



## Einführung in die Systemverwaltung unter UNIX

Dr. Ernst Bötsch / Dr. Petra Einfeld  
**5. Überarbeitete Auflage ( September 2005 )**

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

## Themengebiete: Übersicht



- Umfeld und allgemeine Aufgaben
- Stand-alone-Betrieb**
- Netzaspekte
- Security-Grundlagen
- Workshop

## Stand-alone-Betrieb: Übersicht



- Manualsystem
- Dateibaum, Filesysteme, Plattenorganisation
- Prozesse, Dämonen
- Benutzerverwaltung
- Datenhaltung
- Datensicherung
- Betrieb von ASCII-Terminals
- Spooling
- Ablauf des Boot-Vorgangs
- Wichtige Kommandos

Einführung in die Systemverwaltung unter UNIX

3

## Manualsystem : Übersicht



- Grundlagen
- Aufbau des Manualsystems
- Aufbau der einzelnen man-Pages
- Sonstiges

Einführung in die Systemverwaltung unter UNIX

4

## Manualseystem : Grundlagen (1)



### Zweck :

- Beschreibung der Kommandos und Optionen; sollte in der Regel online verfügbar sein

Aufruf: `man [-s section ] kommando` ⇔ "man-Page"

### Aufbau :

- Gliederung in (normalerweise 8) Einzelmanuale ("Sections")
- Thematik und Numerierung stimmt bei den meisten gängigen UNIX-Systemen überein

## Manualseystem : Grundlagen (2)



### Aufruf :

- Einzelmanuale werden *der Reihe nach* durchsucht
- Reihenfolge beachten !  
z.B. `man (1)` ⇔ `man (5)`

### Tipp :

- 1 vollständiger *gedruckter* Satz (der wichtigsten Dokumente)
- Mindestanforderung :
  - Command Reference (d.h. die man-Pages)
  - Administrator Guide(s)

## Manualseystem: Aufbau (SysV.4)



| <u>Band</u>                 | <u>Inhalt</u>                    | <u>Bsp.:</u> |
|-----------------------------|----------------------------------|--------------|
| (1) User Commands           | ⇒ Benutzer                       | man          |
| (1M) Maintenance Commands   | ⇒ Administrator                  | fsck         |
| (2) System Calls            | ⇒ Library-Funktionen (Schale)    | write        |
| (3) C-Library Functions     | ⇒ komfortabler ( "Meta-Schale" ) | printf       |
| (4) File Formats            | ⇒ Konfigurations-Dateien         | vfstab       |
| (5) Headers, Tables, Macros | ⇒ ASCII-Code-Tabelle             | man          |
| (6) Games and Demos         | ⇒ ☺                              |              |
| (7) Special Files           | ⇒ Treiber, Interfaces etc.       | console      |

Einführung in die Systemverwaltung unter UNIX

7

## Manualseystem: Aufbau (BSD)



| <u>Band</u>                                      | <u>Inhalt</u>                  |
|--|--------------------------------|
| (1) User Commands and Application Programs       | ⇒ Benutzer                     |
| (2) System Calls                                 | ⇒ Betriebssystem-Aufrufe       |
| (3) Library Routines                             | ⇒ z.B. für Programmiersprachen |
| (4) Device Drivers                               | ⇒ z.B. für Drucker             |
| (5) File Formats                                 | ⇒ Konfigurations-Dateien       |
| (6) Games and Demos                              | ⇒ ☺                            |
| (7) Miscellaneous:<br>ASCII, Macro Packages etc. |                                |
| (8) Commands for System Administration           | ⇒ Administratoren              |

Einführung in die Systemverwaltung unter UNIX

8

## Man-Pages: Aufbau



| <u>Abschnitt</u>                             | <u>Inhalt</u>                                     |
|--|---|
| <input type="checkbox"/> NAME                | ⇒ <u>Bsp.</u> : ftp                               |
| <input type="checkbox"/> SYNOPSIS            | ⇒ Syntax  |
| <input type="checkbox"/> DESCRIPTION         | ⇒ Kurz-Info                                       |
| <input type="checkbox"/> OPTIONS             | ⇒ mögliche Optionen                               |
| <input type="checkbox"/> COMMANDS            | ⇒ ftp-Sub-Kommandos, <u>z.B.</u> : open, get, put |
| <input type="checkbox"/> RETURN VALUES       | ⇒ (Miß-)Erfolg bei Library-Funktionen             |
| <input type="checkbox"/> ERRORS              | ⇒ Error-Codes                                     |
| <input type="checkbox"/> EXIT STATUS         | ⇒ (Miß-)Erfolg bei Kommandos                      |
| <input type="checkbox"/> FILES               | ⇒ benötigte Dateien                               |
| <input checked="" type="checkbox"/> SEE ALSO | ⇒ <b>wichtig !</b> Querverweise                   |
| <input type="checkbox"/> DIAGNOSTICS         | ⇒ Fehlermeldungen (von Kommandos etc.)            |
| <input type="checkbox"/> BUGS                | ⇒ bekannte Fehler / Probleme                      |
| <input type="checkbox"/> [ div. andere ]     |   |

Einführung in die Systemverwaltung unter UNIX

9

## Manualseite: Hinweise (1)



### Wichtige Kommandos:

- `man -k` ⇒ Suche nach Keywords
- `apropos (1)` ⇒ entspricht "`man -k`"
- `man (1)` ⇒ Infos zum Kommando `man` selbst
- `whatis (1)` ⇒ 1-zeilige Kurzbeschreibung
- `man (5)` ⇒ Infos zu `man`- Makros und Struktur einer `man`-Page
- `catman (8)` ⇒ `man`-Pages für die Ausgabe auf Terminals aufbereiten und Datenbasis aller Kurzbeschreibungen erzeugen

Einführung in die Systemverwaltung unter UNIX

10

## Manualsystem : Hinweise (2)



### Wichtige Verzeichnisse :

- /usr/man[1-8ln]/\*      ⇒ nroff / troff-Quellen
- /usr/man/cat[1-8ln]    ⇒ für Terminal-Ausgabe aufbereitet
- /usr/man/whatis        ⇒ Datenbasis mit Kurzbeschreibungen
- /usr/man/windex        ⇒ Datenbasis mit Kurzbeschreibungen  
( nur bei Solaris )

## Filesysteme : Übersicht



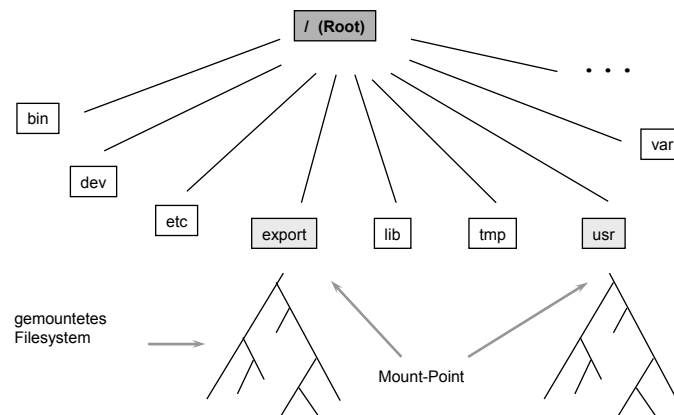
- Dateibaum
- Filesystem
- Realisierungsformen
- Plattenorganisation
- Files, Inode

# Dateibaum



- Darstellung als kopfstehender Baum
- Die Wurzel des Baumes wird mit „ / “ symbolisiert und „Root“ genannt.
- Beliebige Bewegung innerhalb des Baumes möglich (sofern Rechte vorhanden)
- Verlassen des Baumes nicht möglich
- Der Dateibaum setzt sich aus kleineren Einheiten zusammen:
  - Filesysteme
  - Verzeichnisse (Directories)
  - Unterverzeichnisse (Sub-Directories)
  - "eigentliche" Dateien (Files)

# Dateibaum: Typischer Aufbau



## Mount-Point



- ❑ Dienen zum „Einhängen“ von Filesystemen in den Dateibaum.
- ❑ Werden bei der Auflösung von Pfadnamen als Wurzel des betreffenden Filesystems betrachtet.
- ❑ Achtung: Mount-Point-Directories sollten leer sein !  
⇒ Andernfalls „Verschattung“ möglich !
- ❑ Achtung: Die Rechte des Mount-Point-Directories *vor* dem Mount sollten identisch mit der Wurzel des Filesystems *nach* dem Mount sein !  
⇒ Andernfalls „misteriöse“ Effekte möglich !

## Filesystem: Begriff

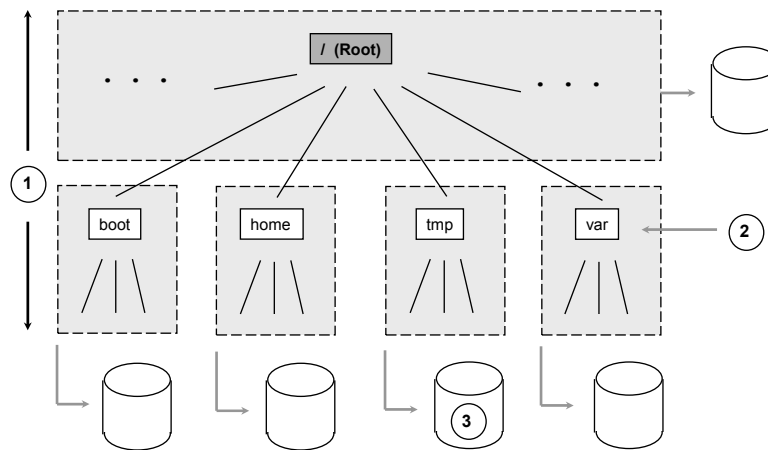


Filesystem — ein Begriff mit verschiedenen Bedeutungen:

- (1) Gesamter Dateibaum  
(von der Wurzel an abwärts; Benutzersicht)
- (2) Gemounteter Teilbereich (Administratorsicht);  
Anbindung einer Datenstruktur auf einem Datenträger
- (3) Abgeschlossene Struktur auf einem Datenträger  
(konkrete Daten)
- (4) Bestimmte Realisierungsvarianten (Syntax, Struktur; abstrakt)

Die konkrete Bedeutung ergibt sich meist aus dem Kontext.

