Linux

Systemübersicht

Kursbegleitende Lehrgangsunterlagen

Dr. Arthur Zimmermann

Verzeichnis

1. Geschichte

UNIX Geschichte:

1969 Ken Thompson und Dennis Ritchie von Bell Laboratories (AT&T) haben ein einfaches Betriebssystem für PDP-Rechner in Assembler geschrieben.

1973 Vollständig in C implementiert. UNIX wurde an die Universitäten weitergegeben, beginnt dynamische Entwicklung und Verbreitung.

1975 Erste kommerzielle Lizenzen.

1984 Kommerzielles System V.

Linux Geschichte:

1991 Linus Benedict Thorvalds studiert die Möglichkeiten des Intel-386-Prozessors mit Hilfe vom Betriebssystem minix. Entsteht ein kleines lauffähiges Betriebssystem. Linus Thorvalds bietet seine Quelltexte frei an.

1992 Stabil laufender Kernel (Version 0.12). Die Quelltexte verbreiten sich weltweit per Internet. Durch POSIX-Standard und umfangreiche C-Bibliotheken laufen fast alle freien Programmen unter Linux.

2. Linux ohne X

2.1. Grundbegriffe des Betriebssystems

UNIX-ähnliche Betriebssysteme sind durch folgende Eigenschaften ausgeprägt:

- Multi-User-Betrieb;
- Multi-Tasking-Betrieb;
- interaktiver Betrieb;
- Netzwerk-Betrieb;
- verteiltes Betriebssystem;
- hohe Leistungsfähigkeit;
- hohe Sicherheit;
- virtuelle Speicherverwaltung;
- Hardware-Unabhängigkeit;
- großes Angebot von Standard-Software.

2.2. Dateisystem

Dateisystem unter UNIX spielt zentrale Rolle, da alle Aktivitäten durch es vorgehen. Dateisystem hat hierarchische Struktur. Im Gegensatz zum MS-DOS hat der Begriff des Dateisystems ganz andere (umgekehrte) Bedeutung:

> Nicht das Dateisystem sich auf der Festplatte befindet, sondern die Festplatte hängt an dem Dateisystem.

Somit beinhaltet Dateisystem auch Peripherie-Geräte, virtuelle Systeme u.s.w.

Es gibt folgende Typen von Dateien:

- normale Dateien (ausführbare und nicht ausführbare);
- Verzeichnisse;
- Pipelines und Sockets;
- Gerätedateien (Block-Devices und Character-Devices).

Fast alle Betriebssysteme unterstützen nur verwandte Dateisysteme. Linux aber - fast alle vorhanden Dateisysteme. Linux unterstützt folgende Dateisysteme:

- ext2 - natives Dateisystem;
- minix;
- xiafs;
- NFS (Sun);
- FAT, VFAT (MS-DOS);
- HPFS (OS/2 - readonly, da Spezifikationen nicht freigegeben worden sind);
- ISO9660 (CD-ROM);
- FFS (Amiga);
- UFS (SunOS, BSD, NeXTstep);
- NCP (Novell Netware);
- u.v.m.

Alle diese Dateisysteme können als Verzeichnisse in den Verzeichnisbaum von Linux montiert werden. Das heißt, daß praktisch die Daten von jedem Betriebssystem benutzt werden können.

2.3. Erste Arbeit mit Linux

2.3.1. Ein- und Ausloggen

Klein- und Großschreibung wird immer unterschieden. Begriffe: Login, Paßwort, Home-Verzeichnis.

```
# who                  wer ist angemeldet
# who -H               mit Überschrift
# whoami               Wie bin ich angemeldet
# who am i             Wie bin ich angemeldet (ausführlich)
# finger [user]        ausführliche Info über angemeldete Benutzer
# pwd                  print work directory
# date                 System-Datum und -Uhrzeit anzeigen
# date -s mm/dd/yyyy   Datum einstellen
# date -s hh:mm:ss     Uhrzeit einstellen
# date -d mm/dd/yyy    Datum und Uhrzeit anzeigen
# clock                Datum und Zeit aus CMOS anzeigen
```

2.3.2. Hilfe

```
# man befehl             Hilfe über befehl
# man man                Hilfe über man
# man -k key             in der Hilfe nach key suchen
# apropos key            in der Hilfe nach key suchen
# befehl --h             kurze Hilfe (ggf)
```

2.3.3. Arbeiten mit Verzeichnissen

Namensgebung: 255 Zeichen, darf man fast alle Zeichen (auch Leerzeichen) benutzen, aber empfohlen: Buchstaben, Ziffer, Punkte, Unterstrichzeichen, z.B.:

Empfohlene Namen:

`.finanz.edv`
`Briefe_an_Finanzamt_077`
`A2OK.Verwaltung.Sicherheitsabteilung`

Nicht empfohlenen Namen:

`-r` (oft wird als Option verwendet)
`+4` (wird als Option verwendet)
`/meine/tabelle` (wird mit Verzeichnissen verwechselt)

Verzeichnis-Struktur: Verzeichnisse enthalten Unterverzeichnisse und/oder Dateien.

⇒ Hauptverzeichnis: **`/`**
⇒ Eigenes Verzeichnis vom Superuser (root): **`/root`**
⇒ Eigenes Verzeichnis vom normalen Benutzer: **`/home/user01`**

```
# mkdir verz            Verzeichnis erstellen
# cd [verz]             ins Verzeichnis verz wechseln
# cd                    ins Home-Verzeichnis wechseln
# rmdir verz            Verzeichnis löschen
# ls -opt [verz]        Verzeichnisinhalt anzeigen (beschreiben!)
# dir [verz]            Verzeichnisinhalt anzeigen
```

2.3.4. Arbeiten mit Dateien

Datei erstellen - Umlenkungen:

```
# ls -a | more          langes Verzeichnis anzeigen
# ls -a | less          viel bequemer!
# >                     Datei neu erstellen
# >>                    an den bestehenden Inhalt anfügen
# date > dat            Datei dat enthält Datum
# dir > dat             Datei dat hat Verzeichnisinhalt
# echo text             text auf den Bildschirm ausgeben
# echo $VAR             Variable $VAR auf den Bildschirm ausgeben
# echo `befehl`         einen Befehl ausführen
# echo -e text          Steuerungszeichen im text ausführen:
                        \a alarm
                        \b backspace
                        \c kein Zeilenende
                        \f form feed
                        \n Zeilenende
                        \t horisontaler Tabulator
                        \v vertikaler Tabulator
# echo text > dat       Datei dat erstellen
# more dat              Datei dat anzeigen
# less dat              Datei dat anzeigen
# cat dat               Datei dat anzeigen
# cat -n dat            - mit Zeilennummern anzeigen
# cat -s dat            - mehrere leere Zeile durch eine ersetzen
# cat dat1 dat2 > dat3  Dateien dat1 dat2 zusammenführen in dat3
# rm dat                Datei dat löschen
# du                    wieviel Platz belegen Verzeichnisse
# df                    wie sind die Dateisysteme belegt
# free                  Auskunft über Hauptspeicher
```

2.4. Wichtige Verzeichnisse

/bin	Ausführbare Dateien vom Betriebssystem.
/boot	Dateien vom Boot-Manager LILO (Boot-Sektoren von anderen Betriebssystemen, Master Boot Record).
/dev	Gerätedateien.
/etc	Konfigurationsdateien für das Betriebssystem.
/home	Verzeichnisse für jeden „normalen“ Benutzer.
/lib	Libraries.
/mnt	Verzeichnisse, auf die externe Geräte oder Dateisysteme aufgesetzt (montiert) werden.
/opt	Kommerzielle Software.
/proc	Prozeßdateisystem - Kernelinformationen in der Form von Dateien.
/root	Dateien vom Systemverwalter (Superuser).
/sbin	Ausführbare Dateien für Systemverwaltung.
/usr	Alle Anwendungen (statische Daten).
/usr/man	Hilfetexte.
/usr/src	Quellcode für alle Programme des Standardsystems.
/usr/X11R6	X Window System.
/var	Alle Anwendungen (Daten, die während des Betriebs geändert werden können oder müssen).

2.5. Wichtige Dateien

/vmlinuz	Kernel.
/etc/fstab	Dauerhafte Parameter für die Zusammensetzung des Dateisystems.
/etc/mtab	Welche Dateisysteme tatsächlich gemountet sind.
/etc/hosts	Konfigurationsdatei des TCP/IP Netzwerkes. Hier sind IP-Adressen und Host-Namen eingetragen.

