21] /etc/inet/protocols - Siehe Konfigurationsdateien Referenz Seite **??**
[22] /etc/rpc - Siehe Konfigurationsdateien Referenz Seite **??**
[23] /etc/inet/services - Siehe Konfigurationsdateien Referenz Seite 423
[24] /etc/resolv.conf - Siehe Konfigurationsdateien Referenz Seite 431

Beispiele

Quelltext 42.0.8 Beispiele für /etc/nsswitch.conf

```
hosts: files nis nisplus dns
passwd: files nis
printers: user nis xfn files nisplus

passwd: nis files # DON'T DO IT. VERY FATAL ERROR
group: dns # ERROR: dns only valid in hosts
```

Namendienstverfügbarkeit

Nein. Die /etc/nsswitch.conf muß local editiert werden. Dazu existieren einige Beispieldateien die man kopieren kann. Unter /etc/nsswitch.????? findet man die anderen Beispiele

Siehe auch

NIS, NIS+, LDAP, FNS

/etc/resolv.conf

Verwendungszweck

Die Datei /etc/resolv.conf ist für die Konfiguration des DNS Clients wichtig. In der Datei wird die DNS Domain und die DNS Server angegeben. Damit dieser auch genutzt werden kann muß in der /etc/nsswitch.conf[25] der Schlüssel **hosts** die Quelle *dns* beinhalten.

Aufbau und Format der Datei

- Die /etc/resolv.conf ist eine ASCII Textdatei
- Kommentare sind nicht erlaubt
- Leere Zeilen sind nicht erlaubt
- Als Spaltentrenner koennen mehrere Leerzeichen oder Tabulatoren verwendet werden

Spaltenbedeutungen

Tabelle 42.9: Spaltenbeschreibung: /etc/resolv.conf

Spalte	Bedeutung
1	Schlüssel
2	Wert

Als Schlüssel kommen folgende Werte in Frage :

Tabelle 42.10: Übersicht der Schlüssel in der /etc/resolv.conf

Schlüssel	Wert
nameserver	Angabe der IP Addresse eines DNS Servers. Maximal dürfen drei Nameserver angegeben werden. Es wird immer der erste gefragt. Hat der keine Antwort wird der zweite gefragt usw.
domainname	Gibt den DNS Domainnamen des DNS Clients an
searchlist	Mit diesem Schlüssel können mehrere DNS Domainnamen angegeben werden in dem der Hostname entsprechend gesucht werden soll.

[25] /etc/nsswitch.conf - Siehe Konfigurationsdateien Referenz Seite 427

Beispiele

Quelltext 42.0.9 Beispiele für /etc/resolv.conf

```
domainname hurst.pnet
nameserver 10.65.13.1
nameserver 10.65.13.4
searchlist hurst.pnet server.hurst.pnet
```

Namendienstverfügbarkeit

Nein.

Siehe auch

DNS

/etc/system

Verwendungszweck

Die Datei /etc/system wird fuer folgende Dienste benotigt :

Aufbau und Format der Datei

- Die /etc/system ist eine ASCII Textdatei
- Als Kommentarzeichen wird das Hash # verwendet
- Leere Zeilen sind nicht erlaubt
- Jede Zeile entspricht
- Als Spaltentrenner koennen mehrere Leerzeichen oder Tabulatoren verwendet werden

Spaltenbedeutungen

Tabelle 42.11: Spaltenbeschreibung: /etc/system

Spalte	Bedeutung
1	bla
2	bla
3	bla
4	bla

Beispiele

Quelltext 42.0.10 Beispiele für /etc/system

Siehe auch

/etc/ttydefs

Verwendungszweck

Die Datei /etc/ttydefs wird für den `ttyadm`[26] und den `tip`[27] verwendet. Und legt die Einstellungen zu einem Terminal fest.

Aufbau und Format der Datei

- Die /etc/ttydefs ist eine ASCII Textdatei
- Als Kommentarzeichen wird das Hash # verwendet
- Jede Zeile entspricht einem Terminaleintrag
- Als Spaltentrenner werden Doppelpunkte verwendet

Spaltenbedeutungen

Tabelle 42.12: Spaltenbeschreibung: /etc/ttydefs

Spalte	Bedeutung
1	Der *ttylabel* des Eintrages
2	Die Einstellungen für das Terminal (siehe `stty`[28])
3	Die gleichen Einstellungen nochmal (siehe `stty`)
4	Steht dort ein A dann handelt es sich um ein AutoBaud
5	Name des nächsten Eintrages aus dieser Gruppe

Beispiele

Quelltext 42.0.11 Beispiele für /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
....
300:300 hupcl:300 hupcl::460800
```

Siehe auch

ttyadm, stty, pmadm, sacadm

[26] `ttyadm` - Siehe Kommandoreferenz Seite 397
[27] `tip` - Siehe Kommandoreferenz Seite 393
[28] `stty` - Siehe Kommandoreferenz Seite 391

TEIL VI

ADVANCED TOPICS

Kapitel 43

ADVANCED PRINTER INTERFACES

Dieses Kapitel soll praktisch zeigen wie man ein universelles Interface für ein Drucker programmiert. Das Interface ist später in der Lage selbst das Dateiformat zu erkennen und entsprechend darauf zu reagieren.

43.1 Vorwort

43.1.1 Benötigte Software

Es wird einige zusatzsoftware benötigt die zum Teil im Solaris System bereits vorhanden ist, aber auch Software die extra aus dem SFW nachinstalliert werden muß, bzw. selbst übersetzt werden muß. Benötigt wird :

- Die Druckservices von Solaris.
- `gs`. GhostView aus dem Paket *SFWgs*.
- `a2ps`. Ascii-to-Postscript Konverter. Diese Software muß selbst übersetzt werden. Dazu muß man sich den Sourcecode besorgen und den laut der README übersetzen. Dafür wird jedoch `SFWgcc` als Compiler benötigt.
- Maildienste und die Bourne-Shell

43.1.2 Planung im groben

Ziel ist es ein Drucker im System zu definieren der mit fast allen Dateitypen umgehen kann. Das Interface muß den Quelltyp entsprechend in das druckbare Format umwandeln können. Das Interface wird ein von uns geschriebenes Shellscript sein.

43.2 Konfiguration des Druckers

Als erstes muß man sich einen Namen für den Drucker einfallen lassen. Ich werde hier den Namen `hp1200c` verwenden, da ich ein HP DeskJet 1200C habe.

43.2.1 Portkonfiguration

Zu erst muß die Schnittstelle an dem der Drucker sich befindet konfiguriert werden. Dazu müssen die Rechte `600` auf das Device dem User *lp* zugewiesen werden :