## Filesystem: Beispiel



Einführung in die Systemverwaltung unter UNIX

17

## Filesysteme : Realisierungsformen (1)



### Disk-basierte Filesysteme

- Platte bzw. Floppy : UNIX-FS (UFS; Berkeley) oder SysV-FS
- CD : High-Sierra-FS (HSFS) oder ISO-9660-FS
- Floppy : PC-FS (PCFS; DOS)

### Netz-basierte Filesysteme

- Network Filesystem (NFS)
- Andrew Filesystem (AFS)
- Distributed File-Service (DFS; OFS / DCE)

Einführung in die Systemverwaltung unter UNIX

18

## Filesysteme : Realisierungsformen (2)



### Pseudo-Filesysteme

- Temporary FS  
( TMPFS; in Memory und Swap; entspricht RAM-Disk bei PC's )
- Loopback FS ( LOFS )
- Process FS ( PROCFS )
- File Descriptor FS ( FDFS )
- Cache FS ( CacheFS )
- Character- und Block-Devices ( "Special Files" )
- Named Pipes
- Swap

Einführung in die Systemverwaltung unter UNIX

19

## Partitionierung (1)



- Aufteilung einer physischen Platte in mehrere logische Platten  
(⇒ "normale" Partition)
- Zusammenfassung mehrerer physischer Platten zu einer  
logischen Platte (⇒ Meta-Partition; Volume-Management)
- Pro Partition maximal 1 Filesystem

### Motivation :

- unterschiedliche Verwendungsarten der Partitionen:  
Unterschiedliche Filesysteme (ext2, ext3), Swap, "Raw Data"
- garantierte Plattenplatzbegrenzung  
( z.B. für /var ⇒ Logfiles )
- Bereich mit garantiertem Plattenplatz, in den hinein sich andere  
Bereiche nicht ausdehnen können ( z.B. Root-Partition )

Einführung in die Systemverwaltung unter UNIX

20

## Partitionierung (2)



### "Kontingentierung der 2. Stufe" für Benutzer-Daten :

- eigentlich ein "Mißbrauch" der Partitions-Eigenschaften  
( Das Betriebssystem erlaubt leider keine Gruppen-kontingente. )
- individuelle* Home-Directory-Partition für *jede Benutzergruppe*
  - ⇒ bei der "Kontingentierung der 1. Stufe" (Benutzer-Quotas)  
kann überplant werdenVoraussetzung: nicht *alle* Benutzer nutzen *gleichzeitig* ihr Quota
- Die Benutzergruppe *als Gesamtheit* kann trotzdem nur einen bestimmten nicht überschreitbaren Plattenplatz belegen.
- interne Koordination zwingend erforderlich
  - ⇒ funktioniert nur bei kleinen homogenen Benutzergruppen

## Partitionierung: Allgemeine Regeln



- Bereiche mit System- und Benutzerdaten in unterschiedlichen Filesystemen
  - ⇒ einfacherer und sicherer Betriebssystem-Update
- "Statische" und "dynamische" Systembereiche in unterschiedlichen Filesystemen
  - ⇒ UNIX nimmt ein volles /-Filesystem extrem übel !
- Ausreichende Reserven bei den Systembereichen
  - ⇒ Platz für neue Software-Pakete
- Evtl. zu Beginn die Platte nicht komplett verplanen.

## Beispielkonfiguration für Linux



### Ab einer Plattengröße von ca. 40 GB

- Swap: 1.1 - 2 mal so groß wie das Memory, primary Partition
- /boot: 64 MB, primary Partition, ext2-Filesystem
- /tmp: 1 - 2 GB, ext3-Filesystem
- /var: 1 - 2 GB, ext3-Filesystem
- /usr/tmp: Soft-Link auf /var/tmp
- /: 4 GB, ext3-Filesystem, mindestens 1 - 2 GB noch frei
- Unbelegt: Evtl. 5 - 10 GB frei lassen (z.B. für ein Windows-System)
- /home: Rest der Platte, ext3-Filesystem

Einführung in die Systemverwaltung unter UNIX

23

## Filesysteme: Kommandos (1)



- mkfs (8) ⇒ Anlegen
- fsck (8) ⇒ Überprüfen und reparieren
- mount / umount (8) ⇒ Einhängen / abhängen
- df (1) ⇒ Infos über gemountete Filesysteme
- fstab (5) / vfstab (4) ⇒ Mount-Konfiguration (muss vom Admin. gepflegt werden)
- mnttab (4) ⇒ Liste aktuell gemounteter Filesysteme (wird automatisch angelegt)
- automount (8) ⇒ Automat., dynamisches [u]mount
- mknod (1) ⇒ Special-File anlegen
- swapon / off (8) ⇒ Administration von Swap-Bereichen

Einführung in die Systemverwaltung unter UNIX

24

## Filesysteme: Kommandos (2)



### Solaris

- `newfs (1M)`           ⇒ Benutzerfreundliche Schale um `mkfs`
- `fstyp (1M)`           ⇒ Ausgabe von Filesystem-Attributen
- `fsirand (1M)`       ⇒ Sicherheits-Feature für NFS
- `mkfile (1M)`       ⇒ Leeren File anlegen
- `swap (1M)`           ⇒ Administration von Swap-Bereichen

## File-Typen



- Regular (plain) Files       ⇒ "normale" Dateien ( Daten, Texte, Source-Code, Executables etc.)
- Directories               ⇒ (Unter-)Verzeichnisse
- Character Device Files   ⇒ z.B. Terminal-Treiber, Platten
- Block Device Files       ⇒ z.B. Platten- und Band-Laufwerke
- UNIX-Domain Sockets     ⇒ ermöglichen raschere Prozeß-Kommunikation über Netze  
(entwickelt für BSD)
- Named Pipes              ⇒ Prozeß-Kommunikation innerhalb  
(entwickelt von AT&T)   eines Rechners
- Hard-Links               ⇒ "Alias" durch Verweis auf Inode
- Soft-Links (Symbolic Links) ⇒ "Alias" durch Verweis auf Pfad

## Inodes : Allgemeines



### Inode: Aufbau

- ❑ Enthält alle Verwaltungsinformationen für Dateien (außer Name).  
Ca. 40 Einzeldaten; Beispiele:
  - Inode-Nr. (eindeutig innerhalb des Filesystems)
  - File-Typ
  - Owner, Group, Permission
  - diverse Zeitstempel
  - Länge
  - Link-Count
  - Pointer auf Datenblöcke

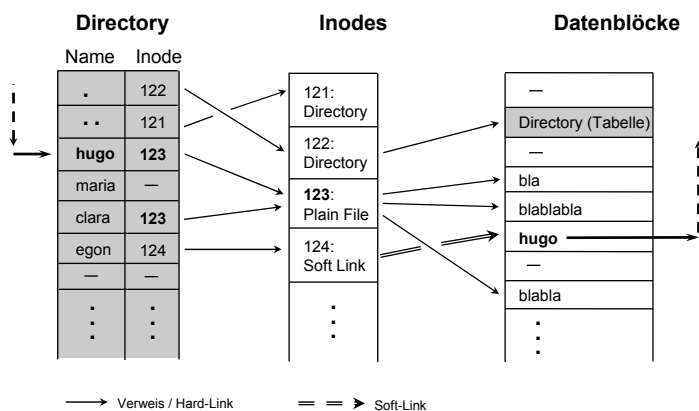
### Directory: Aufbau

- ❑ Tabelle: Name → Inode (über Inode-Nr.)
- ❑ Hard-Link: 1 Inode ist mindestens 1 Name zugeordnet

Einführung in die Systemverwaltung unter UNIX

27

## Inodes: Komponenten einer Datei



Einführung in die Systemverwaltung unter UNIX

28

## Prozesse: Übersicht



- Grundlagen
- Attribute
- Zustände
- Lebenszyklus
- Signale
- Prozeß-Gruppen
- Interpreter-Skripten als Kommandos

Einführung in die Systemverwaltung unter UNIX

29

## Prozesse : Grundlagen



- Definition: Verwaltungs-Einheit für die Bearbeitung von Aufträgen und die Zuteilung von Ressourcen
- Attribute: wichtige Parameter ( "Prozeß-Kenndaten " ); z.B. Prozeß-ID, Priorität u.ä.
- Lebenszyklus: ⇒ Lebensdauer, Prozess-Zustände
- Signale: Infos über Ausnahme-Zustände (vom Betriebssystem oder von anderen Prozessen)  
⇒ Prozess-Steuerung
- Prozessgruppen: ⇒ einheitliche Behandlung mehrerer Prozesse

Einführung in die Systemverwaltung unter UNIX

30

## Prozesse : Attribute (Beispiele)

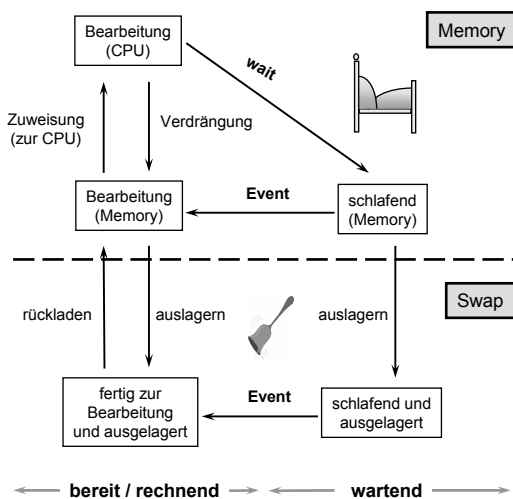


- |   |   |
|---|---|
| <input type="checkbox"/> PID              | ⇒ Process ID                                |
| <input type="checkbox"/> PPID             | ⇒ Parent Process ID                         |
| <input type="checkbox"/> UID / EUID       | ⇒ User ID / Effective User ID               |
| <input type="checkbox"/> GID / EGID       | ⇒ Group ID / Effective Group ID             |
| <input type="checkbox"/> Prozeßgruppe     | ⇒ ( s.u. )                                  |
| <input type="checkbox"/> Prioritäten      | ⇒ nice-Wert; vgl.a. <code>renice (1)</code> |
| <input type="checkbox"/> Control Terminal | ⇒ vgl. Abschnitt " Terminal-Treiber "       |
| <input type="checkbox"/> Statistik-Werte  | ⇒ Rechenzeit, I/O-Durchsatz etc.            |

Einführung in die Systemverwaltung unter UNIX

31

## Prozesszustände



### Verdrängung:

- Abgelaufene Zeitscheibe
- Höher priorisierter Prozess

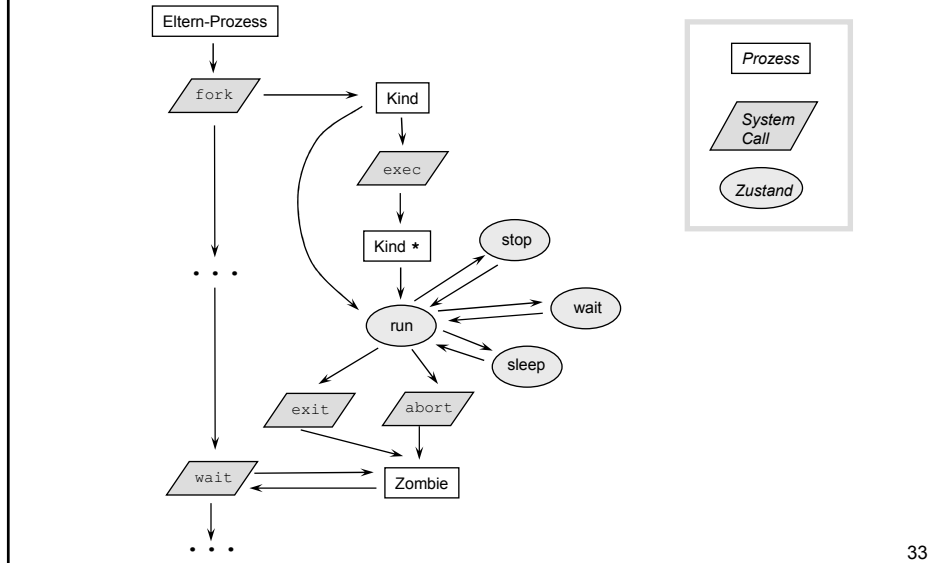
### Scheduler:

Organisation der Zustandsübergänge

Einführung in die Systemverwaltung unter UNIX

32

## Prozesslebenszyklus: Schematische Darstellung



## Prozesslebenszyklus : Erläuterungen (1)



`fork (2)` / `vfork (2)` :

- Erzeugung eines neuen Prozesses ( bis auf [P]PID *identische* Kopie des Eltern-Prozesses)
- `vfork (2)` dupliziert nur Prozeß-Attribute; meist gefolgt von `execve (2)`

Eltern-Prozess :

- startet 1 oder mehrere Kindprozesse
- analysiert Return-Code der Kindprozesse und reagiert darauf
- Im Fehlerfall erbt `init` die Kindprozesse und beendet sie selbst.