/etc/inittab	Beschreibung der zu startenden Prozesse (wird vom Programm *init* benutzt).
/etc/man.config	Laufzeitkonfiguration des Hilfesystems.
/etc/passwd	Hier sind die Benutzer eingetragen.
/etc/group	Hier sind die Gruppen und ihre Mitglieder eingetragen.
/etc/issue	Information auf dem Bildschirm vor dem Anmelden.
/etc/motd	Information auf dem Bildschirm nach dem Anmelden.
/etc/printcap	Beschreibung des Druckers.
/etc/profile	Grundeinstellungen für alle Benutzer, wird von Login-Shell des Benutzers gelesen und ausgeführt (*autoexec.bat*).
/etc/rc*	Initialisierungsdateien für das System (werden von *init* dem Programm *shell* übergeben).

2.6. Erweiterte Arbeit mit Dateien und Verzeichnissen

2.6.1. Struktur des Dateisystems

Festplatte ist eine Kette von Blöcken (1024 Byte). Aufbau des Linux-Dateisystem **minix**:

- Boot-Sektor. Hier befindet sich das Programm, das Linux oder anderes Betriebssystem startet.
- Superblock. Hier werden gespeichert: Typ des Betriebssystems, Größe des Dateisystems (Größe der Inode-Liste + Anzahl der Datenblöcke).
- I-Bitmap (>=1024 Byte). Jedem Bit entspricht ein Eintrag in der Inode-Liste. `1` - Eintrag wird benutzt, `0` - Eintrag ist frei.
- D-Bitmap (>=1024 Byte). Jedem Bit entspricht ein Datenblock. `1` - Datenblock wird benutzt, `0` - Datenblock ist frei.
- Inode-Liste. Jeder Eintrag beschreibt eine Datei (s.u.).
- Datenblöcke.

Aufbau der Inode-Liste:

- UID, Besitzer der Datei;
- GID, Gruppe, zu der die Datei gehört;
- Anzahl der Links (Verweiszähler);
- Größe der Datei;
- Typ (**-**, **d**, **l**, **b**, **c**), Rechte (**rwxrwxrwx**);
- Datum, Uhrzeit;
- Direkte Verweisfelder 1-7, sie enthalten die Nummer des Datenblocks, wo sich die Daten der Datei befinden;
- Indirektes Verweisfeld (1. Grad), es hat die Nummer des Datenblocks, wo sich die Verweise auf Datenblöcke der Datei befinden;
- Indirektes Verweisfeld (2. Grad), es hat die Nummer des Datenblocks, wo sich die Verweise auf die Datenblöcke befinden, die auf die Daten der Datei zeigen.

Verzeichnis ist eine Datei von Typ **d**, die Einträge über entsprechende zugehörige Dateien hat. Nur bestimmte Programme, die diese Struktur kennen, können auf den Inhalt des Verzeichnisses zugreifen (**cat** hat keinen Zugriff!). Jeder Eintrag beinhaltet:

- Dateiname;
- Inode-Nummer (die Reihenfolgenummer des Eintrages in der Inode-Liste).

Wenn eine neue Datei zu erstellen ist, ermittelt das Betriebssystem anhand der I-Bitmap und D-Bitmap freie Inode-Nummer und Datenblöcke. Wird Datei ohne Fehler angelegt, so werden die Bits in I-Bitmap und D-Bitmap auf **1** gesetzt und wird entsprechender Eintrag im Verzeichnis gemacht.

Wenn eine Datei mit einem bestimmten Namen zu finden ist, wird zuerst der Eintrag aus dem Verzeichnis gelesen. Es wird die entsprechende Inode-Nummer ermittelt. Dann wird die Zeile mit dieser Nummer aus der Inode-Liste gelesen, und ab jetzt kann der Inhalt der Datei gelesen werden.

Wenn eine Datei, für die die Anzahl der Links = 1 ist, zu löschen ist, dann werden die entsprechenden Bits in I-Bitmap und D-Bitmap auf 0 gesetzt und der Eintrag in Inode-Liste wird frei (so daß er für eine neue Datei vergeben werden kann). Werden die Daten in Datenblöcken tatsächlich gelöscht oder nicht ist von Sicherheitslinien abhängig. Ist die Anzahl der Links > 1, dann wird Anzahl der Links auf 1 vermindert, so daß Inhalt der Datei und I-Bitmap und D-Bitmap bleiben unverändert.

Wichtige Erweiterungen von **ext2fs**:

- 12 Felder für direkte Adressierung;
- indirekte Adressierung der 3. Grad;
- 3 reservierte Felder;
- I-Bitmap und D-Bitmap sind durch verkettete Liste ersetzt;
- Reihenfolge der Felder hat sich geändert;
- Zeitmarken im POSIX-Format.

Die Festplatte oder Partition kann bis zu 2 GByte groß sein (Erweiterung bis zu 4 TByte ist vorgesehen). Eine Datei kann bis zu 16 GByte groß sein.

2.6.2. Links

Daraus ist klar, daß nur die Inode-Nummer die eindeutige Identifizierung der Datei gewährleistet. Folgendlich, eine Datei kann mehrere Namen haben.

> Harter Link - nur ein Eintrag im Verzeichnis. Dieser Eintrag hat den Typ **-** wie gewöhnliche Datei, einen anderen Namen und dieselbe Inode-Nummer. Anzahl der Links in der Inode-Liste wird inkrementiert (um 1 vergrößert).
>
> Symbolischer Link - eine neue Datei, die den Typ **l** hat und vollen Namen der Original-Datei enthält. Die Inode-Nummer dieser neuen Datei unterscheidet sich von der der Original-Datei. Anzahl der Links in dem Inode-Eintrag der Original-Datei bleibt unverändert.

Der Dateiname ist der erste harte Link auf Datei.

Harter Link auf Verzeichnis ist (fast) nicht möglich (nicht gestatten). Symbolischer Link auf Verzeichnis ist möglich.

```
# ln q-dat z-dat        harter Link z-dat für q-dat
# ln -s q-dat z-dat     symbolischer Link z-dat für q-dat
# ls -i                 Inode-Nummer anzeigen
# ls -l                 Anzahl der Links anzeigen (zweite Spalte)
```

2.6.3. Pipelines

Jeder Prozeß bekommt 3 Kanäle: Eingabe (0), Ausgabe (1), Fehlermeldungen (2).

Pipeline - Output von einem Programm dient als Input für das nächste:

```
# ls -a | more          langes Verzeichnis anzeigen
# ls -a | less          viel bequemer!
```

2.6.4. Dateien kopieren, verschieben und löschen

```
# cp q-dat z-dat        kopieren q-dat in die z-dat
# cp -r q-verz z-verz   kopieren von Verzeichnissen
# mv q-dat z-dat        verschieben/umbenennen q-dat in die z-dat
                        (auch Verzeichnisse!)
# mv -b q-dat z-dat     Backup erstellen
# mv -f q-dat z-dat     überschreiben ohne zu fragen
```

2.6.5. Dateien und Inhalte suchen

Ersetzungszeichen

```
?                       Stellvertreter für ein Zeichen
*                       Stellvertreter für alle Zeichen
[]                      Liste der Zeichen, die erlaubt sind
{,}                     Liste der Zeichen, die zu expandieren sind
z.B.
# ls dat?
# ls -a -d dat*         nützlich, wenn es um Verzeichnisse geht
# ls dat[273]
# ls d?t[A-F]
# mkdir {a,b}qqq{c,d}
```

Reguläre Ausdrücke

Regel 1: Kein Suchmuster wirkt über ein Zeilenende hinaus.

Regel 2: Paßt ein Suchmuster zu mehreren Zeichenfolgen einer Zeile, so wird die längste (und bei mehreren - die erste) als Treffer angesehen.

Regel 3: Jedes ASCII-Zeichen ist regulär außer *\ [ ] . - $ ^*

Regel 4: Sind *r1* und *r2* regulär, so ist *r1r2* auch regulär (kein Trennzeichen).

Regel 5: Ein Backslash \ entwertet die Sonderbedeutung dieser Zeichen, z.B. Suchmuster *3\.14* paßt zu *3.14* in der Zeile.

Regel 6: Ein Punkt . paßt zu einem beliebigen Zeichen in der Zeile, z.B. Such-muster *a.z* paßt zu *abz*, *acz*, *ayz* in der Zeile.

Regel 7: Suchmuster *[s]* paßt zu jedem Zeichen, das in *s* vorkommt. Suchmuster *[^s]* paßt zu jedem Zeichen, das in *s* nicht vorkommt. z.B. Suchmuster *[ML]aus* paßt zu *Maus* und *Laus* in der Zeile.

Regel 8: Suchmuster *r** paßt zu allen Zeilen, wo *r* nullmal oder mehr vorkommt, z.B. Suchmuster *a** paßt zur Leerzeile, zu *a*, zu *aa*, usw. in der Zeile.

Regel 9: Suchmuster *^r* paßt zu einem regulären Ausdruck am Anfang der Zeile, Suchmuster *r$* paßt zu einem regulären Ausdruck am Ende der Zeile.