```
raven # chown lp /dev/ecpp0
raven # chmod 600 /dev/ecpp0
raven #
```

Bildschirmausschnitt 43.2.1: Geraeteberechtigungen

43.2.2 Dienstkonfiguration

Danach muß man feststellen ob der Druckerdaemon laeuft oder nicht. Bei bedarf muß er getartet werden.

```
raven # pgrep lp
240
raven #
```

Bildschirmausschnitt 43.2.2: Dienstcheck

43.2.3 Einrichtung mit `lpadmin`

So nun geht man hin und Konfiguriert einen Drucker mit dem `lpadmin`

```
raven # lpadmin -p hp1200c -v /dev/ecpp0
raven # lpadmin -p hp1200c -I any
raven # lpadmin -p hp1200c -T unknown
raven # lpadmin -p hp1200c -D "HP DeskJet 1200C (Master)"
raven # lpadmin -d hp1200c
raven # /usr/bin/enable hp1200c
printer "hp1200c" now enabled
raven # /usr/sbin/accept hp1200c
destination "hp1200c" now accepting requests
raven #
```

Bildschirmausschnitt 43.2.3: Druckerkonfiguration

43.2.4 Vorläufige Interfacekonfiguration

Da wir zur Zeit noch kein Interface haben geben wir dem `lpadmin` jetzt ersteinmal ein leeres Script welches wir dann entsprechend Editieren und erweitern. Da das Script sowieso kopiert wird in das Interfaceverzeichnis benötigen wir nur ein Fakescript

```
raven # touch /tmp/HP1200C.Interface
raven # lpadmin -p hp1200c -i /tmp/HP1200C.Interface
raven # rm /tmp/HP1200C.Interface
raven #
```

Bildschirmausschnitt 43.2.4: Fake Interface installieren

Das kopierte Script liegt jetzt unter /etc/lp/interfaces unter dem Namen des Druckers.

```
raven # cd /etc/lp/interfaces/
raven # ls
HP*  hp1200c*     hpmonops*    hpmonotext*
raven #
```

Bildschirmausschnitt 43.2.5: Interface Prüfung

43.3 Die Argumente an ein Interfaceprogramm

Ein Druckerinterface Programm bekommt immer ein paar Argumente. Sollte man in keiner Dokumentation der Welt die übergebenen Argumente finden, dann kann man im Interface vorerst einmal schauen was da kommt :

Quelltext 43.3.1 Beispiel Druckerinterface

```
 1     #!/bin/sh
 2     #
 3     # Druckerinterface fuer den HP Deskjet 1200C
 4     #
 5     # Argumente ausgeben in eine Tempdatei zur Ueberpruefung
 6     rm /tmp/x
 7     while test $# -ne 0; do
 8             echo "$1" >>/tmp/x
 9             shift
10   done
11   exit 0
```

In der `/tmp/x` Datei stehen nun die Argumente. Die folgende Tabelle hilft etwas dabei die einzelnen Argumente aufzulisten :

1	Name des Druckauftrages. Dieser wird durch den Namen des Druckers und durch die effektive Druckau
2	Username und Hostname von wo der Druckauftrag kommt. Achtung Solaris 7 trennt die beide
3	Druckauftragstitel (kann mit `lp`
4	
5	Optionen. Dort stehen bei Netzwe
6 ...	Datei

Um die ganzen Optionen im Überblick zu behalten werden wir diese Optionen ersteinmal in unseren eigenen Variablen aufnehmen :

Quelltext 43.3.2 Beispiel Druckerinterface (Variablen)

```
# Argumente in unsere Variablen umwandelt
requestid="$1"
title="$3"
copies="$4"
options="$5"

# Benutzernamen und Hostnamen filtern
case "$2" in
*!*)    # Trennung erfolgt ducht Ausrufezeichen
        # (Solaris 2.6 + 7 ... <hostname>!<username>
        username=‘echo "$2" | cut -d! -f2‘
        hostname=‘echo "$2" | cut -d! -f1‘
        ;;
*@*)    # Trennung erfolgt durch AT
        # (Solaris 8) ... <username>@<hostname>
        username=‘echo "$2" | cut -d@ -f1‘
        hostname=‘echo "$2" | cut -d@ -f2‘
        ;;
*)        # Unbekannt
        username="$2"
        hostname=""
        ;;
esac
```

43.4 Das Erstellen des Druckjobs Tempverzeichnisses

Dieses Tempverzeichnis soll am schluß noch nicht gelöscht werden um eventuelle Fehler des Scriptes im Betrieb besser zu analysieren. Man sucht sich erst einmal ein geeignetes Verzeichnis und setzt dessen Rechte gleich auf die richtigen Werte :

Quelltext 43.4.1 Beispiel Druckerinterface (Tempdir)