## Prozesslebenszyklus : Erläuterungen (2)



Kind-Prozess:

- bearbeitet die durchzuführenden Aufgaben
- Zyklus: run ⇒ wait / sleep / stop ⇒ exit / abort
- gibt Return-Code an Eltern-Prozess zurück
- beenden regulär mit `exit (2)` oder im Fehlerfall mit `abort (3C)`
- Zombies :
  - Prozessattribute (und nur diese !) leben weiter, solange sie der Eltern-Prozess nicht abgeholt hat.
  - Größere Mengen Zombies stören den Betrieb ( ⇒ vollschreiben der Prozess-Tabelle); Beseitigung durch Beenden des Eltern-Prozesses oder durch Reboot

## Prozesse : Signale (1)



Funktion :

- Übermittlung von sehr einfachen Botschaften (von anderen Prozessen oder vom Betriebssystem) an einen Prozeß
- Jede Botschaft besteht aus einer festgelegten Nummer, die nicht geändert werden kann.  
Es gibt ca. 30 - 40 mögliche Nummern.
- anschauliche Kurznamen zur besseren Handhabung

## Prozesse : Signale (2)



Wirkung (1): Steuerung

- Die Interrupt-Tabelle ordnet Signalen bestimmte Wirkungen zu.
- Tabellen-Index ist die Signal-Nummer.
- Die Interrupt-Tabelle kann (mit wenigen Ausnahmen) verändert werden, um Default-Prozeduren durch eigene Prozeduren zu ersetzen.

Bsp.: differenzierte(re) Fehlerbehandlung

## Prozesse : Signale (3)



Wirkung (2): Default

Jedes Signal löst (abhängig von seiner Nummer ) eine der folgenden *Standard*-Prozeduren aus :

- Abbruch des Prozesses mit / ohne Core-Dump
- Anhalten des Prozesses; die Blockade kann nur von einem anderen Prozeß wieder aufgehoben werden.
- keine Aktion, d.h. das Signal wird ignoriert

## Prozesse : Signale (4)



### Wirkung (3) : Benutzersteuerung

- Für fast jede Signal-Nr. kann *nachträglich* eine individuelle Reaktion definiert werden:
  - ignorieren des Signals
  - ausführen einer eigenen Prozedur
  - wiederherstellen des Default-Zustandes
- Bei manchen Signalen kann der Default nicht verändert werden.
- Verzögerung der Signal-Zustellung durch höher priorisierte Vorgänge (z.B. Kernel-wait-Zustände) möglich; Signal ist dann scheinbar wirkungslos; schlimmstenfalls gar keine Zustellung mehr; selbst `kill -9` funktioniert dann scheinbar nicht mehr

## Prozesse : Wichtige Signale



| <u>Nr.</u> | <u>Name</u>                | <u>Häufig verwendete Interpretation</u>                               |
|------------|----------------------------|---|
| 1          | SIGHUP (Hangup)            | ⇒ Abbruch mit Fehler oder erneutes Lesen von Konfigurationsdateien    |
| 2          | SIGINT (Interrupt)         | ⇒ Abbruch mit Fehlerbehandlung  |
| 3          | SIGQUIT (Quit)             | ⇒ Abbruch mit Fehlerbehandlung  |
| 9          | SIGKILL (Kill)             | ⇒ sofortiger Abbruch  |
| 17         | SIGSTOP (Stop)             | ⇒ anhalten des Prozesses  |
| 18         | SIGTSTP<br>(Tastatur-Stop) | ⇒ selber Effekt wie SIGSTOP; kann aber bei Bedarf "abgeklemmt" werden |
| 19         | SIGCONT (Continue)         | ⇒ Wiederaufnahme des Prozesses  |

## Prozesse : Wichtige Kommandos



### kill (1)

#### Anmerkung :

- Signale können über Namen oder Nummer angesprochen werden
- `kill -0 PID` testet, ob ein Prozeß mit der Nummer `PID` existiert
- manchmal wirkt auch `kill -9` nicht  
⇒ Reboot erforderlich

### signal (3C)

⇒ Signal-Steuerung

## Prozessgruppen



### Zweck :

- Signale können an gesamte Prozessgruppen und nicht nur an einzelne Prozesse ausgegeben werden.
- Wichtige Anwendung: Terminal-Betrieb
  - Regelung der Zugriffs-Rechte
  - Signale des tty-Treibers
  - "Control-Terminal"

### Realisierung :

- Kind-Prozeß erbt die Gruppe des Eltern-Prozesses
- Mit `setpgroup (2)` kann später eine beliebige Gruppe gewählt werden.

## Interpreter-Skripten als Kommandos (1)



### Motivation :

Zur Arbeitsvereinfachung können Interpreter-Skripten (meist) wie Kommandos verwendet werden.

### Vorteile :

- Benutzer muß nicht wissen, welcher Interpreter verwendet wird
- Aufruf *direkt mit dem Namen* wie bei einem Binary  
Voraussetzung: Pfad muß im Suchpfad stehen
- Tipp-Ersparnis ☺

Skripten folgender Interpreter werden häufig verwendet :

`sh (1)`, `ksh (1)`, `perl (1)`, `tcl (1)`

Achtung: Nie ein Script mit SUID- oder SGID-Eigenschaften !

## Interpreter-Skripten als Kommandos (2)



### Syntaktische Voraussetzung :

Die 1. Zeile des Interpreter-Skripts hat die Form

`#! absoluter Pfad [ [multi-] Option ]`

### Achtung :

- „#!“ ist eine sogenannte "Magic Number"
- Die 1. Zeile darf meist nicht länger als 32 Zeichen sein

### Beispiele :

```
#!/bin/sh
#!/bin/csh -f
#!/usr/local/bin/perl -w
#!/bin/sh -xv
```

## Interpreter-Skripten als Kommandos (3)



### Funktionsweise :

`execve (2)` startet den Interpreter wie in der 1. Zeile angegeben und fügt als Argument den Pfad des Skripts und alle Argumente der Kommando-Zeile an.

### Beispiel für die Wirkungsweise :

```
1. Zeile des Skripts new-home:  #!/bin/sh -xv
Aufruf von new-home:          new-home login group
Parameter für exec:           0.) /bin/sh
                               1.) -xv
                               2.) new-home
                               3.) login
                               4.) group
„fertiger“ Aufruf :           /bin/sh -xv new-home login group
```

## Dämonen : Übersicht



- Grundlagen
- Beispiele
- crontab
- Der `syslog`-Dämon

## Dämonen : Grundlagen



- ❑ Zweck:  
Prozesse, die keinem Control-Terminal zugeordnet sind und periodische Systemaufgaben wahrnehmen (i.d.R. Programme und keine Shell-Scripten).
- ❑ Statische Dämonen:  
Einmaliger Start; leben potentiell unendlich; bearbeiten Aufträge der Reihe nach  
Bsp.: `nfsd` (1M)
- ❑ Dynamische Dämonen:  
Neustart für jeden Auftrag; beenden sich danach „von selbst“  
Bsp.: `ftpd` (1M), `tftpd` (1M), `getty` (1M)
- ❑ Mischformen:  
Elternprozeß statisch, Kindprozeß dynamisch  
Bsp.: `init` (1M), `inetd` (1M), `sendmail` (1M)

Einführung in die Systemverwaltung unter UNIX

47

## Dämonen : Beispiele (1)



- |  |   |
|--|---|
| <code>init</code> (8)                                  | ⇒ "Urahn" aller Prozesse;<br>automatischer Start beim Boot; PID = 1   |
| <code>inetd</code> (8),<br><code>inetd.conf</code> (5) | ⇒ überwacht Ports der in <code>/etc/inetd.conf</code><br>konfigurierten Dämonen; startet sie bei Bedarf         |
| <code>cron</code> (8),<br><code>crontab</code> (1)     | ⇒ startet mit <code>crontab</code> (1) verwaltete Aktionen zu<br>vordefinierten, period. wiederkehrenden Zeiten |
| <code>at</code> (1)                                    | ⇒ startet Kommandos zu einem vordefinierten,<br>späteren Zeitpunkt einmal                                       |
| <code>atq</code> (1)                                   | ⇒ zeigt <code>at</code> -Prozesse an  |
| <code>batch</code> (1)                                 | ⇒ wie <code>at</code> (1), allerdings Starten erst beim Unter-<br>schreiten einer vorgegebenen Lastschranke     |

Einführung in die Systemverwaltung unter UNIX

48

## Dämonen : Beispiele (2)



|                                 |   |
|---------------------------------|---|
| syslogd (8),<br>syslog.conf (5) | ⇒ Zentrale Verteilung von Log-Meldungen, die vom Kernel, von Dämonen, von Benutzerprogrammen oder von logger (1) erzeugt wurden |
| ftpd (8)                        | ⇒ FTP-Server  |
| tftpd (8)                       | ⇒ Server für das „Trivial File Transfer Protocol“   |
| dhcpcd (8)                      | ⇒ Server für das „Dynamic Host Configuration Protocol“  |

## Beispiel für eine crontab



```
#####  
## Beispiel fuer eine crontab (5)  
## Zeilenumbrüche syntaktisch falsch; nur zum Überblick  
#####  
## min hour day_of_month month day_of_week  
## 0-59 0-23 1-31 1-12 0-6, 0=Sunday  
#####  
## min h md mo wd command  
##  
2 0,2,4,6,8,10,12,14,16,18,20,22 * * * \  
/usr/local/etc/xntpdate -s ntp1 ntp2  
15 4 * * * /usr/local/etc/cron.dayly  
##  
10 2 * * * /usr/local/etc/ADDUSER/bin/adduser  
##  
5 4 * * 6 /usr/lib/newssyslog >/dev/null 2>&1  
##  
0 7 * * 1 /usr/etc/reboot  
#####
```

## syslog-Dämon



### Funktion:

- Zentraler Verteiler für Systemmeldungen (z.B. Kennwort- / Plattenfehler)
- Quellen der Meldungen:
  - Kernel, Systemprozesse
  - Sehr selten Benutzerprozesse
  - logger (1) ⇨ in Shell-Skripten

### Übersicht:

- Meldungs-Format
- Konfiguration
- Schnittstellen
- Anlegen eines neuen Log-Files

Einführung in die Systemverwaltung unter UNIX

51

## syslog-Dämon : Meldungsformat



### Facility (Meldungsklasse, Themengebiet):

auth, authpriv, cron, daemon, kern, lpr, mail, mark,  
news, syslog, user, uucp, local0, ..., local7

### Priority:                      Priorität (abnehmend):

|         |   |
|---------|---|
| emerg   | Panic Conditions                                      |
| alert   | Fehlersituation, die sofort behoben werden muß        |
| crit    | Kritischer Fehler (z.B. HW-Fehler)                    |
| err     | Sonstiger Fehler                                      |
| warning | Wichtige Meldung; kein Fehler, aber „Ungereimtheiten“ |
| notice  | Besondere Behandlung erforderlich                     |
| info    | z.B. für spätere Auswertung                           |
| debug   | z.B. zur Fehler-Diagnose                              |

Meldungs-Text:            Typischerweise „1-Zeiler“ ; sollte nicht länger als 40 Zeichen sein; automatische Ergänzung diverser Zusatz-Infos

Einführung in die Systemverwaltung unter UNIX

52

## /etc/syslog.conf : Allgemeines



- ❑ `syslog.conf` (5): Konfigurations-Datei für `syslogd` (8)
- ❑ Wird vor dem Lesen mit Makro-Prozessor `m4` (1) bearbeitet
  - Bei größeren Änderungen `m4` - Syntax beachten !
  - I.a. genügen geringfügige Modifikationen der Default-Konfiguration.
- ❑ Leerzeilen und mit "`#`" beginnende Zeilen werden als Kommentar ignoriert.
- ❑ Jeder passende Eintrag wird ausgeführt.