Regel 10: Runde Klammer müssen mit dem Backslash \ entwertet werden, z.B. Suchmuster *ha\(ha\)** paßt zu *ha*, *haha*, *hahaha* usw. in der Zeile.

```
# find verz -name dat   suchen Datei dat im verz
# find verz -type t     suchen alle Dateien vom Typ t
# find verz -user name  Dateien gehören dem Benutzer name
# find verz -group name Dateien gehören der Gruppe name
# find verz -nouser     Dateien gehören keinem Benutzer
# find verz -nogroup    Dateien gehören keiner Gruppe
# find verz -used tage  genau tage nach der letzten Änderung
# find verz -used +tage mehr als tage nach der letzten Änderung
# find verz -atime N    auf die Dateien wurde vor N*24 Stunden zu-
                        gegriffen
# find verz -inum N     Dateien mit der Inode-Nummer N
# find verz -links N    Dateien haben N (hard-)Links
# find verz -lname name Dateien sind symbolische Links auf name
# find verz -size N     Dateien mit der Größe N
# find verz -maxdepth N
# find verz -mindepth N
# find verz -opt... -exec less {}\;
# rm -r dat             rekursives Löschen der Dateien und Ver-
                        zeichnissen (sehr gefährlich)
# rm -i dat             löschen mit Bestätigung
# del dat               löschen nur Dateien mit Bestätigung
# grep muster dat       alle Zeilen der Datei dat anzeigen, die
                        den Muster muster haben (wenn die Leerzei-
                        chen im Muster vorhanden sind, muß man  ""
                        oder '' verwenden)
# less dat
# cat dat
```

Alle Daten (Dateien und Unterverzeichnisse, normale und versteckte) von einem Verzeichnis in das andere kopieren:

```
# cp -r /etc/skel/{.[^.]*,*} /home/user1
```

2.7. Editieren von Dateien mit vi

Starten:

```
# vi [datei]
# vi -v datei          readonly
# vi -R datei          readonly
```

Editor **vi** arbeitet in folgenden Modi:

⇒ **Normal Mode.** Fast jedem Befehl kann eine positive Zahl vorangestellt werden (Anzahl der Wiederholungen).

h j k l	Kursor nach links, oben, unten, rechts;
H M L	Kursor in die erste, mittlere, letzte Zeile plazieren;
x (klein)	ein Zeichen löschen (wie Taste *Entf*);
X (groß)	ein Zeichen löschen (wie Taste *Backspace*);
D	die Zeichen bis zum Ende der Zeile löschen;
dd	eine ganze Zeile löschen;
p (klein)	Inhalt der Zwischenablage rechts vom Kursor einfügen;
P (groß)	Inhalt der Zwischenablage links vom Kursor einfügen;
u	vorherigen Befehl rückgängig machen;
.	vorherigen Befehl wiederholen;
:	ins Command Mode umschalten;
i a R	ins Text Mode;
v	ins Visual Mode;

⇒ **Command Mode**. Zum Öffnen und Speichern der Dateien, zum Beenden des Editors. Nach dem Befehl oder Taste *Esc* kehrt man zurück zum Normal Mode.

e *name*	Datei *name* öffnen;
w	Inhalt in der geöffneten Datei speichern;
w *name*	Inhalt in der Datei *name* speichern;
q	Editor beenden, wenn nichts geändert wurde;
wq	Inhalt speichern und Editor beenden;
q!	Editor beenden ohne zu speichern.

⇒ **Text Mode.** Man kann ganz normal den Text editieren. Nach der Taste *Esc* kehrt man zurück zum Normal Mode.

⇒ **Visual Mode.** Man kann den Text mit den Kursortasten markieren und in die Zwischenablage plazieren. Nach den Befehlen **y** und **d** oder Taste *Esc* kehrt man zurück zum Normal Mode.

b	bis zum Anfang der Zeile markieren;
e	bis zum Ende der Zeile markieren;
y	markierten Text in die Zwischenablage kopieren;
d	markierten Text ausschneiden und in die Zwischenablage platzieren.

2.8. Auf- und Absetzen von Festplatten und Geräten

Dateisysteme müssen gemountet werden

```
#/etc/fstab
# static information about file system
/dev/hda3   /               ext2       defaults              1   1
none        /proc           proc       defaults              0   0
/dev/fd0    /mnt/Disk_A     msdos      rw,noauto,user        0   0
/dev/hdc    /cdrom          iso9660    ro,noauto,user,exec   0   0
# End of fstab
```

Allgemein:

```
# mount -t fs Gerätedatei Verzeichnis
# umount Verzeichnis
```

Wenn Dateisystem in der Datei **/etc/fstab** eingetragen ist:

```
# mount Gerätedatei
# mount Verzeichnis
# mount -a
```

Wenn Dateisystem in der Datei **/etc/fstab** nicht eingetragen ist:

```
# mount -t fs Gerätedatei Verzeichnis
```

Z.B. MS-DOS-Diskette mounten:

```
# mount -t msdos /dev/fd0 /mnt/Diskette_A
```

Z.B. minix-Diskette mounten:

```
# mount -t minix /dev/fd0 /mnt/Diskette_A
```

Z.B. CD-ROM mounten (zweiter Kontroller, Master):

```
# mount -t iso9660 /dev/hdc /mnt/CDROM
```

2.9. Druckerbetrieb

Gerätedateien:

Datei	DOS-Name	IRQ	Adress
/dev/lp1	LPT1	7	0x378-0x37A
/dev/lp2	LPT2	5	0x278-0x27A
/dev/lp0	LPT3	5	0x3BC-0x3BE

Polling- und Interrupt-Betrieb.

Druckwarteschlangen (Betrieb und Konfiguration). Druck-Manager **lpd**. Die Datei **/etc/printcap** steuert das Verhalten des Druckers. Alle Rechner, die den Zugriff auf localen Drucker haben sollen, müssen in der Datei **/etc/hosts.equiv** oder in der Datei **/etc/hosts.lpr** aufgeführt werden.

```
# lpr         übergibt den Dokument an den Druckerdämon
# lpq         zeigt den aktuellen Status der Warteschlange
# lprm        löscht den abgeschickten Druckjob
# lpc         steuert den Druckerdämon (disable, abort, down)
```

In der Datei **/etc/apsfilter** sind die Informationen für die Ausgabe von Dokumenten unterschiedlicher Formaten gespeichert.

2.10. Benutzer- und Gruppenverwaltung

2.10.1. Benutzerverwaltung

Benutzerdateien **/etc/passwd** und **/etc/shadow**:

Feld 1: Benutzername;
Feld 2: Benutzerpaßwort;
Feld 3: Benutzernummer;
Feld 4: Gruppennummer;
Feld 5: Kommentar;
Feld 6: Arbeitsverzeichnis;
Feld 7: Name des nach Einloggen zu startenden Befehlsinterpreters oder des Programmes.

Einen neuen Benutzer als **root** anlegen:

- Eintrag in **/etc/passwd** und in **/etc/shadow** mit **vi** machen;
- Home-Verzeichnis mit **mkdir** erstellen;
- alle Daten aus **/etc/skel** ins Home-Verzeichnis kopieren;
- mit **chown** in Besizt des neuen Benutzers übergeben;
- das Paßwort mit **passwd** eingeben.

Benutzer anlegen, ändern, löschen.

```
useradd     neuen Benutzer erstellen
usermod     vorhandenen Benutzer ändern
userdel     Benutzer löschen
passwd      Paßwort für Benutzer ändern
chown       Dateien in Besitz übergeben (als root)
groups      in welche Gruppen kann aktueller Benutzer wechseln
newgrp      in die neue Gruppe wechseln
chfn        Name, Office, Rufnummer ändern (Daten für finger)
```

2.10.2. Gruppenverwaltung

Gruppendateien **/etc/group** und **/etc/gshadow**

Feld 1: Gruppenname;
Feld 2: Gruppenpaßwort;
Feld 3: Gruppennummer;
Feld 4: Liste der Mitglieder durch Komma getrennt.

Gruppen anlegen, ändern, löschen. Änderungen in Gruppendateien beobachten.

```
groupadd    neue Gruppe erstellen
groupmod    vorhandene Gruppe ändern
groupdel    Gruppe löschen
gpasswd     Paßwort für die Gruppe eingeben (damit können auch die
            Benutzer in diese Gruppe wechseln, die zu der Gruppe
            nicht gehören)
```

2.11. bash-Programmierung

bash ist ein Befehlsinterpreter unter Linux. Er wird standardmäßig gestartet, wenn sich der Benutzer beim System angemeldet hat. Alle Befehle in der Eingabeaufforderung werden vom bash ausgeführt. bash-Script ist eine Textdatei mit dem Recht **x**, die Befehle und Steuerungskonstruktionen enthält und auch vom bash ausgeführt wird.

