Unix Grundlagen mit LINUX

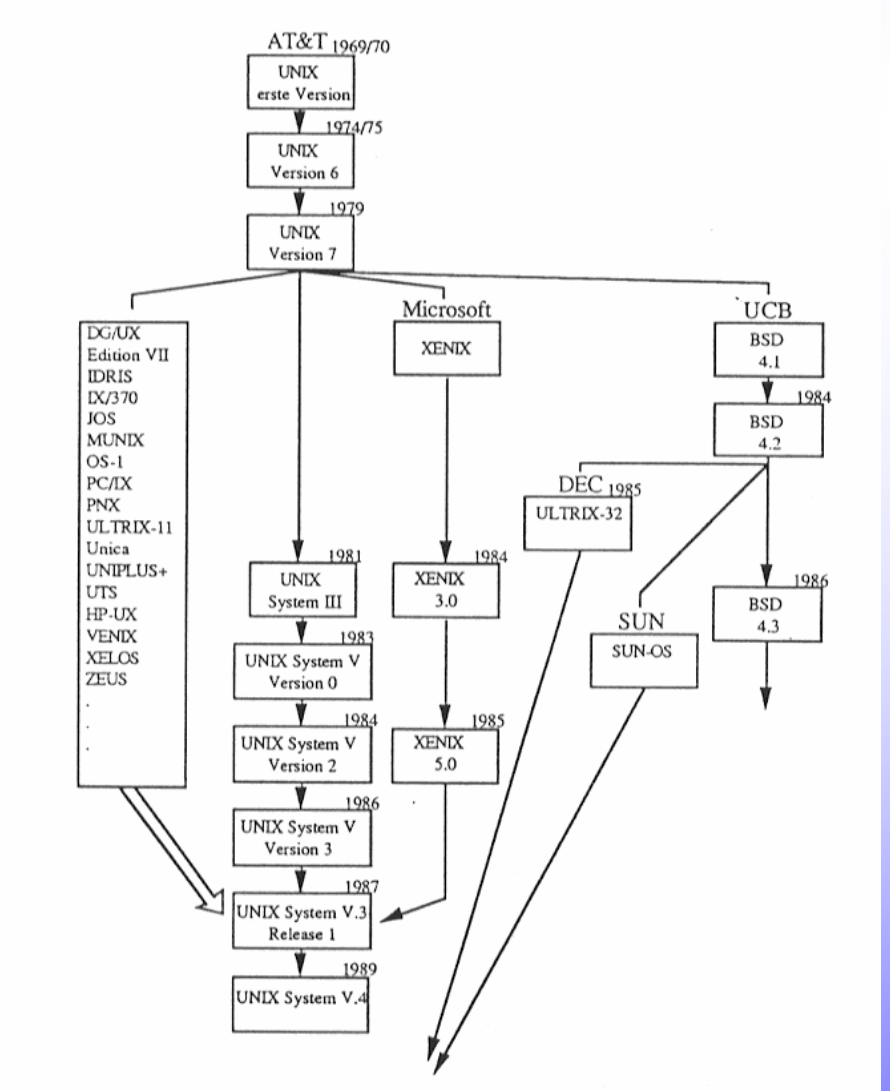
Dipl.-Ing. Christian Prager

Inhalt

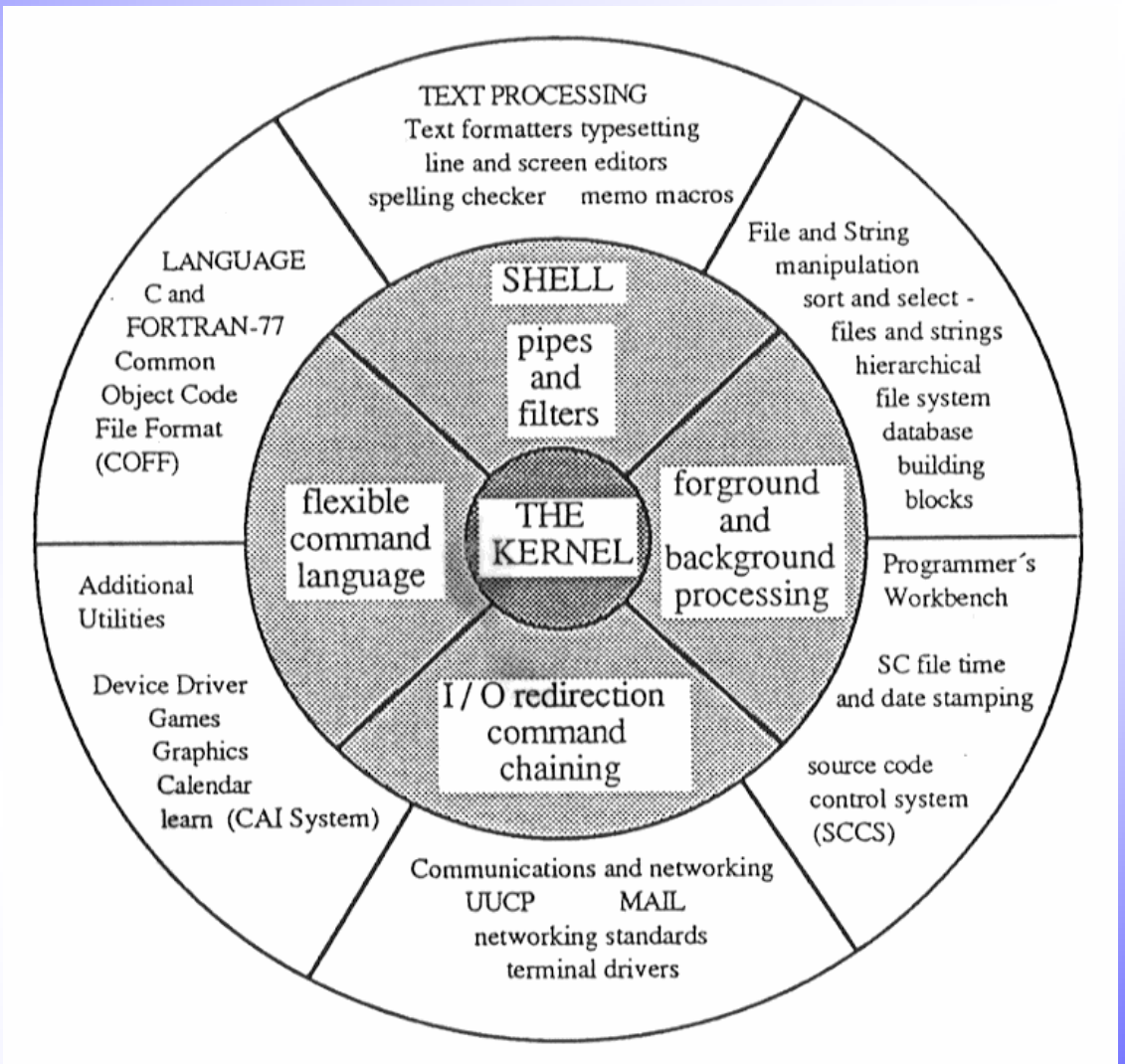
VORWORT

- Die tcsh – Shell wurde als Basis gewählt, da sie viel Komfort bietet und mit der ksh – und bash – Shell fast gleich ist.
- Das % - Zeichen ist das Prompting in den Beispielen
- [ ] geben Wahlmöglichkeiten an
- Übungsaufgaben sind nicht in den Unterlagen und dem Vortragenden freigestellt, Beispiele sehr wohl.
- Bei wichtigen Unterschieden wird gesondert auf die Eigenheiten der ksh- bzw. der bash – Shell eingegangen
- Die Oberfläche KDE sollte selbsterklärend sein.
- Der Systeminstallations-Ablauf wird kurz demonstriert

EINLEITUNG


AT&T 1969/70
UNIX
erste Version
1974/75
UNIX
Version 6
1979
UNIX
Version 7
DG/UX
Edition VII
IDRIS
IX/370
JOS
MUNIX
OS-1
PC/IX
PNX
ULTRIX-11
Unica
UNIPLUS+
UTS
HP-UX
VENIX
XELOS
ZEUS
Microsoft
XENIX
UCB
BSD
4.1
1984
BSD
4.2
DEC 1985
ULTRIX-32
1981
UNIX
System III
1983
UNIX System V
Version 0
1984
UNIX System V
Version 2
1986
UNIX System V
Version 3
1987
UNIX System V.3
Release 1
1989
UNIX System V.4
1984
XENIX
3.0
1985
XENIX
5.0
SUN
SUN-OS
1986
BSD
4.3