```
# Tempverzeichnis erstellen und Rechte setzen
tmp=/var/tmp/HP1200C/$requestid
mkdir -p -m 700 $tmp
umask 077
```

43.5 Fehlerkennung

Um später das Verzeichnis löschen zu können wird ein Fehlerkennzeichnungsverfahren benötigt :

Quelltext 43.5.1 Beispiel Druckerinterface (Tempdir)

```
# Fehlerindikator (Wegen Subshells - eine Datei) zuruecksetzen
rm -f $tmp/$filenumber.iserror
```

43.6 Dateischleife

Die ganzen Dateien müssen nun alle einzelnd abgearbeitet werden. Dazu emphiehlt sich das Shiften der Argumente um 5 um danach alle Dateien in der Schleife laufen zulassen. Deswegen mußten wir uns auch die Argumente selbst merken. Weiterhin benötigen wir ein einfachen Zähler um die einzelnen Dateien von einander zu unterscheiden später. Dazu wird einfach ein Zähler definiert, den man in der Scheife dann einfach hoch zählt.

Quelltext 43.6.1 Beispiel Druckerinterface (Tempdir)

```
# Dateianzahl zaehler
filenumber=0

# Schleife ... fuer alle Dateien
shift 5
for file in $@; do

        # Dateizaehler ein hochzaehlen
        filenumber=‘expr $filenumber + 1‘
```

43.6.1 Dateitypanalyse

Als nächstes muss der Dateityp geprüft werden. Wird eindateitype nicht erkannt soll der Benutzer darüber mittels einer Mail benachrichtigt werden. Es gibt jedoch noch eine Möglichkeit die jedoch nicht zu empfehlen wäre indem man nur Postscript abfängt und den Rest als Asciitext interpretiert. Das kann /und wird/ dann beim Drucken von GIF / ELF Dateien voll nachhinten losgehen.

Quelltext 43.6.2 Beispiel Druckerinterface (Tempdir)

```
        # Dateityp feststellen
        format="‘file $file‘ | cut -d: -f2"
        case "$format" in
```

ASCII Texte Konvertieren

Um Asciitexte in ein passendes Format zu pressen verwende ich zum einen das Programm `nl`[1] mit dem man Zeilennummer ausgegeben bekommt und danach das Programm `a2ps` um den ASCII Text in das richtige Postscriptformat zu bringen. Wird `a2ps` mit einem Errorlevel ungleich NULL vgerlassen wird der Fehlerindikator auf ungleich NULL gesetzt um ein Fehler zu markieren. Der Anwender bekommt eine entsprechende Mail zugesendet.

[1] `nl` - Siehe Kommandoreferenz Seite **??**

Quelltext 43.6.3 Beispiel Druckerinterface (Tempdir)

```
        *ascii*|*ASCII*|*Ascii*|*shell*         |*text*|*Text*)
                # ASCII Text erkannt
                nl -ba | /usr/local/bin/a2ps \
                  --output=- \
                  --portrait \
                  --medium=A4 \
                  --borders=no \
                  --left-footer= \
                  --footer= \
                    --right-footer= \
                  --header="Printed by $username from $hostname" \
                  --left-title="%e %T" \
                  --center-title="$file" \
                  --right-title="%Q" \
                   --margin=20 \
                  --pretty-print=off

                if test $? -ne 0; then
                        touch $tmp/iserror
                        mail $username@$hostname <<!
Subject: PrintJob $requestid failed

Der ASCII Text $file
konnte nicht richtig in eine Postscriptdatei von a2ps
umgewandelt werden. Ihr Druckauftrag befindet sich noch
im Verzeichnis $tmp
!
                        continue
                fi
                ;;
```

Wie man bemerkt wird hier kein Dateiname übergeben. Lassen Sie sich komplett überraschen was da noch so kommt.

Konvertierung von PDF Dateien

Um PDF Dateinen auszudrucken kann man `gs` verwenden. Der macht aus der PDF Datei eine Postscriptdatei.

Quelltext 43.6.4 Beispiel Druckerinterface (Tempdir)

```
        *PDF*)
                # PDF erkannt
                /opt/sfw/bin/gs \
                  -dNOPAUSE \
                  -sPAPERSIZE=a4 \
                  -sDEVICE=pswrite \
                  -sOutputFile=-

                if test $? -ne 0; then
                        touch $tmp/iserror
                        mail $username@$hostname <<!
Subject: PrintJob $requestid failed

Die PDF Datei $file
konnte nicht richtig in eine Postscriptdatei von gs
umgewandelt werden. Ihr Druckauftrag befindet sich noch
im Verzeichnis $tmp
!
                        continue
                fi
                ;;
```

Konvertierung von Postscript

Jetzt kommt kein Konverter sondern nur `cat`.

Quelltext 43.6.5 Beispiel Druckerinterface (Tempdir)

```
        *Postscript*|*postscript*)
                # Postscript erkannt
                cat
                ;;
```

Unbekannte Dateitypen

Wenn ein Dateityp unbekannter herkunft kommt soll der Benutzer eine Mail bekommen wo Ihm das gesagt wird.

Quelltext 43.6.6 Beispiel Druckerinterface (Tempdir)

```
        *)
                # Unbekanntes Dateiformat
                touch $tmp/iserror
                mail $username <<!
Subject: Printjob $requestid failed

Die Datei $file hat ein
unbekanntes Dateiformat : $format
!
                continue
                ;;
```

43.6.2 Ende der Dateianalyse

Jetzt wird der `case` Zweig geschlossen. Aber der Case-Zweig bekommt von und ein stdin, stdout und ein stderr verpasst. Somit wird alles innerhalb des Casezweiges umgelenkt. Somit benötigt kein einziges Programm den Originaldateinamen und die Konvertierten Daten werden aus dem Casezweig nach stdout geschrieben, wo wir diese Daten Umlenken in eine Postscript Druckvorstufendatei :

Quelltext 43.6.7 Beispiel Druckerinterface (Tempdir)