Auf die Argumente eines Scriptes kann durch spezielle Variablen zugegriffen werden:

$0 - das nullte Argument - Name des Scriptes;

$1, $2, ..., ${12} - das erste, das zweite, das zwölfte Argument u.s.w.;

$# - Anzahl der Argumenten;

$? - Errorlevel des vorherigen Befehls;

$name - Inhalt der Variable name;

name=wert - Zuweisung des Wertes wert der Variable name.

Es gibt folgende Steuerungskonstruktionen:

```
#=====================
if bedingung
then
          befehle
else
          befehle
fi
#=====================
while bedingung
do
          befehle
done
#=====================
until bedingung
do
          befehle
done
#=====================
for Var in Liste
do
          befehle
done
#=====================
case Var in
          Wert1) befehle;;
          Wert2) befehle;;
          *) befehle;;
esac
#=====================
```

Die Bedingungen können so aussehen:

```
[ $var -gt 0 ]
[ "$var" = "text" ]
```

2.12. Zugriffsrechte

Benutzergemeinschaft besteht aus:

- Benutzer;
- Gruppe von Benutzern;
- restlichen Benutzern.

Jeder Benutzer soll mindestens zu einer Gruppe gehören.

Objekte: Dateien und Verzeichnisse.

Wer ein Objekt angelegt hat, wird Eigentümer des Objektes. Das Objekt wird der Gruppe zugeordnet, in der der Eigentümer sich zum Zeitpunkt des Anlegens befindet. Der Eigentümer und die Gruppe des Objektes können zu jeder Zeit geändert werden.

Der Eigentümer legt fest die Zugriffsrechte für:

- sich selbst;
- die Gruppe;
- restliche Benutzer.

Der Benutzer kann auf das Objekt zugreifen, wenn:

- er der Eigentümer des Objektes ist, oder
- er zur Gruppe gehört, der das Objekt zugeordnet ist, und das Zugriffsrecht dieser Gruppe gewährt ist, oder
- das Zugriffsrecht den restlichen Benutzern gewährt ist.

> Die Rechte des Superusers (**root**) können nicht eingeschränkt werden.

```
# ls -l
```

Dateityp:

d	Verzeichnis;
l	symbolischer Link;
-	normale Datei;
b	Block-Device;
c	Character-Device.

Bedeutung für	**r**	**w**	**x**
Datei	Inhalt kann gelesen werden	Inhalt kann geändert werden	Datei kann ausgeführt werden
Verzeichnis	Inhalt kann aufgelistet werden	Dateien können angelegt und gelöscht werden	In das Verzeichnis kann gewechselt werden

> Kann eine Datei gelöscht werden oder nicht, ist nicht von der Datei, sondern vom Verzeichnis abhängig.

Zugriffsrechte ändern:

```
# chmod modus dat/verz
modus:                Benutzerklass    Operator       Recht
                      u (user)         + (add)        r (read)
                      g (group)        - (remove)     w (write)
                      o (other)        = (set)        x (execute)
                      a (all)
z.B.:
# chmod go-rx /tmp/temp
# chmod 751   /tmp/temp
# umask xyz       777-xyz sind Standardrechte für neues Verzeichnis
                  666-xyz sind Standardrechte für neue Datei
```

Zugriffsrechte als dreistellige Oktalzahl: erste Ziffer - Rechte vom Eigentümer, zweite - von der Gruppe, dritte - von restlichen Benutzern. Für jede Ziffer gilt: **r** heißt 4, **w** heißt 2, **x** heißt 1, **-** heißt 0; **rwx** heißt **r+w+x**, z.B. **rwx** heißt 4+2+1=7, **rw-** heißt 4+2+0=6, **r-x** heißt 4+0+1=5, **rwxrwxrwx** heißt 777, **rwxr-x--x** heißt 751. Beim Anlegen bekommt Datei voreingestellte Rechte, die mit **umask** geändert werden können. Bei Änderungen durch **chmod** wird **umask** auch berücksichtigt.

> Rechte auf symbolische Links können nicht geändert werden.

Eigentümer und/oder Gruppe ändern:

```
chown [-R] new-owner[.new-group] dat
       -R  Besitzer/Gruppe in allen Verzeichnissen rekursiv ändern
chgrp [-R] new-group dat
       -R  Gruppe in allen Verzeichnissen rekursiv ändern
```

> Eigentümer und/oder Gruppe können nur vom aktuellen Eigentümer oder Superuser geändert werden.

Spezielle Zugriffsrechte für Dateien und Verzeichnissen: *Set-User-ID-Bit*, *Set-Group-ID-Bit* und *Sticky-Bit* werden im nächsten Abschnitt besprochen.

2.13. Prozesse und Prozeßverwaltung

2.13.1. Begriff des Prozesses

Prozeß = Programme + Umgebung (Verwaltungsinformation). Also, zum Prozeß gehören:

- Programmcode;
- Speicherbelegung;
- Registerinhalte;
- offene Dateien;
- Benutzer;
- Gruppe;
- Prozeßvariablen (**HOME**, **PATH**, u.s.w.);
- aktuelles Verzeichnis.

1 Aufruf vom Kommando = 1 Prozeß

Jeder Prozeß bekommt eine PID und PPID (Parent PID - PID des Vater-Prozesses).

Prozesse sind hierarchisch aufgebaut: Vater-Prozeß, Sohn-Prozeß. Systemaufrufe:

- **fork**
- **exec**

```
ps       momentane Liste der eigenen Prozesse
ps -a    Liste aller Prozesse
ps -f    umfangreiche Information über eigenen Prozesse
ps -u    Userinformation
ps -j    Jobinformation
ps -l    langes Format
ps -e    Liste aller Prozesse
top      immer aktuelle Information über Prozesse
```

2.13.2. Prozesse beim Hoch- und Herunterfahren

Kernel wird vom Boot-Loader in den Speicher geladen.

Kernel wird dekomprimiert und an die richtige Stelle verschoben.

Einige Funktionen und Tabellen des Kernels werden initialisiert: Speicherseitenverwaltung, IRQ, Systemzeit.

Parameter werden dem Kernel übergeben.

Systemconsole wird eingerichtet.

PCI-Subsystem wird initialisiert.

Virtuelles Dateisystem im Kernel wird initialisiert.

Geräteunabhängige Netzwerkschichten (IP) werden initialisiert.

Es wird Kernel-Funktion **`init`** (Thread) gestartet. Sie teilt den Speicherbereich mit anderen Funktionen.

Dieser Thread **`init`** erzeugt weitere Threads zum Unterstützung des virtuellen Speichers und Cash.

Es wird Kernel-Funktion **`setup`** gestartet. Sie ruft Gerätetreiber nach einander, die erfolgreichen Gerätetreiber registrieren sich beim Kernel.

Die Partitionen werden geprüft, Dateisysteme gemountet, so daß Dateien von der Festplatte geladen werden können.

Es wird erster Prozeß erzeugt, indem Thread **`init`** die Datei **`/sbin/init`** in den Speicher lädt und ausführt. Dieser Prozeß heißt auch **`init`** und er befindet sich im Adreßraum, der sich von dem des Kernels unterscheidet (Userspace). Er erstellt alle weitere Prozesse. Zuerst wird Datei **`/etc/inittab`** gelesen, die Einstellungen für virtuelle Terminals beinhaltet und voreingestellten Runlevel, dann werden virtuelle Terminals mit einem Prozeß **`getty (mingetty)`** belegt, der auf Anmeldung wartet.

Runlevels (Betriebsarten des Betriebssystems):

`0`	halt;
`1`	ohne Netz;
`2`	mit Netz;
`3`	mit Netz + **`xadm`**;
`4`, **`5`**	nicht verwendet;
`6`	reboot;
`S`, **`s`**	single user mode.

```
# init runlevel
```

Eintrag in **`inittab`** ist folgendermaßen aufgebaut:

ID:Runlevel:Aktion:Prozeß

ID:	Identifikator der Zeile (in Fehlermeldungen verwendet);
Runlevel:	Bezeichnung von Betriebsart;
Aktion:	vordefinierte Bezeichnung der Aktion;
Prozeß:	Programm, das zu starten ist.

Beispiel: mit *Alt*+*Strg*+*Entf* den Editor **`vi`** starten.

Das System herunterfahren:

```
# init 0
# init 6
# shutdown -h          System herunterfahren und aufhalten
# shutdown -r          System herunterfahren und neu starten
# shutdown -h now      System sofort herunterfahren
# shutdown hh:mm       System zu dieser Zeit herunterfahren
# shutdown +mm         System in mm Minuten herunterfahren
```

2.13.3. Befehle zur Prozeßverwaltung

Signale:

```
kill -l            Liste aller Signale
kill -9 PID        Prozesse mit der Nummer PID beenden
```

Time-Sharing, Zeitscheibe, Prioritäten (**nice**), Dämon, Swap, Vorder- und Hintergrundprozesse (**&**, **fg**, **bg**, u.a.).