## /etc/syslog.conf : Syntax (1)



- ❑ Jeder Eintrag ist eine Zeile der Form  
`selector TAB [ ... ] action`
  - `selector`: Auswahl-Kriterium für Meldungen
  - `action`: Aktion, falls `selector` erfüllt ist
- ❑ `selector` hat die Form  
`facilities.level [ ; facilities.level [ ... ] ]`
  - ⇒ Abarbeitung von links nach rechts
  - ⇒ letzter Match gilt

## /etc/syslog.conf : Syntax (2)



- *facilities* hat die Form

*facility* [ , *facility* [ ... ] ]

- mögliche Werte für *facility* :

- Eine der Meldungs-Klassen  
*user, kern, mail, daemon, auth, lpr, news, uucp, cron, local0, ..., local7*
- \* ⇒ match all ( d.h. Passt auf jede Meldungs-Klasse )
- mark ⇒ periodischer Zeitstempel (alle 20 Min.)

## /etc/syslog.conf : Syntax (3)



- Mögliche Werte für *level* :

- *none* hat die höchste Priorität
- eine der Meldungs-Prioritäten : *emerg, alert, ... debug*  
(absteigende Priorität)

- Eine Meldung erfüllt *selector*, wenn

- die Meldungs-Klasse zu *facility* passt und
- die Meldungs-Priorität  $\geq$  *level* ist.

## /etc/syslog.conf : Syntax (4)



### □ Mögliche Werte für *action* :

- absoluter Pfad ⇒ anhängen an **bereits existente** Datei
- *@host* ⇒ Weiterleitung der Meldung an den Rechner *host*
  - ⇒ zentrale Auswertung
  - ⇒ Verhinderung der nachträglichen Manipulation von Logs
- weitere technisch mögliche Varianten nicht Praxis-relevant !

## /etc/syslog.conf : Beispiele



| <u>selector</u>       | <u>action</u>        |
|-----------------------|----------------------|
| *.notice; mail.info   | /var/log/notice      |
| *.crit                | /var/log/critical    |
| kern, mark.debug      | /dev/console         |
| *.err; auth.info      | @loghost             |
| *.emerg               | *                    |
| *.alert               | root, operator       |
| *.alert; auth.warning | /var/log/auth        |
| *.err; news.none      | /var/log/errors      |
| news.err              | /var/log/errors.news |



## Benutzerverwaltung : Übersicht



- Überblick
- /etc/passwd
- Exkurs : Umgang mit Passwörtern
- /etc/shadow
- /etc/group
- Zuordnung von Kennungen zu Gruppen
- Einrichten von Benutzer-Kennungen

Einführung in die Systemverwaltung unter UNIX

61

## Benutzerverwaltung : Überblick



Benutzer-Verwaltung ist eine der wichtigsten und häufigsten Aufgaben von Administratoren.

Hierzu gehören :

- Einheitliche, konsistente Richtlinien entwickeln und konsequent in die Praxis umsetzen
- Kennungen einrichten / pflegen / löschen
- Einfache Standardbenutzer-Umgebung einrichten und anbieten
- Wichtige Dateien :
  - /etc/passwd
  - /etc/shadow
  - /etc/group

Einführung in die Systemverwaltung unter UNIX

62

## /etc/passwd (1)



### Jede Zeile besteht aus durch ":" getrennten Feldern :

- Kennung (Login-Name)
- verschlüsseltes Passwort ⇒ `crypt (3c)`
- UID
- Login-GID
- GECOS ⇒ "fullname"
- Home-Directory
- Login-Shell ⇒ Default (bei leerem Feld): `/bin/sh`

### Achtung :

- nie leeres verschlüsseltes Passwort !!!
- sichere **Passwörter wählen !**

## /etc/passwd (2)



### Sperren der Kennung :

i.a. durch "\*" oder "nologin" als verschlüsseltes Passwort

### reguläres verschlüsseltes Passwort :

- 13 Zeichen lang
- Groß- bzw. Kleinbuchstaben
- Ziffern
- "/" (Slash) und "." (Punkt); keine anderen Sonderzeichen

### Wichtige man-Pages :

- `passwd (1)`, `passwd (5)`
- `vipw (1B)`, `chfn (1)`, `chsh (1)`, `yppasswd (1)`

## Umgang mit Paßwörtern (1)



### Schlechte Paßwörter (1)

- am schlimmsten: kein Paßwort, vom Admin. gesetztes Default-Paßwort oder zu kurzes Paßwort ( *ich, 123, abc* )
- ⊗ Paßwort und Kennung lauten gleich: *u1234ab*
- ⊗ eigener Nachname: *ruehmann*
- ⊗ eigener Vorname oder Vorname der Freundin: *hertha*
- ⊗ Daten des persönlichen Umfelds (evtl. leicht verfremdet) , z.B.:
  - Geburtsdatum: *290264*
  - Adresse: *Altstr12*
  - Arbeitsplatz: *bio\_lmu*
  - Autonummer: *M\_XY\_123*

Einführung in die Systemverwaltung unter UNIX

65

## Umgang mit Paßwörtern (2)



### Schlechte Paßwörter (2)

- ⊗ leicht zu tippende Zeichenfolgen:  
*12345678, xxxxxxxx, qwertz, qwerty, huikmnj*
- ⊗ Ortsnamen: *muenchen*
- ⊗ gängige Wörter aus natürlichen Sprachen:  
*passwort (!), geheim, secret, mydates, handsup*
- ⊗ typische Gastkennungen: *guest, gast, public, common*
- ⊗ bekannte "Kult-Wörter": *gandalf, spock, dilbert, dr Mabuse*
- Name der Institution, bei der man rechnet: *lrz, lmu, tum, fhm*  
( Diese Beispiele sind außerdem zu kurz ! )

Einführung in die Systemverwaltung unter UNIX

66

## Umgang mit Paßwörtern (3)



### Gute Paßwörter (1)

- ☺ vermeiden alle Eigenschaften schlechter Paßwörter
- ☺ sind leicht zu merken ( jeder Zettel ist eine potentielle Gefahr ! )
- ☺ sind lang genug (8 Zeichen, mindestens 1 Ziffer od. Sonderzeichen)

### Collage-Methode :

Man wählt ein Wort einer natürlichen Sprache, übersetzt es in eine andere natürliche Sprache, wählt jeweils 3 Buchstaben aus und fügt 2 Sonderzeichen dazu.

Bsp.: "Haus" → "HOUSE", Hausnr. 23 ⇨ *hau23HOU*

Verfremdung z.B. durch Großbuchstaben, Komplement-Zahlen ...

## Umgang mit Paßwörtern (4)



### Gute Paßwörter (2)

### Akronym- Methode :

Man wählt einen Satz und verwendet dann als Paßwort die Anfangsbuchstaben der einzelnen Wörter, wobei noch 1-2 Ziffern oder Sonderzeichen eingefügt werden.

Bsp.: *MweS&&vd* (Akronym für den Anfang des obigen Satzes)

Weitere Hinweise unter:

<http://www.lrz.de/services/security/>

## Umgang mit Paßwörtern (5)



### Gefährliche Akronym-Paßwörter

- ☛ Volkstum / Kultur : *AmEsadS*  
*GmDDLdB*
- ☛ Klassik : *WrssdN&W*  
*HsiniaT*
- ☛ Naturwissenschaft : *MVEMJSUNP*  
*3.141592* bzw. *2.718281*
- ☛ "Kult" : *HkdM* bzw. *DSmdM*  
*H!FdWv!* bzw. *H!MdWa!*  
*DWuWwsdJ* bzw. *DW:uW-Ws*

## Umgang mit Paßwörtern (6)



### Paßwort ändern

- Das Paßwort sollte unbedingt regelmäßig geändert werden und zwar :
  - alle 3-6 Monate (Benutzer) bzw. alle 1-2 Monate (Admins.)
  - auf allen Rechnern, die (regelmäßig oder auch nur gelegentlich) benutzt werden
- Kommando `passwd`

```
$ passwd
Old password:           ( zur Legitimation )
New password:
Re-type new password:  ( wegen "Blind-Eingabe" )
```
- Bei der Paßwort-Eingabe nie über die Schulter schauen lassen !

Jede Zeile besteht aus durch ":" getrennten Feldern :

- Kennung (Login-Name)
- verschlüsseltes Paßwort ⇒ crypt (3c)
- Datum der letzten Paßwort-Änderung
- minimaler Zeitraum, der bis zur nächsten Paßwort-Änderung verstreichen muß
- maximaler Zeitraum, der bis zur nächsten Paßwort-Änderung verstreichen darf
- Anzahl Tage, nach deren Ablauf vor Paßwort-Verfall gewarnt wird
- maximal-Zeitraum, in dem die Kennung nicht benutzt werden darf, ohne gesperrt zu werden
- absolutes Verfallsdatum

Wichtige Man-Page :

shadow (5)

Jede Zeile besteht aus durch ":" getrennten Feldern :

- Gruppe
- verschlüsseltes Paßwort
- GID
- Gruppenmitglieder, getrennt durch ",", "

Achtung :

- Feld für verschlüsseltes Paßwort nie leer lassen !!!
- verschlüsseltes Paßwort meist "\*" "

- Feld für Gruppenmitglieder oft leer :
  - Login-Gruppe eines Benutzers ist bereits in `/etc/passwd` spezifiziert
  - wird nur benötigt, wenn Benutzer zu mehr als 1 Gruppe gehört
  
- Wichtige Man-Pages :
  - `groups (1)`, `group (4)`
  - `id (1M)`, `newgroup (1)`

- in `/etc/passwd` ⇒ Eintrag der Gruppe als Login-GID
- in `/etc/group` ⇒ Eintrag der Kennung in " Mitglieder "
- SGID-Eigenschaft ( bei Kommandos )
- Die Gruppe hat ein "echtes" Paßwort, das der Benutzer kennt  
⇒ `newgrp (1M)`; in der Praxis selten genutzt

Anmerkung :

- Jeder (Benutzer-)Prozeß hat zu jedem Zeitpunkt die Rechte von mindestens 1 Gruppe ( Login-Gruppe ) .
- Berkeley- und neuere SysV-Systeme erlauben auch noch zusätzliche Gruppen ( maximal ca. 16 )

## Einrichten von Benutzerkennungen (1)



### Konzeptionelle Überlegungen

- unbedingt vorher Konzepte überlegen für
  - UID- / GID-Bereiche
  - Organisation von Home-Directories
  - Welche Gruppen sollen angeboten werden ?
- Vorschlag für UID-Bereiche:
  - ≤ 99      ⇒ für System freihalten
  - 100 - 999 ⇒ lokale Spezialkennungen
  - ≥ 1000   ⇒ "normale" Benutzer
- Vorschlag für GID-Bereiche:
  - ≤ 99      ⇒ System und lokale Spezial-Gruppen
  - ≥ 100     ⇒ "normale" Gruppen

Einführung in die Systemverwaltung unter UNIX

75

## Einrichten von Benutzer-Kennungen (2)



### Vorbereitende Aktionen

- einfache Standard-Benutzer-Umgebung anbieten und darauf hinweisen :
  - ☺ Benutzer können sofort anfangen, zu arbeiten
  - ☺ einfachere Fehlerbehebung
- Hinweis: "alle Änderungen an der Standard-Umgebung auf eigene Gefahr"
- eventuelle Benutzungs-Ordnungen dokumentieren und bekannt machen

Anderenfalls riskieren Sie "von irgendwo übernommene" Profiles, die schwer durchschaubare Fehler verursachen können.