```
        esac <$file >$tmp/ps.$filenumber 2>$tmp/case-error.$filenumber
```

Da alles was auf dem Casezweig konvertiert wird immer Postscript ist, kann die Postscriptdatei theoretisch auf den Drucker geworfen werden. Jeider ist mir vor ein paar Wochen das Postscriptmodul des Druckers kaputt gegangen und deswegen :

43.6.3 Postscript konvertieren fuer den Drucker

Zum Glück wird der HP 1200C durch den HP PaintJet emuliert. Das habe ich aus der Dokumentation von `gs` erlesen können. Also wandle ich meine gerade erstellte Postscriptdatei um in PaintJet Format

Quelltext 43.6.8 Beispiel Druckerinterface (Tempdir)

```
        /opt/sfw/bin/gs \
          -dNOPAUSE \
          -sPAPERSIZE=a4 \
          -sDEVICE=pjxl300 \
          -r300 \
          -sOutputFile=$tmp/pcl.$filenumber \
          $tmp/ps.$filenumber >$tmp/gs-out.$filenumber 2>$tmp/gs-error.$filen

        if test $? -ne 0; then
                touch $tmp/iserror
                mail $username@$hostname <<!
Subject: PrintJob $requestid failed

Das Postscript Vorstufendruckresultat konnte nicht in die
Druckersprache durch gs uebersetzt werden.
!
                continue
        fi
```

43.6.4 N-mal drucken

Da die Ausgabe des Interfaces, spich stdout automatisch vom Druckdienst mit der Druckerwarteschlage verbunden wird, kann das Druckerformat nun ganznormal mit `cat` ausgegeben werden, das jedoch n-mal :

Quelltext 43.6.9 Beispiel Druckerinterface (Tempdir)

```
        x=0
        while test $x -lt $copies; do
                cat $tmp/pcl.$filenumber
                x=‘expr $x + 1‘
        done
```

43.6.5 Der Rest

Und danach wird noch die Schleife für die Dateien geschlossen und Aufgeräumt, sofern kein Fehler vorlag

Quelltext 43.6.10 Beispiel Druckerinterface (Tempdir)

```
        done

if test ! -f $tmp/iserror; then
# Keine Fehler aufgetreten, also loeschen des Tempverzeichnisses
rm -rf $tmp
fi

exit 0
```

43.7 Das komplette Interface

```
 1    #!/bin/sh
 2    #
 3    # Druckerinterface fuer den HP Deskjet 1200C
 4    #
 5
 6    # Argumente in unsere Variablen umwandelt
 7    requestid="$1"
 8    title="$3"
 9    copies="$4"
10  options="$5"
11
12  # Benutzernamen und Hostnamen filtern
13  case "$2" in
14  *!*)        # Trennung erfolgt ducht Ausrufezeichen
15          # (Solaris 2.6 + 7 ... <hostname>!<username>
16          username=‘echo "$2" | cut -d! -f2‘
17          hostname=‘echo "$2" | cut -d! -f1‘
18          ;;
19  *@*)        # Trennung erfolgt durch AT
20          # (Solaris 8) ... <username>@<hostname>
21          username=‘echo "$2" | cut -d@ -f1‘
22          hostname=‘echo "$2" | cut -d@ -f2‘
23          ;;
24  *)        # Unbekannt
25          username="$2"
26          hostname=""
27          ;;
28  esac
29
30  # Tempverzeichnis erstellen und Rechte setzen
31  tmp=/var/tmp/HP1200C/$requestid
32  mkdir -p -m 700 $tmp
33  umask 077
```

```
34
35   # Fehlerindikator (Wegen Subshells - eine Datei) zuruecksetzen
36   rm -f $tmp/$filenumber.iserror
37
38   # Dateianzahl zaehler
39   filenumber=0
40
41   # Schleife ... fuer alle Dateien
42   shift 5
43   for file in $@; do
44
45           # Dateizaehler ein hochzaehlen
46           filenumber=‘expr $filenumber + 1‘
47
48           # Dateityp feststellen
49           format="‘file $file‘ | cut -d: -f2"
50           case "$format" in
51           *ascii*|*ASCII*|*Ascii*|*shell*\
52           |*text*|*Text*)
53                   # ASCII Text erkannt
54                   nl -ba | /usr/local/bin/a2ps \
55                     --output=- \
56                     --portrait \
57                     --medium=A4 \
58                     --borders=no \
59                     --left-footer= \
60                     --footer= \
61                       --right-footer= \
62                     --header="Printed by $username from $hostname" \
63                     --left-title="%e %T" \
64                     --center-title="$file" \
65                     --right-title="%Q" \
66                      --margin=20 \
67                     --pretty-print=off
68
69                   if test $? -ne 0; then
70                           touch $tmp/iserror
71                           mail $username@$hostname <<!
72   Subject: PrintJob $requestid failed
73
74   Der ASCII Text $file
75   konnte nicht richtig in eine Postscriptdatei von a2ps
76   umgewandelt werden. Ihr Druckauftrag befindet sich noch
77   im Verzeichnis $tmp
78   !
79                           continue
80                   fi
```