Spezielle Zugriffsrechte für Dateien und Verzeichnissen:

Set-User-ID-Bit;
Set-Group-ID-Bit;
Sticky-Bit.

Grund: Jeder Prozeß startet mit den Rechten des aktuellen Benutzers (nicht Besitzers!).

Sinn: Bestimmte Dateien sind nur mit bestimmten Programmen von beliebigen Benutzer zu ändern. Unbedingt Beispiel: Datei **/etc/passwd** nur mit **/bin/passwd** von jedem Benutzer geöffnet werden muß (die Zugriffsrechte von diesen Dateien mit **ls -l** anzeigen). Im Rahmen der bisherigen Rechten ist das Problem nicht lösbar. Wenn ein Prozeß ein Programm abarbeitet, für das *Set-User-ID-Bit* gesetzt ist, dann gelten die Zugriffsrechte des Besitzers für den Prozeß. Dasselbe gilt für *Set-Group-ID-Bit*. *Sticky-Bit* verwaltet das Ein- und Auslagern von Programmen im Hauptspeicher; außerdem die Dateien, auf die dieses Bit gesetzt wurde, können nur vom Besitzer gelöscht werden.

Aber: Bei Scripts wird *Set-User-ID-Bit* nicht ausgewertet, nur bei Programmen (Unix).

2.14. Editieren von Dateien mit emacs

2.15. S.u.S.E-Distribution von Linux

2.15.1. YaST

2.15.2. Benutzeroberflächen

mc, **git**,

2.15.3. Editoren

jove, elvis, edy

3. Netzwerk

3.1. Grundlage

Kommunikation geht auf Grund des Open Systems Interconnection (OSI) Models vor. Dieses Model besteht aus 7 Schichten:

Schicht 1 — Bitübertragungsschicht *(Physical Layer)* beschreibt das Übertragungsmedium, z.B. Ethernetkabel. Hier werden die Rohdaten tatsächlich über das Netz übertragen.

Schicht 2 — Sicherungsschicht *(Data Link Layer)* ist für die physikalischen Adressen verantwortlich (Media Access Control Address - MAC-Adresse). Diese Schicht ist für die Übertragung der Informationen zwischen zwei Knoten (Rechnern) verantwortlich. Einfache Fehler können korrigiert werden.

Schicht 3 — Vermittlungsschicht *(Network Layer)* ist für die logischen Adressen verantwortlich (IP-Adresse). Ist wichtig, wenn keine direkte Verbindung zwischen Sender und Empfänger vorhanden ist. Jedes Paket bekommt IP-Adresse des Ziel-Rechners und kann deswegen unabhängig von anderen Paketen zum Ziel kommen. Diese Schicht ist für die Übertragung der Informationen zwischen zwei Netzwerken verantwortlich. Keine Fehlerkontrolle.

Schicht 4 — Transportschicht *(Transport Layer)* fragmentiert (beim Senden) und defragmentiert (beim Empfang) die Datennachrichten. Jedes Paket bekommt seine Reihenfolgenummer. Die Fehler werden korrigiert (acknowledge).

Schicht 5 — Kommunikationssteuerungsschicht *(Session Layer)* ist eine Schittstelle zwischen physikalischen und logischen Schichten (Remote Procedure Calls).

Schicht 6 — Darstellungschicht *(Presentation Layer)* übersetzt die Daten in eine Form, die von allen Rechnern verstanden wird.

Schicht 7 — Anwendungsschicht *(Application Layer)* stellt die kommunizierende Anwendungen dar.

Otto’s Brief an Lili:

Otto in Berlin	**ISO OSI**	**Lili in Paris**
Otto’s Brief in Deutsch.	*Application Layer*	Otto’s Brief in Französisch.
Dolmetscher: Deutsch => Englisch.	*Presentation Layer*	Dolmetscher: Englisch => Französisch.
Brief geht zur Post.	*Session Layer*	Den ganzen Brief dem Empfänger zustellen.
Brief wird zerlegt, Teile werden numeriert.	*Transport Layer*	Brief wird zusammengestellt. Überprüfen, ob alle Teile angekommen sind.
Jedes Teil (Päkchen) bekommt Adresse des Empfängers.	*Network Layer* (logische Adressen)	Päkchen netzwerkübergreifend übertragen.
Päkchen werden vom Punkt zum Punkt im lokalen Netz zugestellt.	*Data Link Layer* (physische Adressen)	Päkchen werden vom Punkt zum Punkt im lokalen Netz zugestellt.
Bit-Übertragung.	*Physical Layer*	Bit-Übertragung.

UNIX/Linux unterstützen Protokolle: TCP/IP, NetBIOS, NetWare.

3.2. TCP / IP

3.2.1. Aufbau von TCP/IP

Das Transmission Control Protocol / Internet Protocol (TCP/IP) beschreibt einen ganzen Satz weit verbreiteter Transport- und Netzprotokolle. Standard für die Vernetzung von verschiedensten Systemen, aber ursprünglich - in UNIX-Welt. Advanced Research Project Agency (Department of Defence) hat ARPANET entwickelt.

Schichte:

Anwendungen (**ftp**, **telnet**, **rlogin**, Benutzer)
TCP (Transport Layer)
IP (Network Layer)
Netzwerkkarte (Ethernet, ISDN)

- IP. Es wird zum Datenaustausch benutzt. Vor dem Senden wird Information in mehrere kleine Teile zerlegt (Pakete). Jedes Paket hat die Ziel-Adresse und wird unabhängig von anderen zum Ziel geschickt. Adressierung, Routing und Weiterleitung der Datenpakete im Netz. (Allgemeine Post).

- TCP. Betriebssystem verwaltet verschiedene Prozesse, die dem rufenden Rechner zur Verfügung gestellt werden können. Der rufende Rechner kann diese Prozesse durch bestimmte numerierte Ports beanspruchen. Hier muß man unter dem Port einen Socket (Software) verstehen. Dieser Socket definiert, welcher logische Ein- und Ausgang eines Prozesses mit den aktuell anstehenden Daten versorgt werden soll, also Socket ist eine Adresse, die Port-Identifikation enthält. Post innerhalb des Unternehmens.

Internet-Adresse 32 Bit, Port-Adresse 16 Bit.

3.2.2. Adreßstruktur

Die Adresse wird in 4 Zahlen unterteilt je 8 Bit (1 Byte: 0-255), z.B: 123.234.123.234 - eindeutig im ganzen Netz.

IP-Adresse = Netzwerk-Adresse Rechner-Adresse.

Klassen von Adressen (Netzen): A, B, C. Netzwerk-Adresse:

	Byte 1	*Byte 2*	*Byte 3*	*Byte 4*
Klasse A	0 – 127	0	0	0
Klasse B	128 – 191	0 – 255	0	0
Klasse C	192 – 255	0 – 255	0 - 255	0

> Die Adressen 0.0.0.0 und 255.255.255.255 darf man nicht benutzen, die Adresse für loopback: 127.0.0.1. Folgende Adressen sind für firmen-interne Benutzung bestimmt:
> Klasse A: 10.0.0.0 - 10.255.255.255
> Klasse B: 172.16.0.0 - 172.31.255.255
> Klasse C: 192.168.0.0 - 192.168.255.255

3.2.3. Routing

Aber: Wichtigste Rolle spielt die Maske: z.B. 255.255.255.0 für Klasse C.

Beschreiben - wie geht das Routing vor. Datei **`/etc/hosts`**, Gateway.

Die Maske wird mit der IP-Adresse durch logische AND verknüpft, um die Adresse des Netzwerkes zu ermitteln.

3.3. Netzwerkdienste

3.3.1. NFS

Network File System (NFS) ermöglicht es, die Daten von anderen Rechnern im Netz gemeinsam zu benutzen. Es schafft heterogene Arbeitsumgebung, d.h. man kann mit dem beliebigen Betriebssystem kommunizieren (NFS ist für viele Betriebssysteme verfügbar). Anwendungen müssen nicht wissen, wo die entfernten Dateien gespeichert sind. NFS erfordert TCP/IP als Transportmittel. Das System ist als Dämonen realisiert, die beim Hochfahren des Rechners gestartet werden müssen.

Der Rechner, der seine Daten den anderen Rechnern zur Verfügung stellen will, muß in der Datei **`/etc/exports`** die Verzeichnisse, die Namen von Rechnern (für die der Zugriff erlaubt ist) und die Rechte eingetragen haben, z.B. auf dem Rechner **`dresden`**:

```
/usr/local berlin (ro)
```

Auf dem Rechner, der die Daten benutzen will, muß entsprechendes Verzeichnis montiert werden, z.B. auf dem Rechner **`berlin`**:

```
# mount -t nfs dresden:/usr/local /mnt/dresden
```

3.3.2. NIS

Network Information Service (NIS) ermöglicht es, jedem Benutzer sich an beliebigen Rechner anzumelden, und der Benutzer bekommt immer seine Arbeitsumgebung. NIS verwaltet eine administrative Datenbank, die Daten über die Kontos der Benutzer enthält. Der andere Namen für NIS ist Yellow Pages.