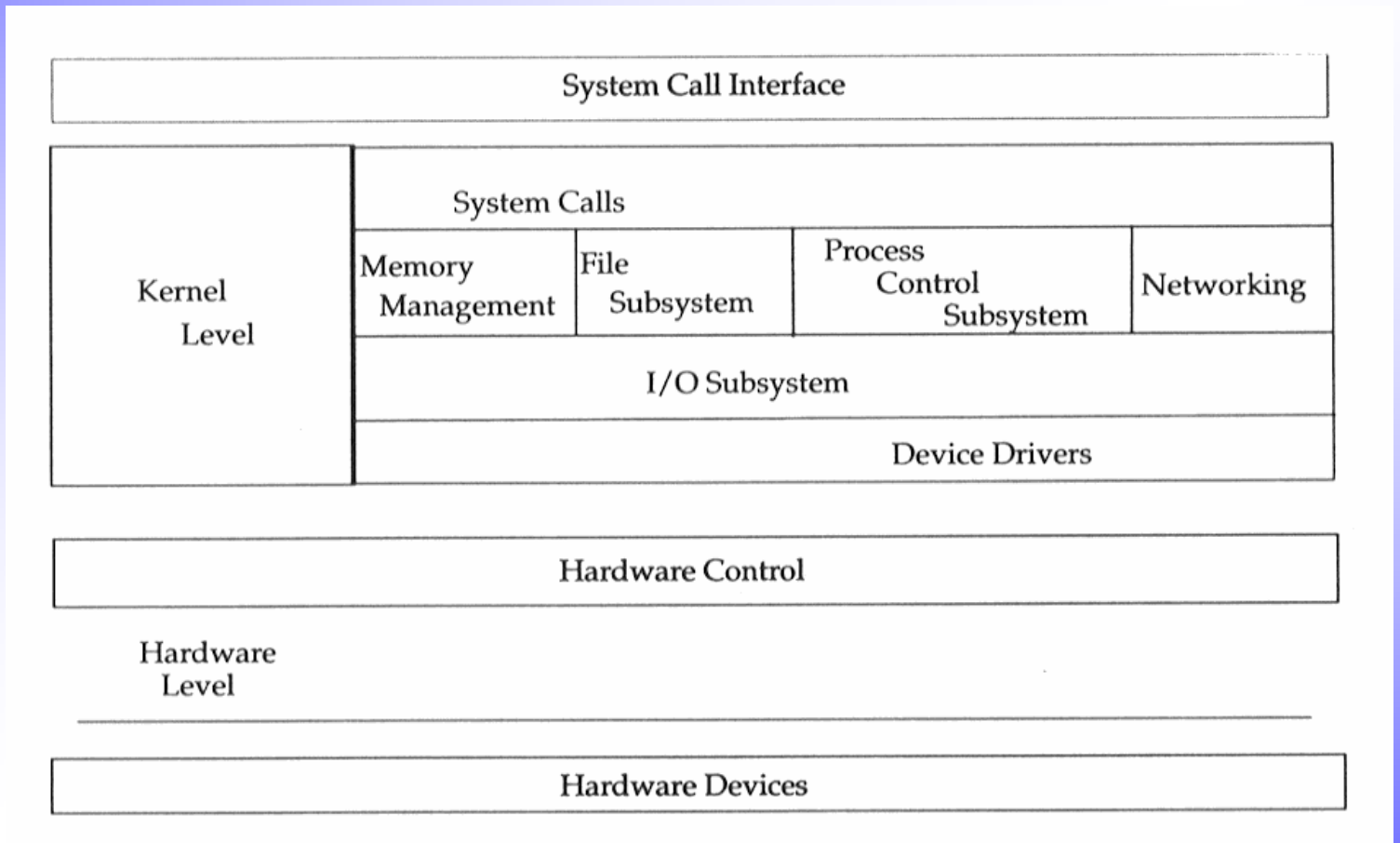
UNIX AUFBAU


TEXT PROCESSING
Text formatters typesetting
line and screen editors
spelling checker memo macros
File and String
manipulation
sort and select -
files and strings
hierarchical
file system
database
building
blocks
Programmer´s
Workbench
SC file time
and date stamping
source code
control system
(SCCS)
Communications and networking
UUCP MAIL
networking standards
terminal drivers
Device Driver
Games
Graphics
Calendar
learn (CAI System)
Additional
Utilities
LANGUAGE
C and
FORTRAN-77
Common
Object Code
File Format
(COFF)
SHELL
pipes
and
filters
flexible
command
language
THE
KERNEL
forground
and
background
processing
I / O redirection
command
chaining

UNIX Aufbau

- **Kernel**
  - verwaltet Systemspeicher (reell und virtuell)
  - steuert Prozesse
  - verwaltet Dateisystem
  - realisiert Schutzmechanismen (Dateizugriff, Benutzerrechte)
  - steuert Hardware (I/O, Disks, ...
- **Shell**
  - Benutzerinterface
  - interpretiert und expandiert Kommandoeingaben
  - ruft Utilities und Benutzerprogramme auf
  - sorgt für Bildschirm Ein-/Ausgabe, Dateiumleitung
  - enthält Programmiersprache (Shell - Scripts)

Schichtenmodell - Kernel



Kernel wird beim Systemboot geladen

von Filesystem (SUN: /vmunix, SGI: /unix, HP: /hp-ux)

UNIX Prozesse

- von Kernel verwaltet
- durch abwechselnde CPU-Zuteilung Multi Tasking
- Prozeß hat Priorität (nice-value, standard 0, minimal 64)
- Prozeßnummer (PID) wird aufsteigend vergeben
- erster Prozeß "swap" für virtuelle Speicherverwaltung
- zweiter Prozeß "init" sorgt für Systemstart/-stop (Process-id 1)
- wichtige mögliche Prozeßzustände
  - running
  - stopped
  - active (wartet weniger als 20 s)
  - idle (wartet länger als 20 s)
  - deactivated

Erste Schritte in UNIX

LOGIN:

Möglichkeiten Login durchzuführen

- **direkt: serielles Terminal (Konsole)**
- **telnet: `telnet hostname/ip-adresse`**
  erlaubt Terminalemulation (z.B.:VT100, 3270)
- **rlogin: `rlogin hostname/ip-adresse`**
  Von UNIX-System zu UNIX-System;sendet Benutzername mit
- **xdm: X-Windows-Display-Manager**

Login

- **Nach Herstellen der Verbindung zum Host**
  1. **RETURN solange drücken bis am Schirm**
     `login: _` **erscheint.**
  2. **Benutzernamen gefolgt von RETURN eingeben**