Einführung in die Systemverwaltung unter UNIX

76

## Einrichten von Benutzer-Kennungen (3)



### Benutzer eintragen (Vorbereitung)

- kennung* in `/etc/passwd` mit "\*" als Passwort-Sperrzeichen einrichten; *kennung* ist jetzt bereit, aber noch nicht aktiv
- ggfs. entsprechende Aktionen in `/etc/shadow`

### Benutzer eintragen (konkrete Kennung einrichten)

- Kennung eintragen
- Passwort setzen: `passwd kennung`
- evtl. Gruppe in `/etc/group` einrichten bzw. Kennung eintragen
- Home-Verzeichnis anlegen
- Standard-Umgebung in Benutzer-Home-Verzeichnis kopieren
- `chown -R kennung homedir`
- evtl. `chgrp -R gruppe homedir`

## Datenhaltung : Übersicht



- Konzepte der Datenhaltung
- Konzepte zur Organisation von Home-Directories
- Home-Directories: Beispiele
- Kontingentierung von Plattenplatz
- Pflege von Software

## Datenhaltung : Konzepte



- auf keinen Fall Benutzer- und System-Daten mischen und unstrukturiert / durcheinander auf beliebige Rechner verteilen !
- Besser : Client-/Server-Lösung, d.h.
  - dedizierte Fileserver (möglichst) ohne Benutzerbetrieb (⇒ Stabilität, Performance, Security)
  - Zugang zum Fileserver für möglichst wenige Personen (⇒ Security)
- Server :
  - möglichst wenige, aber große Platten (mindestens 40 GB)
  - möglichst große Partitionen (mindestens 1 GB)

### Anmerkung :

Automounter erleichtert Administration und verbessert Stabilität

## Organisation von Home-Directories : Konzepte (1)



### Allgemeine Anforderungen :

- gleicher Pfad und gleiche Umgebung auf allen Rechnern
- gleiche NFS-Konfiguration auf allen Rechnern

### Realisierung :

- Verwendung des Automounters (s.u.)
- Konfiguration über NIS-Maps (⇒ Konfigurations-Dateien zentral)

### Nachteile :

- Bei älteren Automounter-Versionen liefert `pwd` unerwartete Ergebnisse.
- Nicht gemountete Datei-Äste sind scheinbar nicht vorhanden.

## Organisation von Home-Directories : Konzepte (2)



### Wichtig :

Die Pfade der Home-Directories sollten keinesfalls Hinweise auf den Rechner, die Platte oder die Partition des *physischen* Speicherungs-Ortes enthalten !

Grund: umständlich, oft inkonsistent und Fehler-anfällig !  
( z.B. beim Auswechseln einer defekten Platte )

⇒ Pfade der Home-Directories über Soft-Links konfigurieren

## Organisation von Home-Directories : Beispiele (1)



### Fall 1 :

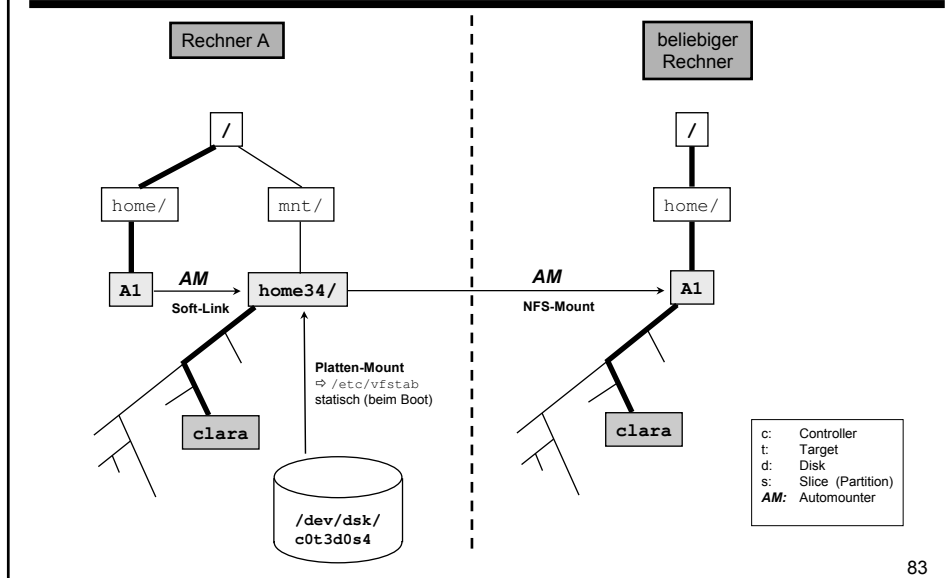
Die gesamte Partition `/dev/dsk/c0t3d0s4` am Rechner A soll für Home-Directories genutzt werden.

Für Rechner A gilt :

- `/dev/dsk/c0t3d0s4` ist auf `/mnt/home34` gemountet
- `/mnt/home34` wurde für die lokalen Rechner schreibbar exportiert

⇒ Automounter stellt `A:/mnt/home34` überall als `/home/A1` zur Verfügung (Verzeichnis-Name beim Automounter konfiguriert).

## Organisation von Home-Directories : Beispiele (2)



## Organisation von Home-Directories : Beispiele (3)



### Fall 2:

Die Partition `/dev/dsk/c0t3d0s1` am Rechner A soll teilweise für Home-Directories genutzt werden.

Für Rechner A gilt:

- `/dev/dsk/c0t3d0s1` ist auf `/mnt/misc31` gemountet
  - `/mnt/misc31/users` wurde für die lokalen Rechner schreibbar exportiert
- ⇒ Automounter stellt `A:/mnt/misc31/users` überall als `/home/A2` zur Verfügung (Verzeichnis-Name beim Automounter konfiguriert).

## Kontingentierung von Plattenplatz (1)



### Motivation für Quotas :

Kontingentierung des Plattenplatzes und der Anzahl Dateien;  
"Überbuchung" aus ökonomischen Gründen in der Regel sinnvoll

### Realisierung :

2 Obergrenzen pro Partition und Benutzer  
( jeweils für Plattenplatz und Datei-Anzahl ) :

- Hard-Limit : kann prinzipiell nicht überschritten werden
- Soft-Limit : kann zeitweise überschritten werden ( Zeitraum konfigurierbar )  
erst, wenn das Soft-Limit 1 mal wieder unterschritten wurde,  
kann es wieder überschritten werden

## Kontingentierung von Plattenplatz (2)



### Voraussetzungen :

- mount-Option `quota` ( z.B. in `/etc/vfstab` )
- Datei `quotas` im Wurzelbereich des Filesystems muß `root` gehören und sollte nur für `root` lesbar sein;  
muß bei Bedarf z.B. mit `touch (1)` angelegt werden
- Hard- und Soft-Limit für alle Benutzer, für die die Kontingentierung gelten soll
- Aktivierung mit `quotaon (1M)`  
( erfolgt i.a. beim Boot automatisch )

## Quota : Kommandos



- `quotaon (1M)`      ⇒ Kontingentierung aktivieren
- `quotaoff (1M)`    ⇒ Kontingentierung deaktivieren
- `edquota (1M)`     ⇒ Limits administrieren
- `quota (1M)`        ⇒ Anzeige individueller Quotas und Belegungen ( auch für Benutzer ! )
- `repquota (1M)`    ⇒ Übersichtslisten erstellen
- `quotacheck (1M)` ⇒ Konsistenz-Check / -Abgleich :  
                          `quota`-Datenbasis ⇔ reale Belegung  
                          im Filesystem

Einführung in die Systemverwaltung unter UNIX

87

## Das Kommando `edquota (1M)`



- Zweck : setzen / verändern von Quotas (an dem Rechner, wo die Daten *physisch* liegen)  
Aufruf : `edquota <kennung> [...]`; startet per Default `vi (1)`
- Option "`-p proto_user`" gestattet die Übernahme der Limits von `proto_user` ⇒ Batch-Betrieb ⇒ Prototyp-Klassen  
Anwendungs-Bsp.: Prototypen für verschiedene Benutzerklassen
- Trick für individuellen Batch-Betrieb :
  - ⇒ Environment-Variable `$EDITOR` auf ein Kommando setzen, das als Argument einen Datei-Namen erwartet und diese Datei im Batch-Betrieb verarbeiten kann
- Mit `edquota (1M)` können auch die Zeit-Limits verändert werden.

Einführung in die Systemverwaltung unter UNIX

88

## Pflege von Software / Konfigurationen (1)



- ❑ wichtig : lokale Erweiterungen zusammenfassen  
⇒ /usr/local

- ⇒ übersichtlicher und weniger Fehler-anfällig
- ⇒ weniger Probleme beim Release-Wechsel

- ❑ vorher Sicherungskopie erzeugen: `cp -p file file.orig`

### Vorteile :

- `diff file.orig file` zeigt vorgenommene Modifikationen an
- `diff -c file.orig file` erzeugt einen Kontext-Diff.
- Aus `file.orig` und dem Kontext-Diff. kann mit `patch (1)` wieder die Datei `file` erzeugt werden.