```
 81                 ;;
 82
 83         *PDF*)
 84                 # PDF erkannt
 85                 /opt/sfw/bin/gs \
 86                   -dNOPAUSE \
 87                   -sPAPERSIZE=a4 \
 88                   -sDEVICE=pswrite \
 89                   -sOutputFile=-
 90
 91                 if test $? -ne 0; then
 92                         touch $tmp/iserror
 93                         mail $username@$hostname <<!
 94   Subject: PrintJob $requestid failed
 95
 96   Die PDF Datei $file
 97   konnte nicht richtig in eine Postscriptdatei von gs
 98   umgewandelt werden. Ihr Druckauftrag befindet sich noch
 99   im Verzeichnis $tmp
100   !
101                         continue
102                 fi
103                 ;;
104
105         *Postscript*|*postscript*)
106                 # Postscript erkannt
107                 cat
108                 ;;
109
110         *)
111                 # Unbekanntes Dateiformat
112                 touch $tmp/iserror
113                 mail $username <<!
114   Subject: Printjob $requestid failed
115
116   Die Datei $file hat ein
117   unbekanntes Dateiformat : $format
118   !
119                 continue
120                 ;;
121
122         esac <$file >$tmp/ps.$filenumber 2>$tmp/case-error.$filenumber
123
124         /opt/sfw/bin/gs \
125           -dNOPAUSE \
126           -sPAPERSIZE=a4 \
127           -sDEVICE=pjxl300 \
```