     `login: ui103`
  3. **Es erscheint die Passwortabfrage**
     `login: ui103`
     `password: _`
  4. **Passwort eingeben (wird am Schirm nicht ausgegeben) dann RETURN**
  5. **Falls login ferhlerhaft erscheint**
     `login incorrect`
     und das login kann wiederholt werden

- WICHTIG:**UNIX unterscheidet Groß- und Kleinschreibung**
  Benutzernamen IMMER in kleinbuchstaben **eingeben**

Login und Logout

- **Nach erfolgreichem Login erscheinen**
  1. **Systemmeldungen (aktuelle Informationen,Datum, Zeit,...)**
  2. **falls Post vorhanden ist:** `you have new mail`
  3. **der Login-Prompt abhängig von Benutzer und Shell:**
     - **$ Bourne Shell (sh)**
     - **% C-Shell (csh, tcsh)**
     - **# Superuserlogin (root)**
- **Für das Logout gibt es 3 Möglichkeiten**
  1. **den Befehl** logout
  2. **drücken von CTRL-d (CTRL-Taste und d zugleich)**
  3. **Den Befehl** exit
- **es erscheint wieder**
  `login: _`
- **bei telnet/rlogin erscheint auf UNIX-Rechner**
  `connection closed by foreign host`

Befehlseingabe

Die Befehlseingabe

erfolgt zeilenweise
wird mit RETURN abgeschlossen
erlaubt Voraustippen der Befehle

Fehlerkorrektur mit folgenden Tasten

Backspace, ← ,CTRL-h	löscht Zeichenweise
CTRL-u	löscht ganze Zeile
und Cursor - Tasten	

Weitere Tasten

CTRL-s	stoppt Befehls-Ausgabe
CTRL-q	setzt Befehls-Ausgabe fort
CTRL-c	bricht laufenden Befehl ab

Befehlseingabe

- **Befehlseingabe erfolgt in der Form**

befehl optionen dateinamen

befehl	**immer ein Wort (ls, cat, man, lpr, who,...)**
optionen	**immer durch Leerzeichen getrennt (cc -c -O3 ...) meist mit "-" eingeleitet (ls -l)**
dateinamen	**bestimmen Eingabe/Ausgabe des Befehls können auch mit optionen gemischt stehen werden durch Leerzeichen getrennt**

- **Ist die Eingabezeile zu kurz, kann eine Fortsetzungszeile mit "\" am Zeilenende angehängt werden. Werden am Anfang der neuen Zeile Leerzeichen benötigt, müssen diese VOR dem "\" eingefügt werden.**

Einfache Befehle

- `% date` liefert aktuelles Datum und Uhrzeit
- `% uptime` liefert Informationen über System
- `% who` liefert aktive Benutzer, deren Terminals, Loginzeit
- `% finger` liefert Informationen über aktive Benutzer
- `% finger` Benutzername
  liefert Informationen über einen Benutzer