Einführung in die Systemverwaltung unter UNIX

89

## Pflege von Software / Konfigurationen (2)



### Vorgehensweise :

- ❑ "Was wurde letztes Mal geändert ?"

```
diff file.orig file
```

- ❑ Nachziehen lokaler Modifikationen :

```
file.orig      }  
Kontext-Diff. }  → patch (1L) → file
```

```
file.V2       }  
Kontext-Diff. (alt) }  → patch (1L) → file.V2'  
                                     (mit lok. Modifikationen)
```

Einführung in die Systemverwaltung unter UNIX

90

## Datensicherung : Übersicht



- Motivation
- Backup-Arten
- Backup mit `dump / ufsdump`
- Exkurs : Backup-Level
- Zurückladen mit `restore / ufsrestore`
- Backup durch Benutzer

Einführung in die Systemverwaltung unter UNIX

91

## Datensicherung : Motivation (1)



### Wozu braucht man Datensicherung ?

- Restaurierung verloren gegangener Daten
- Zurücksetzen auf ältere Version
- Verlagerung von Daten

### Achtung :

- Datensicherung  $\neq$  Archivierung
- Datensicherung  $\neq$  Datenschutz
- Datensicherung  $\neq$  Security
- Datensicherung hat *indirekt* mit Datenschutz / Security zu tun

Einführung in die Systemverwaltung unter UNIX

92

## Datensicherung : Motivation (2)



### " Lebensversicherung " gegen Daten-Verlust durch

- Bedienungs-Fehler    ⇨    `rm -r *`
- HW- / SW-Fehler
- Sabotage : Zerstörung, Entwendung, Verfälschung, Notebooks / PDA's (!)
- Katastrophen ( Feuer, Wasserrohrbruch etc. )

### Arten der Datensicherung :

- System-interne* Schutzmechanismen ( z.B. Spiegelplatten, RAID-Systeme, Migrations-Systeme )
- Externe* Mechanismen ( Backup, d.h. Sicherungs-Kopien )

Einführung in die Systemverwaltung unter UNIX

93

## Datensicherung : Backup-Arten



### Durch die Systemadministratoren :

- kompletter Bereich ( z.B. gesamte Home-Directories, System-Partitionen)
- regelmäßig ( z.B. nachts, am Wochenende )
- Schema für Durchführung von Vollsicherungen bzw. inkrementellen Sicherungen

### Durch den Benutzer selbst :

- meist nur Teilbereiche
- sporadisch, d.h. nach Bedarf
- meist nur Vollsicherung

Einführung in die Systemverwaltung unter UNIX

94

## Backup mit `dump / ufsdump` (1)



- Sicherung kompletter Partitionen durch den Systemadministrator
- Plattform-spezifisch !
- garantiert konsistenter Dump nur bei nicht oder "read-only"-gemounteten Partitionen  
funktioniert aber auch im normalen Betrieb meist problemlos (lokale Inkonsistenzen einzelner Dateien sind in der Regel unkritisch)
- Sehr selten ist ein Dump ab einer gewissen Zeit komplett unbrauchbar  
⇒ Full-Dump
- wegen direktem Zugriff sehr effizient  
( d.h. Filesystem-Mechanismen werden umgangen )

## Exkurs : Backup-Level



- 0 gesamter Datenbestand
- 1 Daten, die seit dem letzten Level-0-Backup verändert wurden
- 2 Daten, die noch nicht in einem Level-0- bzw. Level-1-Backup gesichert wurden
- ...
- n Daten, die noch nicht in einem Level-0, -1, ..., (n-1)-Backup gesichert wurden

Praxis-relevant : Level 0 - 2

## Backup mit `dump / ufsdump` (2)



- Umfang der Sicherung :
  - Vollsicherung (Level 0) normalerweise 1 mal pro Woche / Monat
  - inkrementelle Sicherung (Level 1-n): bei genügend Plattenplatz z.B. täglich komprimiert auf Platte
- Backup auf Ausgabe-Datei / -Gerät an einem anderen Rechner :  
⇒ `rdump` (8)
- Einfache Shell-Skripten reichen zur Automatisierung aus.

## Zurückladen mit `restore / ufsrestore`



- Teilbereiche können selektiv geladen werden.
- Man kann problemlos an einen *beliebigen* Ort zurückladen.
- interaktive und Batch-Schnittstelle
- Beide Kommandos verwenden Filesystem-Mechanismen und sind deshalb viel langsamer als `dump` → `ufsdump`.

## Backup durch Benutzer



- in der Regel einzelne Verzeichnisse / Dateien
  - ⇒ `tar (1)`, `cpio (1)`
- Dump-Archive sind zwischen Plattformen austauschbar.
- Benutzer-Dump problemlos im laufenden Betrieb möglich
- Zugriff über Filesystem-Mechanismen, d.h. langsamer
- Sicherungs-Umfang :
  - alle spezifizierten Dateien / Verzeichnisse
  - genauere Auswahl mit `cpio (1)` in Kombination mit `find (1)`
  - Bsp.: `find . -newer referenz -print | cpio -pdlm > device`
  - `touch referenz`
- Sicherung auf einen anderen Rechner : `ssh (1)`
- zurückladen ⇒ `tar (1)`, `cpio (1)`

Einführung in die Systemverwaltung unter UNIX

99

## Betrieb von ASCII-Terminals : Übersicht

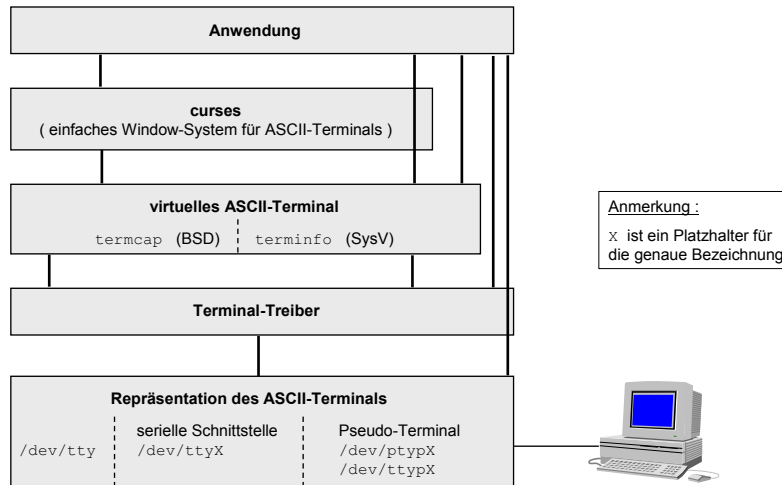


- Ein- / Ausgabe auf ASCII-Terminals (Schichtenmodell)
- Terminal-Virtualisierung
- Die Datei `/etc/termcap`
- Der Dateibaum `/usr/lib/share/terminfo`
- Terminal-Treiber
- Das Kommando `stty`
- Das Kommando `getty`
- Anschließen eines neuen Terminals

Einführung in die Systemverwaltung unter UNIX

100

## Ein- / Ausgabe auf ASCII-Terminals : Schichtenmodell



101

## Ein- / Ausgabe auf ASCII-Terminals : Erläuterungen



- `curses (3)` ist ein einfaches Windows-System für ASCII-Terminals mit relativ wenigen Anwendungen  
**Bsp.:**
  - `sunddiag` (Sun)
  - `smit` (IBM)
  - `sam` (HP)
- `termcap (4)` bzw. `terminfo (4)` werden für nahezu alle Bildschirm-orientierten Anwendungen benötigt ( z.B. `vi (1)` ).
- Der Terminal-Treiber ist die Betriebssystemschnittstelle und wird von allen Anwendungen mit Ein- / Ausgabe auf das Terminal benötigt.

Einführung in die Systemverwaltung unter UNIX

102

## Terminal-Virtualisierung



- Anwendung verwendet *logische* Bezeichnungen für Terminal-Eigenschaften (z.B. `cursor_down`).
- Eine Datenbasis definiert für jeden Terminal-Typ die Abbildung *logische* Bezeichnung  $\Rightarrow$  *realer* Wert
- Beschreibungs-Elemente (Typen) :
  - Bool'sche Werte: `has_status_line, auto_right_margin`
  - Numerische Werte: `lines, columns`
  - Strings: `clear_screen, cursor_address (row, col)`
- C-Funktionen zur Vereinfachung der Programmierung
- Kommandoschnittstelle: `tput (1)`
- Environment-Variable `$TERM` spezifiziert Terminal-Typ

Einführung in die Systemverwaltung unter UNIX

103

## Die Datei `/etc/termcap`



- älterer Virtualisierungsmechanismus (BSD)
- Datenbasis: `/etc/termcap` (normale Text-Datei; enthält die Beschreibung aller lokal bekannten Terminal-Typen)
- Zugriffs-Funktionen : `termcap (3), tput (1)`
- Environment-Variable `$TERMCAP` :
  - absoluter Pfad : Die Datei wird anstelle von `/etc/termcap` verwendet
  - Default : Datensatz zur Beschreibung des Terminals `$TERM` nach der Syntax in `termcap (4)`

Einführung in die Systemverwaltung unter UNIX

104

## Der Dateibaum

/usr/lib/share/terminfo (1)



- neuerer, mächtigerer und effizienterer Virtualisierungsmechanismus (SysV)
- Datenbasis :  
Die Datei /usr/lib/share/terminfo/X/XYZ enthält für den Terminal-Typ "XYZ" die Beschreibung in einem binären Intern-Format, z.B. .../terminfo/v/vt100
- Zugriffs-Funktionen :  
tput (1), low-level-Funktionen von curses (3)
- Environment-Variable \$TERMINFO :  
Terminal-Beschreibung wird zuerst in \$TERMINFO/X/XYZ gesucht

Einführung in die Systemverwaltung unter UNIX

105

## Der Dateibaum

/usr/lib/share/terminfo (2)



### Administrations-Kommandos :

- tic (1M)  
erzeugt aus der Terminal-Beschreibung in ASCII-Form die Intern-Darstellung
- infocmp (1M) ( nur bei Sun )  
Vergleich und Aufbereitung von intern-Darstellung, d.h. Erzeugung
  - ⇒ der ASCII-Darstellung
  - ⇒ einer möglichst gleichwertigen termcap-Beschreibung
- captainfo (1M) ( nur bei Sun )  
wandelt eine termcap-Beschreibung in eine möglichst gleichwertige terminfo-ASCII-Beschreibung um

Einführung in die Systemverwaltung unter UNIX

106

## Terminal-Treiber (1)



- Betriebssystemschnittstelle für Ein- / Ausgabe auf
  - asynchrone Kommunikations-Ports ( physische ASCII-Terminals, PC's mit Terminal-Emulation, Modem-Ports )
  - Pseudo-Terminals, z.B. bei `telnet (1)`, `xterm (1)`, `script (1)`, `ssh (1)`
  - die Schnittstelle `/dev/tty`
- Control-Terminal :  
Nur von hier aus können Eingabe und durch Control-Characters veranlasste Signale des Treibers kommen.
- nur *Vordergrund*-Prozesse erhalten Eingabe und Signale
- Eingriffsmöglichkeiten :  
`stty (1)`, `tset (1B)`, `reset (1B)`, `termios (3)`, `ioctl (2)`

Einführung in die Systemverwaltung unter UNIX

107

## Terminal-Treiber (2)



- Eingabemodi :
  - Canonical Mode :  
Erst nach Eingabe von `<lf>` wird die Eingabe übergeben (einfachste Editier-Möglichkeit).
  - Non-Canonical Mode :  
Eingabe wird sofort und unmittelbar übergeben (z.B. bei Editoren).
- Teilfunktionen :
  - Leitungs-Charakteristik, z.B. Übertragungs-Geschwindigkeit, Parity
  - Eingabesteuerung, z.B. Abbildung von `<lf>` auf `<cr>`
  - Ausgabesteuerung, z.B. Abbildung von Klein- auf Großbuchstaben
  - lokale Aktionen; lokales Echo; stoppen von Hintergrund-Prozessen, die ausgeben wollen
  - Steuerungs-Parameter, z.B. Zeilen / Spalten