```
128             -r300 \
129             -sOutputFile=$tmp/pcl.$filenumber \
130             $tmp/ps.$filenumber >$tmp/gs-out.$filenumber 2>$tmp/gs-error.
131
132           if test $? -ne 0; then
133                   touch $tmp/iserror
134                   mail $username@$hostname <<!
135   Subject: PrintJob $requestid failed
136
137   Das Postscript Vorstufendruckresultat konnte nicht in die
138   Druckersprache durch gs uebersetzt werden.
139   !
140                   continue
141           fi
142
143           x=0
144           while test $x -lt $copies; do
145                   # cat $tmp/pcl.$filenumber
146                   x=‘expr $x + 1‘
147           done
148
149   done
150
151   if test ! -f $tmp/iserror; then
152           # Keine Fehler aufgetreten, also loeschen des Tempverzeichnisse
153           rm -rf $tmp
154   fi
155
156   exit 0
157
```

TABELLENVERZEICHNIS

1.1 Übersicht der verwendeten Beispielrechner . . . 3
4.1 Übersicht von Standardsignalen unter UNIX . . . 14
4.2 Standardfiledescriptoren eines Prozesses unter UNIX . . . 16
7.1 Übersicht der Fluchtzeichen der Bourne Shell . . . 30
8.1 Übersicht der Dateitypen Anzeigesymbole . . . 35
8.2 Übersicht der Auswirkungen der Lese-, Schreib- und Ausführrechte . . . 36
8.3 Übersicht der numerischen Schreibweise von Rechten . . . 39
9.1 Jobcontrol . . . 49
19.1 Slice Tabelle . . . 94
19.2 Eine Verzeichnissdatei als Tabelle . . . 100
19.3 Tools und Programme für Massenspeicherverwaltung . . . 102
20.1 Bedeutungen und Übersicht der Runlevel . . . 105
20.2 Übersicht der Runlevelverzeichnisse . . . 106
20.3 Tools und Programme zum Systemstart . . . 114
20.4 Config und Scripts zum Systemstart . . . 114
21.1 Tools und Programme zum SAF . . . 125
21.2 Config und Scripts zum Systemstart . . . 125
24.1 Befehle für Quotas . . . 139
27.1 cron und at . . . 157
34.1 NFS Operationen Version 2 und Version 3 . . . 189
34.2 Wiederholbare und nicht wiederholbare NFS Operationen . . . 190
34.3 Programme und Konfigurationsdateien für NFS . . . 199
41.2 Exitcodes von `bg` . . . 249
41.5 Exitcodes von `cfsadmin` . . . 256
41.10 Exitcodes von `crontab` . . . 265
41.14 Exitcodes von `fg` . . . 273
41.16 Exitcodes von `find` . . . 277
41.18 Exitcodes von `grep` . . . 281
41.20 Exitcodes von `halt` . . . 285
41.25 Exitcodes von `init` . . . 295
41.26 Exitcodes von `installboot` . . . 297
41.27 Exitcodes von `jobs` . . . 299
41.29 Exitcodes von `ln` . . . 303
41.35 Exitcodes von `ls` . . . 319
41.41 Exitcodes von `netstat` . . . 341
41.42 Exitcodes von `od` . . . 343
41.43 Exitcodes von `patchadd` . . . 345
41.44 Exitcodes von `patchrm` . . . 347
41.45 Exitcodes von `pkgadd` . . . 349
41.46 Exitcodes von `pkgchk` . . . 351
41.47 Exitcodes von `pkginfo` . . . 353
41.48 Exitcodes von `pkgrm` . . . 355

41.49Exitcodes von pmadm 359
41.50Exitcodes von poweroff 361
41.51Exitcodes von prtvtoc 363
41.56Exitcodes von reboot 373
41.59Exitcodes von sac 379
41.60Exitcodes von sacadm 383
41.62Exitcodes von shutdown 387
41.64Exitcodes von stty 391
41.65Exitcodes von tip 393
41.66Exitcodes von touch 395
41.67Spaltenbeschreibung: Ausgabe von ttyadm 398
41.68Exitcodes von ttyadm 399
41.72Exitcodes von which 407
42.1 Spaltenbeschreibung: crontab 415
42.2 Spaltenbeschreibung: /etc/inet/hosts 419
42.3 Spaltenbeschreibung: /etc/inet/netmasks 421
42.4 Spaltenbeschreibung: /etc/inet/services 423
42.5 Spaltenbeschreibung: /etc/inittab 425
42.6 Spaltenbeschreibung: /etc/nsswitch.conf 427
42.7 Übersicht der Datenbankschlüssel in der /etc/nsswitch.conf 427
42.8 Übersicht der Namensdienstquellen in der /etc/nsswitch.conf 428
42.9 Spaltenbeschreibung: /etc/resolv.conf 431
42.10Übersicht der Schlüssel in der /etc/resolv.conf 431
42.11Spaltenbeschreibung: /etc/system 433
42.12Spaltenbeschreibung: /etc/ttydefs 435

ABBILDUNGSVERZEICHNIS

8.1 Grundaufbau des Modewortes . . . 34
8.2 Binärkodierung der Dateitypen im Modewort . . . 35
8.3 Binärkodierung der Rechte im Modewort . . . 36
17.1 b . . . 79
17.2 b . . . 80
17.3 b . . . 80
17.4 b . . . 81
17.5 b . . . 81
17.6 b . . . 82
17.7 b . . . 82
17.8 b . . . 83
17.9 b . . . 83
17.10b . . . 84
17.11b . . . 84
17.12b . . . 85
17.13b . . . 85
17.14b . . . 86
18.1 OBP: Der Banner einer Ultra-10 . . . 89
19.1 Einteilung eines Laufwerkes in Slices . . . 94
19.2 Aufbau des UFS Dateiensystems . . . 95
19.3 Basis Aufbau einer INode . . . 97
19.4 Direkte Datenblockaddressen . . . 98
19.5 Blockaddresse 11 . . . 99
19.6 Blockaddresse 12 . . . 99
19.7 Blockaddresse 13 . . . 100
19.8 Verzeichnisstruktur nach der Erstellung von subdir . . . 101
19.9 Ein Hardlink im Detail . . . 101
20.1 Allgemeiner Bootvorgang . . . 103
20.2 Die Links . . . 107
21.1 Grundlegenderaufbau von SAF . . . 119
21.2 Konfiguration des EAGLE als Nullmodem Server . . . 122
21.3 Der Portmonitor `ttymon` im Detail . . . 125
22.1 Group . . . 129
22.2 Group . . . 129
22.3 Group . . . 129
22.4 Group . . . 130
23.1 Rechteprüfung mit ACL . . . 132
25.1 Funktionsweise des SystemV Drucksystems (Einfach) . . . 141
25.2 Kontrolle behalten . . . 143
31.1 Hinzufügen eines Remoteprinters mit dem `admintool` . . . 182

BILDSCHIRMAUSSCHNITTSVERZEICHNIS

5.2.1 Beispiele von Prompts 19
7.3.1 Teilung in Argumente Beispiel 1 26