Befehle

- `% finger -l` liefert ausführliche Informationen über Benutzer
- `% tty` liefert den Namen des login terminals
- `% ps` liefert Liste der laufenden Prozesse des Benutzers

```
 PID   TT     STAT    TIME    COMMAND
```

- `% su Benutzername`
  erlaubt temporären Wechsel auf anderen Benutzer
  su ohne benutzername wechselt auf root
- `%su - Benutzername`
  home directory benutzername
  .login und .cshrc (profiles) werden ausgeführt

Password ändern

- **Gültige Passwörter**
  - + mind. 6 Zeichen lang
  - + mind. 1 Buchstabe

  bei mehr als 8 Zeichen, werden nur die ersten 8 Zeichen verwendet, nicht verwendet sollen werden: Benutzername, eigener Name, Geburtstage, ...
  Optimales Passwort: Groß-, Kleinbuchstaben und Sonderzeichen (Ziffer, ...)
  Keine Umlaute !
- **Ändern des Passworts**

  % passwd Benutzername

  ```
  Changing password for benutzername on LINUXPC
  Old password: _
  New password: _
  Retype new password: _
  ```
- **Wird das eigene Passwort geändert, kann Benutzername weggelassen werden**

man - Page

- **Es gibt 2 Möglichkeiten**
  - **1.)Befehl nicht bekannt**

    % apropos thema oder man -k thema

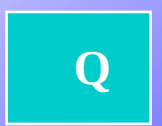
    ls(1) ... Liste von Manualpages die angegebenes Thema

    cp(1) ... behandeln, mit Kurzauszug der Textstelle
  - **2.)Befehl bekannt**

    % man [sektion] thema/befehl

    Manualpage der angegebenen Sektion über Thema oder Befehl.

    Wird keine Sektion angegeben, wird der erste gefundene Handbuchteil ausgegeben.



man Aufbau

- **Manual Pages sind in folgende Sektionen unterteilt:**
  - **man1** allgemeine Befehle und Kommunikation
  - **man2** Systemaufrufe, Fehler
  - **man3** Bibliotheksfunktionen (C, FORTRAN, ...)
  - **man4** Gerätetreiber und -funktionen, Netzwerk
  - **man5** Dateiformate
  - **man6** Spiele
  - **man7** Diverse Befehle (Textverarbeitung, ...)
  - **man8** Systemmanagement und -wartung
- **Will man die manual-page eines Befehls von einer bestimmten Sektion ist diese im man Befehl anzugeben:**
  - **% man tty** liefert tty(1) Befehlsbeschreibung
  - **% man 4 tty** liefert tty(4) Gerätetreiberfunktion

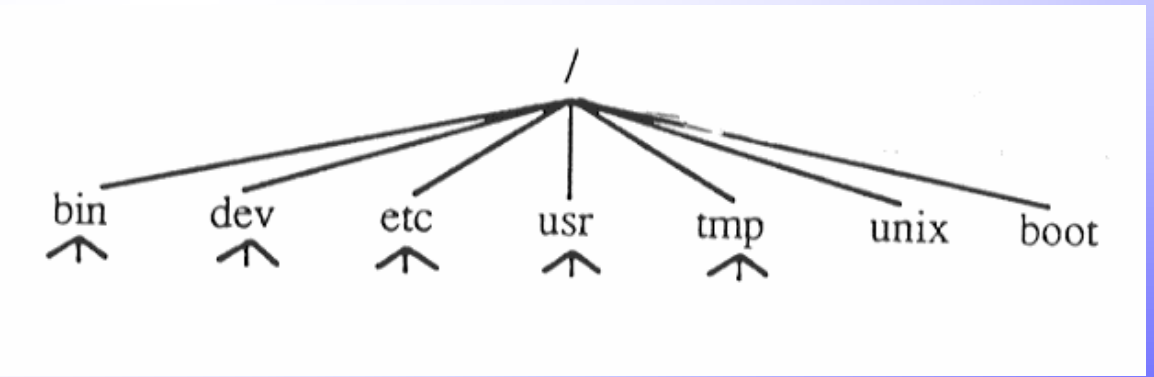
man Aufbau

- **Beispiel:**
  Bitte geben Sie ein:
  %man passwd

- **Die einzelnen man-pages bestehen aus folgenden Teilen**
  - **NAME** Name und Kurzbeschreibung des Themas/Befehls
  - **SYNOPSIS** Syntax oder Format des Befehls/Themas
  - **DESCRIPTION** Erklärung, Verwendung, Argumente
  - **FILES** Dateien die vom Befehl verwendet werden
  - **SEE ALSO** Querverweis auf andere Befehle/Themen
  - **DIAGNOSTICS** Beschreibung der Fehler-/Meldungen
  - **BUGS** Erklärung von bekannten Fehlern und Schwächen

UNIX Dateisystem

- Das File–System (=Dateisystem) ist
  - hierarchisch (baumartig) aufgebaut
  - Ausgangspunkt (Wurzel) ist root-Directory "/"
  - Äste sind Unterverzeichnisse (Directories)
  - Blätter sind Dateien (Files)



File System - Eigenschaften

- **UNIX Filesystem hat**
  - **eine sequentielle, direkt-zugriff Dateiverarbeitung**
  - **erlaubt auf File einen zeichen- oder blockweisen Zugriff**

- **<u>4 Dateiarten werden unterschieden:</u>**

1. **normale Dateien (ordinary files)**
2. **Unterverzeichnisse (directories)**
3. **Geräteinformationsdateien (special files)**
4. **Verweise auf andere Dateien (symbolic links)**