3.3.3. rlogin, telnet

Funktionalität von diesen Programmen ist fast gleich, **`rlogin`** wird in der Zukunft durch **`telnet`** ersetzt. Durch diese Programme kann man sich an den entfernten Rechner anmelden und die Ressource (Prozessorzeit, Dateisysteme) von diesem Rechner benutzen. Der Benutzer muß dem entfernten Rechner selbstverständlich bekannt sein.

3.3.4. ftp

Mit diesem Programm kann man die Dateien zwischen zwei Rechnern übertragen. In der Datei **`/etc/ftpusers`** sind die Gruppe eingetragen, die mit **`ftp`** keinen Zugriff auf den anderen Rechner haben, weil sie genug Rechte haben, mit dem entfernten Rechner auf andere Weise zu arbeiten, z.B. **root**, - muß man kommentieren.

3.3.5. WWW

Das ist ein modernes Mittel, das die Möglichkeit anbietet, fast alle Dienste wie **`ftp`**, **`mail`**, **`telnet`** u.s.w. in einer Shell zu benutzen. Man braucht dafür einen Browser wie z.B. Mosaic. Die Daten werden dabei in dem HTML-Format dargestellt.

3.3.6. e-mail

Elektronische Post (e-mail) stellt eine bequeme Möglichkeit zur Verfügung, die Nachrichten (Texte) zwischen Rechnern zu verschicken. Es gibt viele Programme, die es gewährleisten: **`sendmail`** (schwer zu installieren), **`smail`**, **`pine`**, **`elm`**.

3.3.7. Drucken im Netz

Damit die entfernte Rechner den Zugriff auf lokalen Drucker haben, müssen sie auf dem lokalen Rechner in der Datei **`/etc/hosts.lpd`** eingetragen werden.

4. Administrieren

4.1. Aufgaben des Systemadministrators

- Anwender hinzufügen und entfernen;
- Hardware hinzufügen und entfernen;
- Installation neuer Software;
- Sicherungskopien anlegen;
- Systemüberwachung;
- Systemsicherheit;
- Fehlersuche;
- Verwaltung lokaler Dokumentation;
- Anwenderunterstützung.

4.2. Datensicherung

`tar`, **`cpio`**

4.3. Sicherheit im Netz

4.4. Softwaremanagement

4.5. Batch-Betrieb

`crond`

4.6. Recompilieren des Kernels

5. Installation

5.1. Aufbau der Festplatte

Die ersten 512 Bytes auf der Festplatte heißen Master Boot Record (MBR). Der rest kann höchstens in 4 Teilen (Partitionen) unterteilt werden. Die ersten 512 Bytes auf der Partition heißen Boot-Sektor. Rest der Partition ist für das Dateisystem vorgesehen.

Der MBR hat folgende Struktur:

Byte	Inhalt
0-445	Initialisierungsprogramm IPL (446 Byte)
446-511	Partitionstabelle (64 Byte + 2 Byte):
446-461	1. Partition (16 Byte):
446	Aktivierung (Ja/Nein)
447, 448, 449	Beginn: Seite, Sektor, Zylinder
450	Typ der Partition
451, 452, 453	Ende: Seite, Sektor, Zylinder
454-457	Relative Sektoren (4 Byte)
458-461	Anzahl der Sektoren (4 Byte)
462-477	2. Partition (16 Byte)
478-493	3. Partition (16 Byte)
494-509	4. Partition (16 Byte)
510-511	Endmarkierung (2 Byte):
510	55h
511	AA

Die Partitionstabelle beinhaltet Information über maximal 4 Partitionen. Es gibt folgende Partitionstypen (MS-DOS kennt nur wenige davon):

01	MS-DOS 12 bit FAT
02	XENIX root
03	XENIX usr
04	MS-DOS 16 bit FAT < 32 MB
05	Extended
06	MS-DOS 16 bit FAT >= 32 MB
07	OS/2 HPFS
a	OS/2 Boot-Manager
64	Novell Netware
83	Linux
82	Linux Swap

Diese Liste hat keinen Anspruch auf Vollständigkeit.

Programmcode von IPL und Boot-Sektor sind vom Betriebssystem abhängig, aber die Struktur der Partitionstabelle stellt einen Standard dar. Nicht verwechseln die Partitionstabelle mit File Allocation Table (FAT).

5.2. Boot-Konzepte

5.2.1. Boot Konzept von MS-DOS

Boot-Vorgang besteht aus 3 Schritten.

- Beim Hochfahren des Rechners wird automatisch das Initialisierungsprogramm (IPL) in den Hauptspeicher geladen. Das MD-DOS-IPL befindet sich im MBR - das sind immer die ersten 446 Byte auf der Festplatte.
- Das IPL findet auf der aktiven Partition anhand der Partitionstabelle den Boot-Sektor und startet ihn.
- Das Programm aus dem Boot-Sektor startet das MS-DOS-Betriebssystem von der aktiven Partition.

Dieser Vorgang ist fest vorgeschrieben und kann nicht geändert werden.

5.2.2. Boot Konzept von Linux

Der Boot-Vorgang vom Linux ist viel flexibler. Linux hat eigenen Loader LILO. Zuerst muß die erste Stufe des Loaders gestartet werden. Sie kann sich auf der Diskette, im MBR (erste 446 Byte), im Boot-Sektor oder im Datenbereich der Partition befinden. Dieser Code enthält die Adresse der Linux-Partition und der zweiten Stufe des Loaders auf dieser Partition. Die zweite Stufe kann jetzt Linux oder beliebiges anderes Betriebssystem starten, d.h. sie ist tatsächlich ein Boot-Manager.

Liegt die erste Stufe:

- im MBR, so startet sie automatisch;
- im Boot-Sektor, so muß sie vom IPL (aus MBR) oder vom fremden Boot-Manager (z.B. OS/2) gestartet werden;
- im Datenbereich auf der Partition, so muß sie vom extra Programm gestartet werden (z.B. **`loadlin.exe`** aus DOS-Umgebung).

Die ganzen Bootprogramme befinden sich unter dem Verzeichnis **`/boot`** auf der Linux-Partition. In der ersten Stufe vom LILO sind die absoluten Adressen von diesen Dateien und vom Kernel gespeichert, deswegen muß man immer den Befehl **`lilo`** eingeben, wenn irgendwelche Änderungen in diesem Verzeichnis durchgeführt wurden.

5.3. LILO

5.4. Installationvorgang

5.5. Paketenauswahl

6. X Window System

Das ist netzwerkbasierte grafische Benutzeroberfläche (GUI). Von DEC und MIT entwickelt (erste Version - 1987). Markname: **`Xfree86`**.

Es ermöglicht, mehrere Programme gleichzeitig in mehreren Fenstern ablaufen zu lassen. MS-Windows arbeitet stark kernelorientiert. X Window System ist ein normales (aber recht kompliziertes) Programm auf Benutzerebene.

6.1. Client-Server-Architektur

Kein Bestandteil des Betriebssystems. Echte Client-Server-Architektur von Anfang an. Diese Architektur ist durch Netz verteilt. X-Server stellt die Anforderungen von Clients grafisch dar. Client und Server tauschen die Daten durch X-Protokoll. Die Operationen *Rechnen* und *Anzeigen* sind entkoppelt. Window-Manager ist nur ein Client unter allen, aber wichtiger, es gibt mehrere.

6.2. Start vom X

Für den Start muß der Pfad **`/usr/X11R6/bin`** in die Umgebungsvariable **`PATH`** aufgenommen werden, und für den Zugriff auf die Hilfe muß der Pfad **`/usr/X11R6/bin`** in die Datei **`/usr/lib/manpath.config`** eingetragen werden.

`startx` => **`xinit`** (startet X-Sever, i.e. **`/usr/X11R6/bin/X`** - das ist ein Link auf z.B. XF86_SVGA) => **`xinit`** arbeitet weiter die Konfigurationsdatei **`~/.xinitrc`** oder **`/usr/X11R6/lib/X11/xinit/xinitrc`** ab. Der letzte Client in der Konfigurationsdatei (üblicherweise Window-Manager) darf nicht im Hintergrund (mit **`&`**) gestartet werden!

6.3. Netz-Transparenz

Es gibt zwei Möglichkeiten, eine Anwendung als Client für den X Server auf dem anderen Rechner zu starten: rechner- und benutzerbezogene.