7.3.2 Teilung in Argumente Beispiel 2 26
7.3.3 Auf der Flucht mit einem Backslash 27
7.3.4 Auf der Flucht mit einfachen Anführungsstrichen 27
7.3.5 Auf der Flucht mit doppelten Anführungszeichen 27
7.3.6 Beispielverzeichnis für Wildcards 28
7.3.7 Beispiele zum Metazeichen Stern 29
7.3.8 Metazeichen Stern ohne Auflösung 29
9.1.1 Anwendungsbeispiel für Hintergrundprozesse 45
9.1.2 Anwendungsbeispiel für NOHUP Hintergrundprozesse 46
9.2.1 Anwendungsbeispiel zum Stoppen 47
9.3.1 Anwendungsbeispiele zur Kontrolle 48
10.1.1 Einfaches Pipe Beispiel mit `head` und einem Dateinamen 51
10.1.2 Einfaches Pipe Beispiel mit `head` und ohne einem Dateinamen 52
10.1.3 Einfaches Pipe Beispiel mit `head` und `cat` 53
10.1.4 Einfaches Pipe Beispiel mit einer doppeldeutigen Bedeutung 53
12.1.1 Beispiel einer einfachen Wertzuweisung an eine Environmentvariable 59
12.1.2 Beispiel einer Wertzuweisung mit Leerzeichen an eine Environmentvariable 59
12.1.3 Beispiel der Verwendung von Variableninhalten 60
12.1.4 Beispiel der Verwendung von Variableninhalten mit Leerzeichen 60
12.1.5 Variablengültigkeitsbereich: Lokale Shell 61
12.1.6 Variablengültigkeitsbereich: Lokale Shell mit Überschreiben 61
12.1.7 Variablengültigkeitsbereich: Exportierte Variablen 61
12.1.8 Variablengültigkeitsbereich: Exportvererbung 62
13.1.1 Die erste Funktion 65
13.2.1 Funktion mit Argument 66
13.2.2 Funktion mit Argumenten 66
13.3.1 Umdefinierung von `ls` 66
13.3.2 Umdefinierung von `ls` mit variablen Argumenten 67
14.4.1 Beispiel Ausgabe von `infome` 71
18.2.1 OBP: Gerätemanagement 90
18.2.2 OBP: Booten von Platte 91
18.2.3 OBP: Booten von Platte mit Aliasnamen 91
18.2.4 OBP: Aliasnamen vereinbaren 91
18.2.5 OBP: Anzeigen und setzten von Variablen 92
20.1.1 Anzeigen des aktuellen Runlevels 106
21.7.1 Verbindungsaufbau durch Nullmodemkabel 123
23.2.1 Anzeigen und Übertragen von ACL's 134
24.5.1 Erstellen der Quota Kontrolldatei 136
24.6.1 Quotas für ein Benutzer 137
24.7.1 Kopieren von Quotas 137

24.8.1 Einstellen des Standard Zeitzählers . . . 137
24.9.1 Ausgabe von `quota` bei Softlimit ausnutzern . . . 138
24.9.2 Ausgabe von `quota -v` bei Benutzern . . . 138
24.9.3 Ausgabe von `quota -v` bei `otto` . . . 138
24.9.4 Ausgabe von `repquota -a` . . . 138
25.2.1 Einstellen der Deviceberechtigungen . . . 144
25.2.2 Starten des Druckdienst, sofern dieser nicht schon läuft . . . 145
25.2.3 Einrichten des Druckers via lpadmin . . . 145
25.2.4 Freischalten des Druckers . . . 145
25.3.1 Konfiguration eines nicht Postscriptdruckers . . . 146
27.1.1 Einstellen des Editors . . . 155
28.3.1 Auskünfte von Paketen einholen . . . 160
28.3.2 Suche nach Paketen . . . 161
28.4.1 Überprüfen von Paketen . . . 161
28.5.1 Suchen von Dateien zu einem Paket . . . 162
28.5.2 Entfernen von Paketen . . . 162
29.1.1 Liste der netzfähigen Devices . . . 171
29.1.2 Deviceaktivierung für Netz . . . 171
29.1.3 Temporäre IP Konfiguration . . . 172
29.1.4 Temporäre IP Konfiguration mit Hostnamen . . . 172
29.1.5 Temporäre Netzmasken Konfiguration . . . 172
29.1.6 Temporäre Broadcastaddressen . . . 173
29.1.7 Temporäre Aktivierung des Interfaces . . . 173
29.2.1 Temporäres hinzufügen von Routen . . . 174
29.2.2 Anzeigen von Routingeinträgen . . . 174
29.2.3 Löschen von Routingeinträgen . . . 175
31.3.1 Einrichting des Druckclients LX . . . 181
34.3.1 Beispiele von `share` . . . 191
34.3.2 Beispiele von `share` . . . 192
34.3.3 Beispiele von `unshare` . . . 192
34.3.4 Beispiele von `unshare` . . . 193
34.4.1 Mounten von Shares . . . 194
38.2.1 Erstellen eines CacheFS . . . 224
38.2.2 Anzeigen der CacheFS Einstellungen . . . 224
38.3.1 Erstellen eines CacheFS . . . 225
41.0.1 Beispiel: `netstat` . . . 333
41.0.2 Beispiel: `netstat -g` . . . 334
41.0.3 Beispiel: `netstat -i` . . . 334
41.0.4 Beispiel: `netstat -p` . . . 335
41.0.5 Beispiel: `netstat -r` . . . 336
41.0.6 Beispiel: `netstat -s` Teil 1 . . . 337
41.0.7 Beispiel: `netstat -s` Teil 2 . . . 337
41.0.8 Beispiel: `netstat -s` Teil 3 . . . 338
41.0.9 Beispiel: `netstat -s` Teil 4 . . . 339
41.0.10 Beispiel: `netstat -s` Teil 5 . . . 339
41.0.11 Beispiel: `netstat -s` Teil 6 . . . 340
41.0.12 Beispiel: `netstat -s` Teil 7 . . . 340

41.0.1 Beispiel: `netstat -s` Teil 8 341
43.2.1 Geraeteberechtigungen 439
43.2.2 Dienstcheck 440
43.2.3 Druckerkonfiguration 440
43.2.4 Fake Interface installieren 440
43.2.5 Interface Prüfung 440

QUELLTEXTVERZEICHNIS

2.2.1 Conditional Execution Example 1 6
2.2.2 Conditional Execution Example 2 6
2.3.1 Superscalar Example 1 7
2.3.2 Superscalar Example 2 7
2.3.3 Speculative Computation Example 1 8
12.2.1 Beispiel einer .profile 64
14.3.1 Beispielscript `helloworld` 69
14.3.2 Beispielscript `helloworld2` mit Variablen 70
14.3.3 Beispielscript `lsubin` 70
20.1.1 Muster Startmodul 104
20.1.2 Dokumentiertes `/sbin/rc2` *(Teil 1)* 108
20.1.3 Dokumentiertes `/sbin/rc2` *(Teil 2)* 109
20.1.4 Dokumentiertes `/sbin/rc2` *(Teil 3)* 110
20.1.5 Dokumentiertes `/sbin/rc2` *(Teil 4)* 110
20.1.6 Dokumentiertes `/sbin/rc2` *(Teil 5)* 111
20.1.7 Dokumentiertes `/sbin/rc2` *(Teil 6)* 111
20.1.8 Dokumentiertes `/sbin/rc2` *(Teil 7)* 111
20.5.1 Original /etc/inittab von Solaris 113
20.7.1 Lösungsansatz für runlevel (Lösung 1) 116
20.7.2 Lösungsansatz für runlevel (Lösung 2) 116