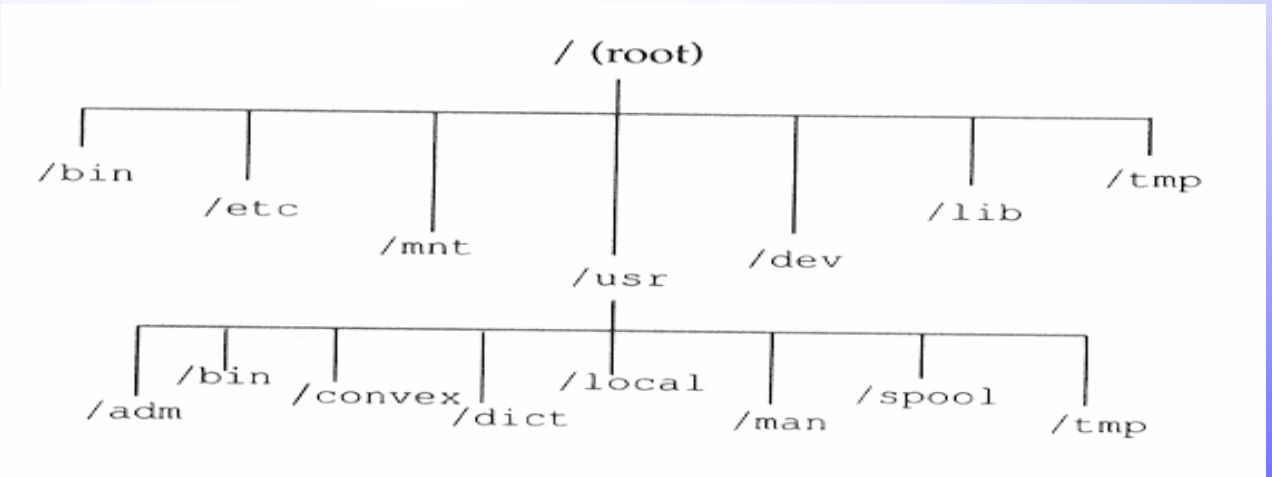
Ordinary Files

- **enthalten Daten, ausführbare Programme**

```
-rw-r-----    1 chris    100          165 Oct 15  1996 text
-rw-r-----    1 chris    100         1547 Apr 19  1999 texttext
-rw-r--r--    1 chris    100       115773 Dec 11  1998 tfessl
-rw-r-----    1 chris    100         2140 Sep  8  1997 tibm
-rw-r-----    1 chris    100         1137 Dec 11  1997 tor
-rw-r--r--    1 chris    100         7442 Sep 29  1998 transalp
-rw-r--r--    1 chris    100        10895 Sep 29  1998 transalp.gif
-rw-r--r--    1 chris    100        12956 Sep 29  1998 transalp.jpg
-rwxr-x---    1 chris    100          428 Jun  4  1996 trash
-rwxr-x---    1 chris    100          242 Jun 25  1996 tree
-rw-r-----    1 chris    100         2008 Jun 25  1996 treelist
-rw-r-----    1 chris    100         3834 Dec 10  1998 ts_suspected-error-report.cgi
-rw-r-----    1 chris    100         1112 Apr 10 11:05 tt
-rw-------    1 chris    100         2364 Nov  9  1998 tt2
-rw-r-----    1 chris    100         4709 Apr 20  1999 ttt
-rw-r-----    1 chris    100           29 Jan 14  1998 ty
-rw-r-----    1 chris    100        24496 Nov 16  1999 uasm
```

Directories

- baumartige file-system (FS) Struktur durch Unterverzeichnisse realisiert
- ausgehend von Wurzel (root "/") können directories angelegt werden
- es kann beliebig weit verzweigt werden
- jedes directory kann beliebige Dateitypen enthalten



Directories

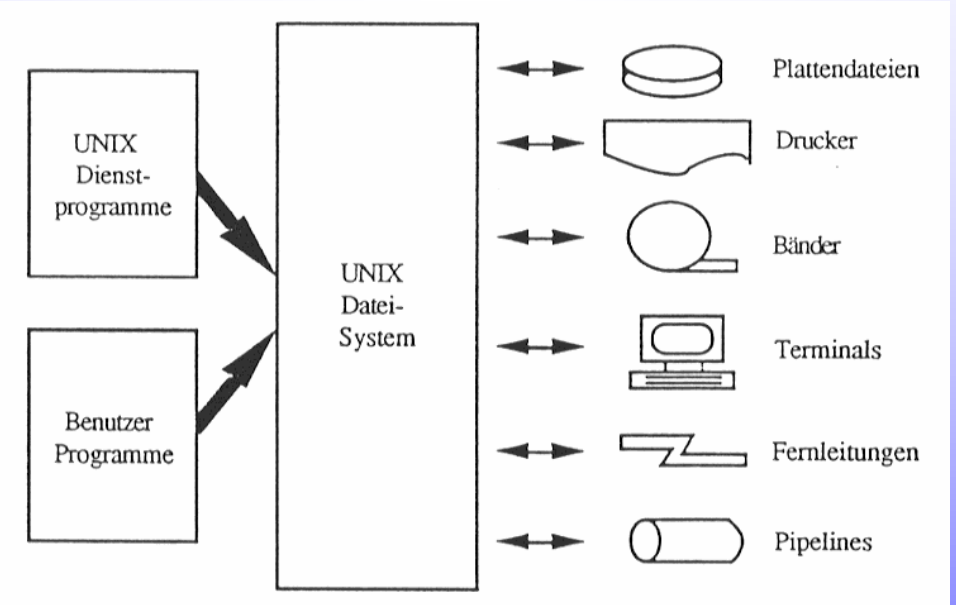
- **Wichtige directories des "root-file-systems"**

directory	Inhalt
/	root, Wurzelverzeichnis, Startpunkt des UNIX-file-systems
/bin	UNIX-Utilities
/etc	Konfiguration, Administrator-Utilities
/dev	special files für I/O-Geräte
/lib	Bibliotheken für F77, C, ...
/tmp	temporäre Dateien
/usr	Utilities, Applikationen, ...