Einführung in die Systemverwaltung unter UNIX

108

## Das Kommando `stty` (1)



### Wichtige Argumente :

- `-a` bzw. „everything“ ⇒ Anzeige aller aktuellen Einstellungen
- `echo / -echo` ⇒ lokales Echo an- / ausschalten
- `rows i / columns i` ⇒ Anzahl Zeilen / Spalten
- `sane` ⇒ normieren aller Parameter auf einen „sinnvollen“ Default-Wert
- `tostop / -tostop` ⇒ Hintergrund-Prozesse werden bei Ausgabe gestoppt bzw. dürfen ausgeben

## Das Kommando `stty` (2)



### Control-Characters : (1)

Syntax: *Keyword Zeichen* [ Voreinstellung in eckigen Klammern ]

- `erase/werase/kill` ⇒ letztes Zeichen / Wort / ganze Zeile löschen [ `^?` / `^W` / `^U` ]
- `rprnt` ⇒ bisherige Eingabe anzeigen [ `^R` ]
- `eol/eol2` ⇒ Zeichen zur Signalisierung (zusätzlich zu `<n1>` ) des Zeilenendes [ `^@` ]
- `eof` ⇒ Simulation des Datei-Endes [ `^D` ]

## Das Kommando `stty` (3)



### Control-Characters : (2)

- `lnext`           ⇒ Quoten des nächsten Zeichens [ ^V ]
- `flush`           ⇒ unterdrücken der noch anstehenden Ausgabe [ ^O ]
- `intr/quit/susp`   ⇒ erzeugen der Signale SIGINT / SIGQUIT / SIGSTP [ ^C / ^\ / ^Z ]
- `stop/start`       ⇒ Flußkontrolle [ ^S / ^Q ]

Einführung in die Systemverwaltung unter UNIX

111

## Das Kommando `getty` (8)



- unternimmt die ersten Schritte beim Einloggen auf einem Terminal :
  - überwacht, ob sich ein Benutzer einloggen will
  - initialisiert die Leitungs-Charakteristik
  - präsentiert Login-Prompt
  - fragt Benutzer-Kennung ab
  - ruft `login(1)`, auf, das seinerseits folgende Aktionen durchführt :
    - Abfrage und Überprüfung des Paßworts
    - Ausgabe von Meldungen in `/etc/motd` (sofern vorhanden)
    - Aufruf der Login-Shell
- Neustart durch `init(8)` nach Beendigung einer Terminal-Sitzung

Einführung in die Systemverwaltung unter UNIX

112

## Spooling : Übersicht

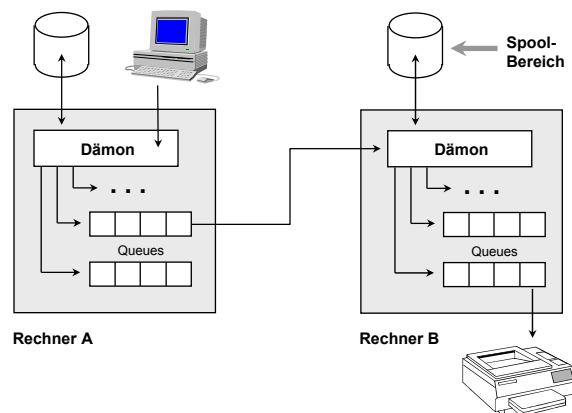


- Architektur
- Konzepte
- BSD-Spooling
- Beispiel für die Datei `/etc/printcap`
- SysV-Spooling

Einführung in die Systemverwaltung unter UNIX

113

## Spooling: Architektur



Einführung in die Systemverwaltung unter UNIX

114

## Spooling: Konzepte



- Einsatz überwiegend zum Drucken bzw. Plotten
- unterschiedliche Systeme bei BSD und SysV  
vgl.a. <http://www.linuxprinting.org>
- Benutzer-Kommandos für Spool-Jobs: erzeugen, löschen, Info
- Konfigurations-Dateien und Kommandos zur Administration
- Öffnen / Schließen der Spooling-Queue und Starten / Stoppen der Queue-Abarbeitung sind unabhängige Aktionen.
- Jede Spool-Queue, über die ausgegeben werden soll, muß lokal bekannt, d.h. konfiguriert sein.

Einführung in die Systemverwaltung unter UNIX

115

## BSD-Spooling (1)



- älteres Spooling-System
- Zuordnung Spool-Queue ⇔ genau 1 Ausgabe-Gerät / andere Queue
- Benutzer-Kommandos :
  - `lpr (1B)` ⇒ Jobs erzeugen
  - `lpq (1B)` ⇒ Informationen über eine gegebene Queue
  - `lprm (1B)` ⇒ Jobs löschen
- Spezifikation eines Druckers :  
mit Option `'-P<Druckername>'`; Default-Drucker: `lp`
- Administration von Queues : `lpc (1M)`

Einführung in die Systemverwaltung unter UNIX

116

## BSD-Spooling (2)



### Konfigurations-Dateien :

- für jeden Drucker konfigurierbar :
  - Queue-Directory; Vorschlag für Rechte: `daemon/daemon/0775`
  - Pfad des Log-Files
  - Pfad der Datei, die die Abrechnungs-Daten enthält
- Für jeden Drucker gibt es interne Hilfsdateien .
- Konfiguration: `/etc/printcap`
- Zugangs-Kontrolle:
  - `/etc/hosts.lpd` ⇨ Liste der Rechner, die Jobs abschicken dürfen
  - `/etc/hosts.equiv` ⇨ Liste vertrauenswürdiger Rechner (leer; kein '+' !)

## Beispiel für die Datei `/etc/printcap`



```
#####
# Beispiel für /etc/printcap
#####
# Direkt angeschlossener Drucker
lw|ps|lp|PostScript:\
:lp=/dev/ttya:af=/usr/adm/lw.acct:\
:br#38400:rw:fc#0000374:fs#0000003:xc#0:xs#0040040:mx#0:sf:sb:\
:if=/usr/tran/sparc/lib/psif:of=/usr/tran/sparc/lib/psof:gf=/usr/tran/sparc/lib/psgf:\
:nf=/usr/tran/sparc/lib/psnf:tf=/usr/tran/sparc/lib/pstf:rf=/usr/tran/sparc/lib/psrf:\
:vf=/usr/tran/sparc/lib/psvf:cf=/usr/tran/sparc/lib/pscf:df=/usr/tran/sparc/lib/psdf:\
:sd=/usr/.spool/lw:lf=/usr/local/etc/logs/lpd.lw:
#####
# Direkt angeschlossener Drucker
pp|ps-pure|PostScript ohne Filter:\
:lp=/dev/ttya:af=/usr/adm/pp.acct:\
:br#38400:fc#0300:fs#06020:sh:\
:sd=/usr/spool/lpd/pp:lf=/usr/local/etc/logs/lpd.pp:
#####
# Remote Drucker
nv|nosve|NOS/VE-Ausgabe:\
:lp=:rm=NosVeLPD.LRZ-Muenchen.DE:\
:rp=NosVePrt:\
:sd=/usr/spool/lpd/nv:lf=/usr/local/etc/logs/lpd.nv:
```

## SysV - Spooling (1)



- ❑ neueres und mächtigeres Spooling-System, erlaubt aber keine remote-Spooling-Steuerung ⇒ Rückgriff auf BSD erforderlich
- ❑ Spool-Queue kann einer Gruppe von Druckern zugeordnet werden.
- ❑ Benutzer-Kommandos :
  - `lp (1)` ⇒ Jobs erzeugen
  - `lpstat (1)` ⇒ Infos über das Spooling-System / die Queues
  - `cancel (1)` ⇒ Jobs löschen
  - `lpalt (1)` ⇒ Job-Attribute nachträglich ändern
- ❑ Dämon : `lp sched (1M)`

Einführung in die Systemverwaltung unter UNIX

119

## SysV - Spooling (2)



- ❑ Administrator-Kommandos :
  - `lpshut (1M)`, `lpmove (1M)`, `lpfence (1M)`, `lpadmin (1M)`, `slp (1)`,  
`lpana (1)`, `enable (1)/disable (1)`, `accept (1M)/reject (1M)`
- ❑ interne Kommandos für Remote-Spooling :
  - `rlp (1M)`, `rlpstat (1M)`, `rcancel (1M)`, `rlpdaemon (1M)`
- ❑ Zugangs-Kontrolle :
  - `/usr/adm/inetd.sec` (nur bei HP)
  - `/usr/spool/lp/.rhosts`
  - `/etc/hosts.equiv` (sollte leer sein !)

Einführung in die Systemverwaltung unter UNIX

120

## Boot-Vorgang: Inhalt (Daniel Schedel)



- Bootvorgang auf dem PC
- Bootkonzepte
- Bootprozess bei SuSE
- Pre-Execution Environment (PXE)
- Intelligent Platform Management Interface (IPMI)

## Bootvorgang (PC) und Bootkonzepte (1)



- Nach Einschalten des Rechners Initialisierung von Bildschirm und Tastatur und Testen des Hauptspeichers durch das **Basic Input Output System (BIOS)**
- Auslesen von Informationen über aktuelles Datum, Zeit und wichtigste Peripherie aus den CMOS-Werten
- Laden des Betriebssystems von der ersten Festplatte
- Laden des ersten physischen Datensektors (512 Byte) in den Hauptspeicher → **Master Boot Record (MBR)**
- Bootsektoren sind jeweils erste Sektoren der Festplatten-Partitionen
- Bootsektoren nehmen Code für das Starten des Betriebssystems auf

## Bootvorgang (PC) und Bootkonzepte (2)



- ❑ Boot-Medien:
  - Diskette:
    - Bedingung: Bootfähiges Diskettenlaufwerk
    - Bootloader-Installation entfällt
  - USB-Speichermedien (wenn von BIOS unterstützt)
  - Netz
- ❑ Boot-Manager bei Installation von mehreren Betriebssystemen sinnvoll; z.B. `grub (8)` / `lilo (8)`
- ❑ **Extensible Firmware Interface (EFI)**
  - BIOS-Nachfolger
  - Mini-Betriebssystem von Intel
  - flexibler und schneller als BIOS
  - Schnittstelle zwischen Betriebssystem-Loader und Hardware

Einführung in die Systemverwaltung unter UNIX

123

## Boot-Prozess bei SuSE (1)



- ❑ `init (8)`:
  - Kernel übernimmt Kontrolle über die System-Hardware
  - Kernel mountet Root-Filesystem und startet Programm „`init`“
  - `init (8)` über `/etc/inittab` konfiguriert: `inittab (5)`
- ❑ Runlevels:
  - Definieren Systemzustand
  - Level 0: System angehalten (`halt`)
  - Level S: Einzelbenutzerbetrieb mit US-Tastaturbelegung
  - Level 1: Einzelbenutzerbetrieb mit lokaler Tastaturbelegung
  - Level 2: Mehrbenutzerbetrieb ohne Netz
  - Level 3: Mehrbenutzerbetrieb mit Netz
  - Level 5: Mehrbenutzerbetrieb mit Netz und grafischer Oberfläche
  - Level 6: Systemneustart (`reboot`)