6.3.1. Rechner-Abhängigkeit

Die erste Möglichkeit wird durch die Datei **`/etc/X?.hosts`** realisiert (das Fragezeichen **`?`** steht hier für die Nummer der Workstation, im Falle von PC das ist **`0`**). In dieser Datei werden alle externe Rechner eingetragen, die sich bei dem Server anmelden dürfen. Das ist aber schwache Stelle aus der Sicht der Sicherheit - Datei ist unverschlüsselt und kann mit jedem Text-Editor geändert werden.

```
Inhalt der Datei /etc/X0.hosts auf Z_Rechner:
Q_Rechner

user1@Q_Rechner:/home/user1 > xeyes -display Z_Rechner:0
```

Weitere Programme, die auf dieser Weise gestartet werden können: **xterm**, **xcalc**. Terminieren kann man auf beiden Seiten (auf Client-Seite: *Strg-C* oder **kill**, wenn mit **&** gestartet wurde).

6.3.2. Benutzer-Abhängigkeit

Die zweite Möglichkeit sieht das Paßwort für den Benutzer vor und ist durch Cooki-Mechanismus realisiert.

6.4. X Konfiguration

Mit den Befehlen **xf86config** und **XF86Setup** kann man die Konfigurationsdatei **/etc/XF86Config** erstellen. In dieser Datei sind die Einstellungen für X beschrieben.

6.5. Standard-Clients

6.5.1. Standard-Anwendungen

Das Programm **xterm** startet eine Emulation vom Terminal unter X. Es hat u.a. folgende Optionen:

-display *host:ws.d*	das Programm wird auf dem Rechner *host*, auf der Workstation *ws*, auf dem Bildschirm *d* gestartet;
-geometry *WxH+X+Y*	Fensterdefinition: *W* - Breite, *H* - Höhe, *X* und *Y* - Bildschirmposition;
-fg *farbe*	Vordergrundfarbe des Fensters;
-bg *farbe*	Hintergrundfarbe des Fensters;
-title *name*	Titelleiste des Fensters;

Das Programm **xman** bietet Hilfe unter X.

Das Programm **xclock** startet die Uhr.

Das Programm **xcalc** startet Taschenrechner.

Das Programm **xeyes** macht einfach Spaß.

6.5.2. File Managers

Die File Managers gewährleisten einen visuellen Zugriff auf die Daten des Dateisystems, wobei die Dateien schnell und bequem ausgewählt werden können, oder man schnell in das Verzeichnis wechseln kann. Die File Managers, die einfach ausprobiert werden müssen: **`xfm`**, **`FileMan`**, **`tkdesk`**, **`kdm.`**

6.5.3. X Window Managers

Alle Window Managers können ausprobiert werden. Ausführlich werden **`fvwm2`** und **`fvwm95`** betrachtet. Konfigurationsdateien sind **`~/.fvwm2rc`** und **`~/.fvwm95rc`**. Wenn sie fehlen, muß man als Muster die Dateien **`/usr/X11R6/lib/X11/fvwm2/.fvwm2rc`** und **`/usr/X11R6/lib/X11/fvwm95/.fvwm95rc`** kopieren.

6.6. Anwendungen unter X

6.6.1. Editoren

Unter X stehen viele Texteditoren zur Verfügung: **`emacs`**, **`xemacs`**, **`asedit`** u.v.m. Sie sind in der Regel menügesteuert und ganz leicht handzuhaben.

6.6.2. Postscript-Viewer

Postscript ist eine Standardsprache für die Darstellung der zu druckenden Dokumenten in der UNIX-Welt. Solche Dokumente kann man sich mit den Programmen **`ghostview`**, **`xdvi`** ansehen und ausdrucken. In der Linux-Distribution gibt es excellente Bücher mit der Beschreibung dieses Betribssystems.

6.6.3. Andere Anwendungen

Man kann nicht alle nützliche Anwendungen unter X auflisten: Tabellenkalkulation, Textbearbeitung, Datenbanken, Bild- und Filmverarbeitung, CD-ROM-Brennsoftware, Audioverarbeitung, Spiele, kommerzielle Software jeder Art u.v.m. Das alles muß man einzeln installieren und ausprobieren.