Special Files

- **UNIX behandelt I/O Geräte wie Dateien**
  Bsp.: /dev/tape Bandstation

- **jedem Gerät ist ein special file in /dev zugewiesen**
  - **Geräteunabhängigkeit**
  - **Programme sprechen I/O Geräte über Dateinamen an**



Symbolic Links

- erlauben einmal vorhandene Datei unter verschiedenen Namen, in verschiedenen Positionen im file-system anzusprechen
- Datei ist nur einmal physikalisch vorhanden
- Änderung der Datei unter einem Namen ändert Inhalt unter allen Namen
- wird ein link gelöscht bleibt Datei erhalten, erst wenn letzter link gelöscht ist wird auch Datei gelöscht

(Erstellen von symbolic links wird im System Administrator Kurs erläutert)

Dateinamen

- max. 256 Zeichen lang
- praktisch jedes druckbare Zeichen erlaubt
- A-Z, a-z, 0-9, " _ , . "
- nicht erlaubt " / - & * ? \ ", Leerzeichen
- Groß/Kleinschreibung wird unterschieden
- meist aus 2 Teilen filename.extension

Extension charakterisiert Datei:

.c	C-Programm
.s	Assembler
.f	Fortran Programm
.o	Object Programm (nicht lauffähig)
.filename	Konfigurationsdatei
.out	lauffähiges Programm

Beispiele:

```
rechen.f rechen.o rechen a.out .profile
```

Pfadnamen

- um Datei im file-system zu finden wird der sog. Pfadname (path) benötigt
- path besteht aus den Unterverzeichnisnamen wo der „file“ liegt
- den Pfadnamen des aktuellen Verzeichnisses erhält man mit dem Befehl "present working directory - pwd"

  % pwd

  Nach login steht der Benutzer z.B. in "home-directory", wird von Systemadministrator angelegt, enthält meist Benutzernamen

Pfadnamen

- **2 Arten von Pfadnamen gibt es:**
  1. **absolute Pfadnamen (beginnen bei root "/")**
  2. **relative Pfadnamen (ausgehend vom aktuellen VZ)**

- **1. absolute Pfadnamen**
  - **beginnen bei root "/" dh. "/" am Anfang**
  - **jedes Unterverzeichnis wird mit seinem Namen rechts angehängt, gefolgt von "/" falls weiteres Unterverzeichnis oder Dateiname folgt**

Bsp:

```
   /home/wifi/susi/datei.f
root UVZ  UVZ   UVZ     Dateiname
```

Pfadnamen

- **2. relative Pfadnamen**
  - **relativ zum aktuellen Verzeichnis angegeben**
  - **mit Dateinamen ".." kommt man eine Stufe vor das aktuelle Verzeichnis, beliebig wiederholbar**

Bsp:

aktuell	**`/home/wifi/susi`**
`../`	**`/home/wifi`**
`../../`	**`/home`**

In relativen Pfadnamen ist "." das aktuelle, ".." das übergeordnete Verzeichnis des aktuellen Verzeichnisses

Befehle fürs Dateisystem

- aktuelles Verzeichnis anzeigen

```
% pwd
/home/wifi/susi
```

- aktuelles Verzeichnis wechseln

```
% cd /usr/bin          absolut
% cd ..                relativ
% pwd
/usr
% cd ../bin            relativ
% pwd
/bin
% cd                   zurück ins home-VZ
```

Befehle

- **Inhaltsverzeichnis anzeigen**

```
% ls
mylink prg prog.f texte

% ls -a          alle Dateien
. .. mylink prg prog.f texte

% ls -F          alle Dateien mit Typangabe
mylink@         symbolic link
prg*            ausführbares Programm
prog.f          ordinary file
texte/          directory
```

Befehle

- **% ls -l oder ll Gesamtinhalt lang**

```
total 61
drwxr-xr-x      26 root wheel   1024    Aug 19 08:27 .
drwxr-xr-x      18 root wheel   1024    Aug 19 08:21 ..
lrwxrwxrwx       1 root wheel     10    Aug 19 08:27 adm ->
                                                     ../var/adm
drwxr-xr-x       5 bin  staff   3072    Aug 19 08:22 etc
-rwxr-xr-x       2 bin  staff   4711    Aug 01 13:37 myfile
  Access     Links Owner Group Size    Date           Name Type
```

%ls -al Alle Dateien lang

- **Type**

d directory l link - ordinary file

b block special file c character special file

Befehle

- **Unterverzeichnis anlegen: mkdir directoryname**

```
% cd                                  home directory
% pwd
/home/wifi/susi
% mkdir uvz                           relativ
% cd uvz
% pwd
/home/wifi/susi/uvz
% cd
% mkdir uvz/dir1                      relativ
% mkdir /home/wifi/susi/uvz/dir2      absolut
% ls -F uvz
dir1/ dir2/
```

Befehle

- **Datei kopieren:** **cp oldname newname**

```
% cd                      home directory
% cp /etc/hosts datei              in file "datei"
% cp /etc/hosts .         ins aktuelle VZ
% ls
datei    hosts
% cp datei /tmp/xyz       in anderes VZ
% ls /tmp
.... xyz ....
```

Wird für newname **nur ein Verzeichnisname angegeben, wird die Datei dort mit dem** oldname **angelegt!**

Existiert Datei newname **wird sie einfach mit Inhalt von** oldname **überschrieben.**

Befehle

- **Datei bewegen / umbenennen: mv oldname newname**

  % mv hosts hostnamen

  % mv datei ./uvz

  % ls ./uvz

  .... xyz datei ....

- **Datei löschen:** rm dateiname

  % rm hostnamen

  % ls

  % rm uvz

- **interaktiv mit Frage bei jeder Datei:** rm -i dateiname

Befehle

- **Unterverzeichnis löschen:**
  - **wenn leer:** rmdir dirname
  - **wenn nicht leer:** rm -r dirname

```
% cd
% ls -F
uvz/
% ls -F uvz
dir1/    dir2/
% rmdir uvz/dir1
% ls -F uvz
dir2/
% rm -r uvz
% ls
%
```

Zugriffsrechte

- **pro Datei verschiedene Zugriffsrechte**

Recht	Oktal	Datei	Unterverzeichnis
r (ead)	4	Leseerlaubnis	Inhalt anzeigen
w (rite)	2	Schreiberlaubnis	Dateien anlegen und löschen
x (e.ecute)	1	Ausführerlaubnis	Wechseln ins Verzeichnis erlaubt
-	0	keinerlei Erlaubnis	keinerlei Erlaubnis
t (s.icky)	1000	--------	nur Eigentümer darf Datei löschen

- **Benutzer in 3 Kategorien eingeteilt**
  - **User (u)** Besitzer der Datei
  - **Group (g)** Benutzergruppe der der Besitzer angehört (auch mehrere)
  - **Others (o)** alle anderen die nicht unter u oder g fallen

Es existiert noch ein spezielles Ausführungsrecht (s), welches einem Benutzer oder der Gruppe zur Ausführungszeit die Rechte des Dateibesitzers gibt.