20.7.3 Lösungsansatz für runlevel (Lösung 3) 116
20.7.4 Lösungsansatz für runprocess (Lösung 1) 116
20.7.5 Lösungsansatz für runprocess (Lösung 2) 117
20.7.6 Lösungsansatz für Startmodul rwho 117
21.2.1 Eintragungen in der /etc/inittab für `sac` 120
21.3.1 Beispiel Konfiguration eines Portmonitors 121
21.4.1 Beispiel Konfiguration einer Portmonitor-Service Verbindung 121
21.7.1 Beispieleintragung des Hosts bei Nullmodem 122
21.8.1 Beispieleintragung des Hosts bei Modemwahl 123
24.5.1 Mountpointeintrag mit Quota in der /etc/vfstab 136
27.1.1 Beispieleinträge einer crontab 156
28.10.1 pkgfind 167
28.10.2 pkgfiles 168
34.3.1 Beispiel einer /etc/dfs/dfstab 191
41.0.1 Beispiel von `logname` in Shellscripts 305
42.0.2 Beispiele für crontab 415
42.0.3 Beispiele für /etc/default/init 417
42.0.4 Beispiele für /etc/inet/hosts 419
42.0.5 Beispiele für /etc/inet/netmasks 421
42.0.6 Beispiele für /etc/inet/services 423
42.0.7 Beispiele für /etc/inittab 426
42.0.8 Beispiele für /etc/nsswitch.conf 429

42.0.9 Beispiele für /etc/resolv.conf 432
42.0.10 Beispiele für /etc/system 433
42.0.11 Beispiele für /etc/ttydefs 435
43.3.1 Beispiel Druckerinterface 441
43.3.2 Beispiel Druckerinterface (Variablen) 442
43.4.1 Beispiel Druckerinterface (Tempdir) 442
43.5.1 Beispiel Druckerinterface (Tempdir) 443
43.6.1 Beispiel Druckerinterface (Tempdir) 443
43.6.2 Beispiel Druckerinterface (Tempdir) 443
43.6.3 Beispiel Druckerinterface (Tempdir) 444
43.6.4 Beispiel Druckerinterface (Tempdir) 445
43.6.5 Beispiel Druckerinterface (Tempdir) 445
43.6.6 Beispiel Druckerinterface (Tempdir) 446
43.6.7 Beispiel Druckerinterface (Tempdir) 446
43.6.8 Beispiel Druckerinterface (Tempdir) 447
43.6.9 Beispiel Druckerinterface (Tempdir) 447
43.6.10 Beispiel Druckerinterface (Tempdir) 448

ABKÜRZUNGSVERZEICHNIS

ABI . . . . . Application Binary Interface(159)
ACL . . . . . Access Control List(131)
CISC . . . . . Complex Instuction Set Computer(5)
cpi . . . . . Character Per Inches(308)
DFS . . . . . Distributed File System(187)
EOF . . . . . End of File(52)
EPIC . . . . . Explicitly Parallel Instruction Computing(8)
fd . . . . . Filedescriptor(15)
IFS . . . . . Internal Field Separator(62)
IRB . . . . . Instruction Record Buffer(7)
LHS . . . . . Left Hand Side(51)
LUN . . . . . Logical Unit Number(93)
NFS . . . . . Network File System(187)
OBP . . . . . OpenBoot PROM(87)
RFS . . . . . Remote File System(187)
RHS . . . . . Right Hand Side(51)
RISC . . . . . Reduced Instruction Set Computer(5)
SAC . . . . . Service Access Controller(119)
SAF . . . . . Service Access Facility(119)
TCO . . . . . Total Cost of Ownership(32)
UPA . . . . . Ultra Port Architecture(8)
VTOC . . . . . Volume Table Of Contents(95)

REQUEST FOR COMMENTS

Assigned Numbers . . . . . 1700(423)
Classless Inter-Domain Routing (CIDR) . . . . . 1519(422)
DoD Internet Host Table Specification . . . . . 952(420)
Domain Name System Implementation Schedule . . . . . 921(420)
Internet Standard Subnetting Procedure . . . . . 950(422)
Requirements for Internet Hosts . . . . . 1123(420)

INDEX

/etc/default/init
Referenz, 417
/etc/inet/hosts
Referenz, 419
/etc/inet/netmasks
Referenz, 421
/etc/inet/services
Referenz, 423
/etc/inittab
Referenz, 425
/etc/nsswitch.conf
Referenz, 427
/etc/resolv.conf
Referenz, 431
/etc/system
Referenz, 433
/etc/ttydefs
Referenz, 435

Abkuerzungen
ABI, 159
ACL, 131
CISC, 5
cpi, 308
DFS, 187
EOF, 52
EPIC, 8
fd, 15
IFS, 62
IRB, 7
LHS, 51
LUN, 93
NFS, 187
OBP, 87
RFS, 187
RHS, 51
RISC, 5

SAC, 119
SAF, 119
TCO, 32
UPA, 8
VTOC, 95

arch
Referenz, 247

bg
Referenz, 249

Broken Pipe, 53

cat
Referenz, 251

cd
Referenz, 253

cfsadmin
Referenz, 255

chgrp
Referenz, 257

chmod
Referenz, 259

chown
Referenz, 261

cp
Referenz, 263

crontab
Referenz, 265, 415

df
Referenz, 267

edquota
Referenz, 269

export
Referenz, 271

fg
Referenz, 273

file
Referenz, 275

find
Referenz, 277
getfacl
Referenz, 279
grep
Referenz, 281
groups
Referenz, 283
halt
Referenz, 285
head
Referenz, 287
hostname
Referenz, 289
id
Referenz, 291
infocmp
Referenz, 293
init
Referenz, 295
installboot
Referenz, 297
jobs
Referenz, 299
kill
Referenz, 301
Landeseinstellung, 112
ln
Referenz, 303
logname
Referenz, 305
lp
Referenz, 307
lpadmin
Referenz, 311
lpsched
Referenz, 315

lpstat
Referenz, 317
ls
Referenz, 319
mail
Referenz, 321
man
Referenz, 323
mkdir
Referenz, 327
more
Referenz, 329
mv
Referenz, 331
netstat
Referenz, 333
NFS Operationen, 188
od
Referenz, 343
patchadd
Referenz, 345
patchrm
Referenz, 347
pkgadd
Referenz, 349
pkgchk
Referenz, 351
pkginfo
Referenz, 353
pkgrm
Referenz, 355
pmadm
Referenz, 357
poweroff
Referenz, 361
prtvtoc
Referenz, 363

pwd
Referenz, 365

quotacheck
Referenz, 367

quotaoff
Referenz, 369

quotaon
Referenz, 371

reboot
Referenz, 373

Request for Comments
RFC-1123, 420
RFC-1519, 422
RFC-1700, 423
RFC-921, 420
RFC-950, 422
RFC-952, 420

rm
Referenz, 375

rmdir
Referenz, 377

sac
Referenz, 379

sacadm
Referenz, 381

setfacl
Referenz, 385

shutdown
Referenz, 387

sleep
Referenz, 389

stty
Referenz, 391

tip
Referenz, 393

touch
Referenz, 395

ttyadm
Referenz, 397
type
Referenz, 401
umask
Referenz, 403
uname
Referenz, 405
which
Referenz, 407
who
Referenz, 409
write
Referenz, 411
Zeitzone, 112