7. Appendix

7.1. Scripts

```
#=========================   B O F   ==========================
#   di
#   Display directory if too many files found
#
ls -a -l | less
#
#=========================   E O F   ==========================

#=========================   B O F   ==========================
#   ff
#   Find file beginning from /
#   Parameter:
#       file name to seach
#   Temporary file will be created: /tmp/DisplayTMP
#   It is comfort to use if too many files found
#
SF=/tmp/DisplayTMP
find / -name $1 > $SF
if test -s $SF
then
   less $SF
else
   echo "File $SF not found"
fi
rm $SF
#
#=========================   E O F   ==========================

#=========================   B O F   ==========================
#   qu
#   Script für Herunterfahren des Rechners
#
clear
echo "-------------- Es sind noch angemeldet -------------"
finger
echo "----------------------------------------------------"
echo "Soll der Rechner heruntegefahren werden?"
echo "ENTER   Ja, Herunterfahren"
echo "Strg-C  Nein, Weiterarbeiten"
read A
halt
#
#=========================   E O F   ==========================

#=========================   B O F   ==========================
#   qu
#   Noch ein Script für Herunterfahren des Rechners
```

```
clear
echo    "-------------- Es sind noch angemeldet -------------"
finger
echo    "----------------------------------------------------"
echo    "Soll der Rechner heruntegefahren werden?"
echo    "                  J     - Ja, Herunterfahren"
echo    "                  Sonst - Nein, Weiterarbeiten"
echo -e "Ihre Auswahl ===> \c"
read A
if [ "$A" != "" ]
then
   if [ $A = J -o $A = j ]
   then
      echo OK, der Rechner wird heruntergefahren...
      halt
   fi
fi
#
#=========================   E O F   ==========================

#=========================   B O F   ==========================
#   ma
#   Mount & Unmount diskette A
#      Without parameter - Mount
#      Any parameter - Unmount
#
if [ $# -gt 0 ]
then
   umount /mnt/Disk_A
   echo 'Floppy: successful unmounted'
else
   mount /mnt/Disk_A
   echo 'Floppy: successful mounted'
fi
#
#=========================   E O F   ==========================

#=========================   B O F   ==========================
#   mq
#   Mount & Unmount CD-ROM
#     Without parameter - Mount
#     Any parameter - Unmount
#
if [ $# -gt 0 ]
then
   umount /mnt/CD_ROM
   echo 'CD-ROM: successful unmounted'
else
   mount /mnt/CD_ROM
   echo 'CD-ROM: successful mounted'
fi
#
#=========================   E O F   ==========================
```

```
#=========================   B O F   ==========================
#   mu_nfs
#   Mount an directory from NFS.
#   Parameters:
#      NFS-servername
#      remote directory
#   A directory will make automatically on local machine
#   under /mnt
#   Example:
#      mu_nfs hz tmp
#
mkdir /mnt/nfs_$1_$2
mount -t nfs $1:/$2 /mnt/nfs_$1_$2
#
#=========================   E O F   ==========================

#=========================   B O F   ==========================
#  um_nfs
#  Unmount an directory from NFS.
#  Parameters:
#      NFS-servername
#      remote directory
#  A directory will delete automatically on local machine
#  under /mnt
#  Example:
#      um_nfs hz tmp
#
umount /mnt/nfs_$1_$2
rmdir  /mnt/nfs_$1_$2
#
#=========================   E O F   ==========================

#=========================   B O F   ==========================
#   ki
#   kill a process
#
kill -9 $1
#
#=========================   E O F   ==========================

#=========================   B O F   ==========================
#   iam
#   Shell-Script info
#
echo
echo "******************   Info   ********************"
#
#Datum wie: Mon Nov 16 14:05:54 MET 1992
#
date > hilf.dat
read wtag monat mtag zeit zone jahr < hilf.dat
echo "Heute ist $wtag, der $mtag. $monat $jahr."
#
# Benutzername und Terminal wie: user1 tty03 Feb 12 08:02
#
```

```
who am i > hilf.dat
read name term rest < hilf.dat
echo "Ich bin als $name am Terminal $term angemeldet."
#
#   Aktuelles Verzeichnis wie: /usr/home/user1
#
pwd > hilf.dat
read verz < hilf.dat
echo "Mein aktuelles Verzeichnis ist $verz."
#
#   Benutzer
#
who | wc -l > hilf.dat
read anz1 < hilf.dat
cat /etc/passwd | wc -l > hilf.dat
read anz2 < hilf.dat
echo "Derzeit sind $anz1 von $anz2 Benutzern aktiv."
echo "***************   Das war es   *****************"
echo
#
rm hilf.dat
#
#=========================   E O F   ==========================

#=========================   B O F   ==========================
#   uhr
#   Datum und Uhrzeit in einer Zeile anzeigen
#
echo 'Heute ist '      > help    #  create help file
date +%d.%m.%Y        >> help    #  add the date
echo ', Uhrzeit '     >> help    #  add the text
date +%T              >> help    #  add the time
cat help | tr -d '\n'  > help    #  remove CR/LF
echo '' > help2                  #  create CR/LF
cat help2 help help2 help2       #  show the line
rm help help2                    #  remove help files
#
#=========================   E O F   ==========================

#=========================   B O F   ==========================
#   pi
#   easy script to ping
#   must be started with parameter - target host
#clear
if [ $# -gt 0 ]
then
   echo
   echo -n 'Starting ping to' $1 '- wait please... '
   ping -c 3 -q $1 > /tmp/nul
   if [ $? -gt 0 ]
   then
      echo Error
   else
      echo OK
   fi
```

```
else
   echo No host parameter
fi
echo 'Thank you for using'
echo
#
#=========================   E O F   ==========================

#=========================   B O F   ==========================
#   Inhalt von allen Dateien im Verzeichnis /root/bin
cd /root/bin
for i in *
do
   echo
   echo ================== Das ist die datei $i ==============
   cat $i
   echo ================== Press any key ============
   read
done
#
#=========================   E O F   ==========================

#=========================   B O F   ==========================
#   wer_ist_da
#   Rausfinden, welche hosts im Netz sind (aus /etc/hosts)
clear
#
egrep "^[0-9]?[0-9]?[0-9]\.[0-9]?[0-9]?[0-9]\.[0-9]?[0-9]?[0-9]\.
[0-9]?[0-9]?[0-9]" /etc/hosts > temp1
#
#cat temp1
#echo "============================="
#
grep -v 'localhost' temp1 > temp2
#
#cat temp2
#echo "============================="
#
cut -f 2 temp2 > temp3
#
#cat temp3
#echo "============================="
#
cat temp3 | tr -s '\n' '\t' > temp4
#
#cat temp4
#echo "============================="
#
read A < temp4
pi_p $A  #  siehe nächsten Script
#
#=========================   E O F   ==========================

#=========================   B O F   ==========================
#   pi_p
```

```
echo
while [ "$1" != "" ]
do
   echo $1
   ping -c 1 -q $1 > nul
   if [ $? -gt 0 ]   #   errorlevel
   then
      echo 'Error' $1
   else
      echo 'OK' $1
   fi
   shift
done
#=========================   E O F   ==========================

#=========================   B O F   ==========================
#   test of dialog command
#!/bin/bash
#-----------------------------------------------------------
dialog --title "Question"                  \
       --backtitle "To be or not to be"    \
       --yesno "Would you like to answer?" \
       6 40
if [ $? -eq 0 ]  #  exit status
then
   echo "Yes"
else
   echo "No"
fi
read
#-----------------------------------------------------------
dialog --title "Input string"       \
       --backtitle "This is a try"  \
       --inputbox "Input please..." \
       8 40                         \
       2> q
read A < q
echo $A
read
#-----------------------------------------------------------
A=/etc/fstab
dialog --title "  Ihre Datei $A sieht so aus:  " \
       --backtitle "This is a try"               \
       --textbox $A                              \
       12 60
#-----------------------------------------------------------
dialog --title "Menu" \
       --backtitle "This is a MENU" \
      --menu "Bitte schön... \nSie dürfen irgendwas auswählen" \
        16 60 6                                                \
       "China-Pfanne" "Ein Produkt aus China"                  \
       "Pizza"        "Essen für Computer-Freaks"              \
       "Big-Mac"      "Schnell-Essen"                          \
       "Bier"         "Deutsches Nationalgericht"              \
       2> q
```

```
read A < q
echo $A
read
#-------------------------------------------------------
dialog --title "Menu"                                                \
       --backtitle "Checklist"                                       \
       --checklist "Bitte schön... \nSie dürfen alles auswählen" \
       16 60 6                                                       \
       "China-Pfanne" "Ein Produkt aus China"     off                \
       "Pizza"        "Essen für Computer-Freaks" off                \
       "Big-Mac"      "Schnell-Essen"             off                \
       "Bier"         "Deutsches Nationalgericht" on                 \
       2> q
read A < q
echo $A
read
#-------------------------------------------------------
dialog --title "Menu"                                        \
       --backtitle "Radiolist"                               \
       --radiolist "Sie dürfen hier nur eines auswählen" \
       16 60 6                                               \
       "China-Pfanne" "Ein Produkt aus China"     off    \
       "Pizza"        "Essen für Computer-Freaks" off    \
       "Big-Mac"      "Schnell-Essen"             off    \
       "Bier"         "Deutsches Nationalgericht" on     \
       2> q
read A < q
echo $A
read
#=========================   E O F   ==========================

#=========================   B O F   ==========================
clear
echo '                    Menu'
echo '                  -------------'
echo '                    1 - Fisch'
echo '                    2 - Bier'
echo '                    3 - Pizza'
echo '                  -------------'
echo -n 'Ihre Auswahl ---> '
read A
#A=1
case $A in
    1)  echo Fisch ist noch nicht fertig
        ;;
    2)  echo Bier ist frisch
        ;;
    3)  echo Pizza kommt aus Italien
        ;;
    *)  echo So was gibt es nicht
        ;;
esac
#=========================   E O F   ==========================

#=========================   B O F   ==========================
```

```
clear
select V in berlin dresden leipzig bonn quit
do
   case $V in
      berlin)    echo berlin berlin berlin berlin
                 ;;
      dresden)   echo dresden dresden dresden
                 ;;
      leipzig)   echo leipzig leipzig
                 ;;
      bonn)      echo bonn bonn
                 ;;
      quit)      echo und tschuess....
                 break
                 ;;
      *)         echo Falsche Auswahl
                 ;;
   esac
done
#=========================   E O F   ==========================
```

7.2. Dateien

```
#=========================   B O F   ==========================
#   /etc/fstab
#
/dev/hda3    /               ext2       defaults         1   1
/dev/fd0     /mnt/Disk_A     msdos      rw,noauto,user   1   1
/dev/hdc     /cdrom          iso9660    ro,noauto,user   0   0
none         /proc           proc       defaults         0   0
#
#=========================   E O F   ==========================

#=========================   B O F   ==========================
#!/bin/sh
# $XConsortium: xinitrc.cpp,v 1.4 91/08/22 11:41:34 rws Exp $
#
userresources=$HOME/.Xresources
usermodmap=$HOME/.Xmodmap
sysresources=/usr/X11R6/lib/X11/Xresources
sysmodmap=/usr/X11R6/lib/X11/Xmodmap
#
# merge in defaults and keymaps
#
if [ -f $sysresources ]; then
    xrdb -merge $sysresources
fi
#
if [ -f $sysmodmap ]; then
    xmodmap $sysmodmap
fi
#
if [ -f $userresources ]; then
    xrdb -merge $userresources
fi
```

```
#
if [ -f $usermodmap ]; then
    xmodmap $usermodmap
fi
#
# start some nice programs
#
# xclock -geometry 50x50-1+1 &
# xterm -geometry 80x50+494+51 &
# xterm -geometry 80x20+494-0 &
#
#if [ -x /usr/X11R6/bin/fvwm2 ]; then
#    exec fvwm2
#fi
#
#if [ -x /usr/X11R6/bin/fvwm ]; then
#    exec fvwm
#fi
#
#exec twm
#
exec $WINDOWMANAGER
# von mir hinzugefuegt
#
#=========================   E O F   ==========================
```

7.3. Aufgaben

Eine Datei names **`telefon.dat`** ist zu erstellen:

```
Meier:39
Meier Paul:47
Schulze:39
Kurth Peter:13
Walther:19
```

Sortieren diese Datei nach Namen und Nummern vorwärts und zurück:

```
sort telefon.dat
sort -r telefon.dat
sort -t: +1 telefon.dat
sort -t: -r +1 telefon.dat
```

Suchen mit **`grep`** nach Namen und Telefonnummern, die ganz oder teilweise spezifiziert sind:

```
grep 'M' telefon.dat
grep '[KS]' telefon.dat
grep '39' telefon.dat
```

Suchen mit **`grep`** nach **`color`** und **`colour`**

```
grep 'colo[u]*r' dateiname
```

Es wird gemäß dem zweiten Feld (und nur diesem) rückwerts sortiert (Feldtrenner sind Leerzeichen und Tabulator):

```
sort -r +1 -2 dateiname
```

Eigene Benutzerkennung ermitteln:

```
grep 'login-name' /etc/passwd
```

Befehl **`date`** - *`hh mm ss`* statt *`hh:mm:ss`* ausgeben:

```
date | tr ':' ' '
```

Es geht nicht (multiple files):

```
mv *.cc *.c
```