Zugriffsrechte

- **Dateizugriffsrechte werden mit ls -l oder ll angezeigt**

```
%ls -l /
...
drwxr-xr-x    2       bin      staff    6656     Aug 19 08:22 bin
-rw-r-----    1       susi     wifi     4372     Aug 09 18:12 susi
drwxrwxrwt    1       root     staff    3072     Aug 19 08:22 tmp
-rwxr-xr-x    2       bin      staff    4711     Aug 01 13:37 myfile
 111222333            user     group
  Rechte des Benutzers (user)
     Rechte der Gruppe (group)
        Rechte der restlichen Benutzer (others)
```

Bsp: bin

User bin hat alle Rechte, alle anderen haben Lese- und Verzeichnis-wechselerlaubnis

Befehle für die Zugriffrechte

- **Vergabe auf 2 Arten möglich**

  1. **absolut (Oktalzahl)**
  2. **symbolisch**

- **1. absolut:**

chmod oktalwert dateiname

Rechte	Oktal	r w x	Summe
UUUGGGOOO	UGO	4 2 1	
rwxrwxrwx	777	x x x	7
rw-r--r--	644	x x -	6
....			

Befehle für die Zugriffsrechte

- **2. symbolische Vergabe:**

chmod [wer] wie was dateiname

```
wer:  u user   g group   o others   a all
wie:  + erlauben   - streichen   = setzen
was:  r read   w write   x execute   - keine   t sticky
```

Wird wer nicht angegeben, gilt Änderung für user

Bsp:

chmod	**g=rw**	**datei**	**Gruppe darf lesen und schreiben**
chmod	**+r**	**datei**	**erlaube Lesezugriff**
chmod	**u-x**	**datei**	**entziehe Besitzer (user) Ausführerlaubnis**
chmod	**go-w**	**datei**	**entziehe group und others Schreiberlaubnis**

Shell Benutzeroberfläche

- ist Teil der Benutzeroberfläche von UNIX
- analysiert Kommandos (Kommandointerpreter) und führt benötigte Programme aus
- erlaubt Erstellen von Programmen (shell scripts) für komplexe oder oft benötigte Befehlsfolgen
- ermöglicht Ein-/Ausgabeumleitung von Befehlen
- ermöglicht Benutzer Job-Kontrolle
- kann leicht gewechselt werden

Arten der Shell

- **Bourne (sh)**
  - **erster UNIX-Shell**
  - **Start-up-file:** .profile
- **C (csh entspricht der tcsh)**
  - **standard Shell für BSD-UNIX, C-ähnliche Syntax**
  - **Start-up-file:** .cshrc
- **Korn (ksh)**
  - **erweiterte Bourne-Shell, erweitertes Kommandoeditieren (ähnlich csh)**
  - **Start-up-file:** .kshrc
- **LINUX (bash)**
  - **Linux shell**
  - **Start-up-file:** .bash_profile
- **Welche Shell aktiv ist, kann mit** % echo $SHELL **festgestellt werden.**

Login Shell

- **wird bei login aufgerufen**
- **bei Start wird automatisch start-up-file ausgeführt**
- Achtung: **abhängig von der Shell können gewisse Funktionen/Programme nicht mehr richtig funktionieren**
- **Es wird nun die tcsh (bzw. bash) besprochen.**

tcsh-Variable

- tcsh und darin ausgeführte Programme durch Variable konfigurierbar
- 2 Variablenarten in tcsh
  - Umgebungsvariable (environment variables)
  - lokale Variable (local variables)
- Umgebungsvariablen werden bei Aufruf einer neuen tcsh oder eines Programmes vererbt, lokale Variablen nicht.
- Einige Variablen werden von System vorbelegt, können aber vom Benutzer geändert werden.

Umgebungsvariablen

- **Anzeige der definierten Umgebungsvariablen mit**

  % printenv [variablenname]

  HOME=/home/wifi/susi

  SHELL=/bin/csh

  TERM=xterm

  USER=susi

  ...

  Oder z.B.

  % echo $HOME

- **Setzen der Variablen mit**

  %setenv varname [wert]

  wird wert weggelassen wird Variable nur definiert

- **Löschen mit**

  %unsetenv varname

- **In bash mit**

  %set

Standard Umgebungsvariablen

HOME	Pfadname des home-directories
SHELL	Pfadname des login-shells (/bin/csh)
TERM	Terminaltyp (xterm, vt100, ...)
USER	Loginname des Benutzers
PATH	Suchpfad für eingegebene Befehle (einzelne Pfadnamen sind durch ":" getrennt)
HOST	Name des Rechners (LINUXPC, ncube, ...)
EDITOR	Editor der verwendet werden soll (vi, ex, jot, ...)
PRINTER	Name des default-Druckers für lpr Befehl
PAGER	Seitenformatierunsbefehl (less, more,...)
DISPLAY	Name des X-Displays (z.B. 121.22.7.1:0.0)

Lokale Variablen

- **steuern die aktive Shell**
- **werden nicht vererbt**
- **Ausgabe mit**

```
% set
history
home /home/wifi/susi
path (. /usr/bin /bin /usr/etc)
```

- **Setzen mit** **%**set variablenname[=wert]