Einführung in die Systemverwaltung unter UNIX

124

## Boot-Prozess bei SuSE (2)



- ❑ Init-Skripte im Verzeichnis `/etc/init.d`
  - Pro Runlevel ein eigenes Unterverzeichnis: `rc[S0-6].d`
    - `rc` = „resource control“ (Systemkontrolle)
    - `S, 0-6` = Runlevel
    - Enthalten nur Soft-Links, die auf `/etc/init.d/*` verweisen.
  - Innerhalb Runlevel-Verzeichnis Unterscheidung zwischen Start-Skripts (S) und Kill-Skripts (K)  
Beispiel: „`S07hotplug`“ Startskript für „hotplug“, „07“ Rang in der Reihenfolge
- ❑ Alle Init-Skripte besitzen genau ein Argument:
  - Immer möglich: `start, stop`
  - Oft möglich (siehe Usage): `restart, status, reload`

Einführung in die Systemverwaltung unter UNIX

125

## Boot-Prozess bei SuSE (3)



- ❑ Programm `insserve` (8):
  - Anlegen und Entfernen Runlevel-spezifischer Links
  - Datei `/etc/insserve.conf` wird ausgewertet, um Links entsprechend den Abhängigkeiten zu erstellen
- ❑ Verzeichnis `/etc/init.d/boot.d/*`:
  - Boot-Skripte
  - Laufen nur beim Boot-Vorgang, nicht beim Wechsel des Runlevels im laufenden Betrieb

Einführung in die Systemverwaltung unter UNIX

126

## Pre-Execution Environment (PXE) Intelligent Platform Management Interface



- Pre-Execution Environment (PXE):
  - Standardisiertes Feature von Intel für Netz-Chips
    - ➔ Boot-on-Net (DHCP + Bootp)
  - u.a. auch für Neuinstallation über Floppy emulierbar
- Intelligent Platform Management Interface (IPMI):
  - Genormte Schnittstelle zur Hardware (Setzen + Auslesen)
  - Im Allgemeinen mit Service-Prozessor
  - Power-On, Power-Off, Reset, Serial-over-LAN, Wake-Up-on-LAN
  - Im Allgemeinen nur auf Server-Boards
- ROM auf der LAN-Karte auch im Desktop-Bereich weit verbreitet (u.a. Wake-Up-on-LAN, Reset-on-LAN)

Einführung in die Systemverwaltung unter UNIX

127

## Wichtige UNIX-Kommandos: Übersicht



- Auswahl-Kriterien
- UNIX-Kommandos für wichtige Funktionsbereiche
- Anwendungsbeispiele und Tips
- Das Kommando `tar`
- Der Shell-Trace

Einführung in die Systemverwaltung unter UNIX

128

## Wichtige UNIX-Kommandos: Auswahlkriterien



- Ergebnis rein subjektiver Administrations-Erfahrung
- Ausgewählte Kommandos erfüllen  $\geq 1$  der folgenden Kriterien :
  - *häufige* interaktive Verwendung
  - *wichtige* interaktive Funktion
  - wichtige Funktion in *Shell-Skripten*
- Zusammenstellung wichtiger Kommandos, die nicht anderswo eigens behandelt werden ( selbstverständlich nicht vollständig ! )
- Kommandos nach Funktionsbereichen sortiert

Einführung in die Systemverwaltung unter UNIX

129

## UNIX-Kommandos für wichtige Funktionsbereiche (1)



### Einfache Textbearbeitung

cat (1), cut (1) / paste (1), uniq (1), sort (1), comm (1), pr (1)

### Komplexe Textmanipulation

sed (1), awk (1), cpp (1), m4 (1)

### Datenmanipulation

compress (1) / uncompress (1) / zcat (1),  
gzip (1) / gunzip (1) / gzcat, dd (1M), tr (1),  
expand (1) / unexpand (1),  
uuencode (1C) / uudecode (1C)

Einführung in die Systemverwaltung unter UNIX

130

## UNIX-Kommandos für wichtige Funktionsbereiche (2)



### Dateimanipulation

chgrp (1), chmod (1), rm (1), strip (1), chown (1), cp (1), mv (1),  
rcp (1), ln (1), mkdir (1), rmdir (1), tar (1), touch (1),  
vipw (1B), install (1M)

### Informationen über Dateien

view (1), diff (1), du (1M), ls (1), file (1), strings (1),  
sum (1), head (1), tail (1), wc (1), grep (1)

### Manipulation von Datei-Namen

find (1), basename (1), dirname (1), xargs (1), expr (1)

Einführung in die Systemverwaltung unter UNIX

131

## UNIX-Kommandos für wichtige Funktionsbereiche (3)



### Environment

env (1) / printenv (1B), groups (1), id (1M), stty (1), uname (1),  
which (1), who (1), whoami (1B)

### Shell-Builtins

set / unset, setenv / unsetenv, export, umask

### Benachrichtigung von Benutzern

write (1), wall (1M), rwall (1M), talk (1)

Einführung in die Systemverwaltung unter UNIX

132

## UNIX-Kommandos für wichtige Funktionsbereiche (4)



### Shutdown

`shutdown` (1M), `halt` (1M), `reboot` (1M), `fastboot` (1B),  
`fasthalt` (1B)

### Systemüberwachung

`uptime` (1), `who` (1), `w` (1), `finger` (1), `ps` (1), `top` (1L),  
`vmstat` (8), `iostat` (1), `nslookup` (8), `host` (1), `netstat` (8),  
`nfsstat` (8), `ping` (8)

### Sonstiges

`tee` (1), `eeprom` (1M), `true` (1), `false` (1), `nohup` (1),  
`date` (1), `make` (1), `rsh` (1), `ssh` (1), `script` (1)

Einführung in die Systemverwaltung unter UNIX

133

## UNIX-Kommandos Anwendungsbeispiele und Tipps (1)



- Eigene Datenbanken sollten unbedingt praktisch und nicht "schön" formatiert werden
  - Bearbeitung mit *einfachen* Text-Kommandos möglich
  - Varianten von Konfig.-Dateien möglichst aus 1 Quelle generieren
  
- Rekursive Dateimanipulation in einem Datei-Ast :
  - `-R` bei `chgrp` (1), `chmod` (1), `chown` (1), `ls` (1)
  - `-r` bei `rm` (1), `cp` (1), `rcp` (1)
  
- Datei-Attribute übernehmen ( z.B. Rechte, Zeitstempel ) :
  - `-p` bei `cp` (1), `rcp` (1)

Einführung in die Systemverwaltung unter UNIX

134

## UNIX-Kommandos Anwendungsbeispiele und Tipps (2)



- ❑ Optionen von `cat` (1):
  - `-n` Zeilen nummerieren
  - `-s` mehrere aufeinanderfolgende Leerzeilen durch eine einzige ersetzen
  - `-e, -t, -v` Ersatz-Darstellung für bestimmte Zeichengruppen
- ❑ BSD: `chown` (1) ändert *gleichzeitig* Owner und Gruppe  
SysV: für den gleichen Zweck `gchown` (1) verwenden
- ❑ Ausgabeformat eines Datums:  
"date +..." (hilfreich bei Shell-Skripten)

Einführung in die Systemverwaltung unter UNIX

135

## UNIX-Kommandos Anwendungsbeispiele und Tipps (3)



- ❑ Optionen von `diff` (1):
  - `-b, -w` unterschiedliche Folgen von Whitespace als gleich betrachten
  - `-c` Ausgabe von Kontext-Information
  - `-i` Groß- / Kleinschreibung ignorieren
- ❑ Zur Sicherheit `find` (1) *immer* mit der Option `-xdev` aufrufen, damit die Suche nicht in einer anderen Partition fortgesetzt wird.  
⇒ sonst sofort völliger Zusammenbruch des NFS-Servers
- ❑ Soft-Links erzeugen: `ln -s`
- ❑ "`ls -t`" sortiert nach Datum (jüngste Datei zuerst)

Einführung in die Systemverwaltung unter UNIX

136

## UNIX-Kommandos Anwendungsbeispiele und Tipps (4)



- ❑ `"mkdir -p"` erzeugt bei Bedarf auch Zwischen-Verzeichnisse
- ❑ Löschen von Dateien mit unkonventionellen Namen :  
( z.B. SPACE, TAB, nicht druckbare Zeichen, 1. Zeichen "-" )
  - interaktiv mit `rm -i *`
  - mit `rm './Name'`
  - mit `rm - 'Name'`
- ❑ `rm -f` löscht Dateien ohne Rückfrage ( keine Fehlermeldung, falls die Datei nicht existiert )  
⇒ praktisch in Make-Files

Einführung in die Systemverwaltung unter UNIX

137

## UNIX-Kommandos Anwendungsbeispiele und Tipps (5)



- ❑ `"ssh -n"` erwartet keine Eingabe
- ❑ `"tail -f"` verhindert den Abbruch am Datei-Ende  
⇒ hilfreich für Beobachtung wachsender Log-Dateien
- ❑ Ermittlung des Aufrufers einer Shell :  
( v.a. für Shell-Skripten; seltener interaktiv )
  - `who am i` liefert nur den eigenen Eintrag in `/etc/utmp` (falls vorhanden)
  - `whoami (1B)` liefert immer die 1. Kennung aus `/etc/passwd`, die die gleiche UID wie der Aufrufer besitzt

Einführung in die Systemverwaltung unter UNIX

138

## Das Kommando `tar` (1)



- Bei mittleren bis größeren Datenmengen ist `tar` (1) wesentlich effizienter und schneller als "`cp -r`" bzw. "`rcp -r`".
- expandiert keine Hard- und Soft-Links
- Tip: Datei nach Möglichkeit nie mit *absolutem Pfad* archivieren !
- Kopieren eines Datei-Astes :
  - innerhalb eines Rechners :

```
cd Destination_Dir
(cd Source_Dir; tar cfp - .) | tar xfp -
```
  - zwischen zwei Rechnern :

```
cd Destination_Dir
ssh Source_Host 'cd Source_Dir; tar cfp - .' | tar xfp -
```

## Das Kommando `tar` (2)



- Erzeugen von tar-Archiven auf Bändern :  
( Das Laufwerk ist an einen anderen Rechner angeschlossen )
  - Lesen des Band-Archivs :

```
cd Destination_Dir
ssh Host 'dd if=Device ibs=20b' | tar xvfp -
```
  - Schreiben des Archivs :

```
cd Source_Dir
tar cfp - . | ssh Host 'dd of=Device obs=20b'
```

## Shell-Trace



### □ Motivation :

- Überprüfung, was komplizierte Shell-Skripten tatsächlich tun
- Debugging

### □ vorhanden für `sh (1)`, `ksh (1)`, `bash (1)` und `[t]csh (1)`

### □ Optionen :

- `-v` ("verbose") protokolliert Kommandos, wie sie **gelesen** werden
- `-x` ("echo") protokolliert Kommandos, wie sie **ausgeführt** werden.

## Shell-Trace: Aktivieren / deaktivieren



|               | Option                             | Builtin                                    |   |
|---------------|------------------------------------|--|---|
|               |                                    | <code>[k]sh (1)</code>                     | <code>[t]csh (1)</code>                               |
| Aktivierung   | <code>-v</code><br><code>-x</code> | <code>set -v</code><br><code>set -x</code> | <code>set verbose</code><br><code>set echo</code>     |
| Deaktivierung |                                    | <code>set +v</code><br><code>set +x</code> | <code>unset verbose</code><br><code>unset echo</code> |