  Bsp: **%set path=(. /usr/bin /bin)**

- **Löschen mit** **%**unset variablenname

Standard lokale Variablen

home	Pfadname des home-directories (Kopie von HOME)
ignoreeof	wenn gesetzt, ignoriert csh CTRL-D für logout
mail	dateinamen, die nach mail durchsucht werden
shell	Pfadname der Shell
term	Kopie von TERM
user	Benutzername
notify	Schicke Nachricht wenn Job fertig
noclobber	Verhindert Überschreiben von Dateien

tcsh - start-up-files

- **Textdateien die Befehle enthalten**
- **werden von Shell beim Öffnen neuer Shell und/oder bei login ausgeführt**

- **.cshrc** **wird jedesmal beim Starten einer neuen tcsh ausgeführt**
- **.login** **wird nur bei login ausgeführt, hier aber erst nach .cshrc**
- **.logout** **wird bei logout ausgeführt**

- **Änderungen in start-up-files erst nach erneutem login aktiv**
- **Vorzeitige Aktivierung mit**
  % source dateiname **oder** . Dateiname **(in bash)**

Kommandoeingabe

- **Syntax der Kommandoeingabe**

  befehl [optionen] [argumente]

  - **optionen: verändern Arbeitsweise des Befehls**
  - **argumente: Dateien/Directories die Befehl be/verarbeitet**

- **nach jedem Element gehört Leerzeichen**
- **innerhalb der Elemente dürfen keine Leerzeichen sein**
- **Groß/Kleinschreibung ist WICHTIG**
- **da Optionen/Argumente die Arbeitsweise des Befehls verändern, vor erster Verwendung unbedingt man-page lesen!**

Kommandosyntax

- **einfachst: befehl [optionen]**
  Bsp: ls -a ls -F

- **Kombination von Optionen:**
  Bsp: ls -aF entspricht ls -a -F

Beispiele für Kommandos

- ls [option] zeigt Inhaltsverzeichnis
  - option -a zeige auch Dateien mit "." am Anfang
  - -l zeige Inhalt in langer Form
  - -F markiere Dateien je nach Type mit /,*,@
  - -g group anzeigen
- cat [option] dateiname gib Dateiinhalt aus
  - option -n nummeriere Zeilen bei Ausgabe
- who [am i] gibt aktive Benutzer aus
  - option am i gibt Informationen über Aufrufer aus

Dateinamenerzeugung

- **Es sind unvollständige Dateinamen erlaubt**
- **für fehlende Teile werden "Metazeichen" (wildcards) verwendet**

Metazeichen		Funktion
?		ersetzt ein beliebiges Zeichen, ausg. führenden "."
Bsp:	ls key?	ergänzt auf keys, key2, keyx,...
	ls k?y	ergänzt auf key, kay,...
*		ersetzt 0 oder mehrere Zeichen, ausg. führenden "."
Bsp:	ls key*	ergänzt auf key,keys,key.board,...
	ls *.c	ergänzt auf test.c, abc.c, prog.c,...
[ ]		definiert Menge von Zeichen die erlaubt sind
Bsp:	ls key[2468] ergibt	key2, key4, key6, key8
Bsp:	ls key[0-9]	ergibt key0, key1, ..., key9

Namenergänzung in tcsh

- durch TAB-Taste ausgelöst
- csh ergänzt bis Name vollständig oder durch Mehrdeutigkeit eine Entscheidung nötig

Bsp:	more /etc/hosTAB
wird zu:	more /etc/hosts
Bsp:	aprTAB
wird zu:	apropos

- Durch Setzen der lokalen Shell-Variable fignore und recexact kann Ergänzung beeinflußt werden.

Anzeige der Ergänzungen

- ab dem Auftreten von Mehrdeutigkeiten kann mit CTRL-d eine Liste der möglichen Ergänzungen angezeigt werden bzw. nochmals Tab – Taste in bash


```
% lpCTRL-d
lpmv,lpq,lprm,lprman,lprx
% ls
newton newtonville needham boston action cambridge
% more neCTRL-d
newton,newtonville,needham
% more ne
```

Ein- und Ausgabeumleitung

- Standard-Eingabe (standard input) ist vordefinierte Quelle für Kommandoeingabedaten
- Standard-Ausgabe (standard output) ist vordefiniertes Ausgabemedium für Kommandos
- Shell legt standard-input/output auf Tastatur/Bildschirm
- Umleitung der Ein- oder Ausgabe durch Steuerzeichen aus/in Datei

Ausgabeumleitung - output redirection

- **erfolgt durch**

  befehl [optionen, argumente] >datei

- **standard-output wird in datei umgeleitet**

- **ist datei vorhanden, wird sie überschrieben**

- **durch** set noclobber **wird Überschreiben verhindert**

- **will man trotzdem überschreiben >! statt > verwenden**

Bsp: **who >datei**
more datei

Ausgabe an Datei anhängen

- **Ausgabeumleitung erfolgt mit >> statt mit >**

  Bsp:
  % who >datei1
  % date >>datei1

- touch **datei** **erzeugt leere Datei**

  Bsp:
  % touch datei
  % who >>datei

Eingabeumleitung - input redirection

- **erfolgt durch**

  befehl [optionen, argumente] <datei

- **standard-input wird aus datei gelesen**

Bsp:
% who >users

% sort <users >datei **sortiere users zeilenweise**

Pipelines

- leiten Ausgabe eines Befehls in Eingabe des nächsten Befehles
- beteiligte Befehle werden parallel ausgeführt
- Pipelinesymbol: |

Bsp:
% who|sort sortiert who-Bildschirmausgabe

- Pipeline auch mehrmals erlaubt

Bsp:
% who|sort|lpr druckt sortierte Benutzerliste
% cat datei | more

Eingabe mehrerer Kommandos

- **mehrere Kommandos pro Zeile durch ";" getrennt**
- **falls Zeile zu kurz: "\" als Fortsetzungszeichen am Zeilenende, Leerzeichen müssen vor "\" kommen**

Bsp:

```
% who | sort >current ; date >>current; \
cat current
```

Kommandowiederholung - history

- tcsh speichert gewisse Anzahl von zuletzt eingegebenen Kommandos in "history list"
- Ausgabe der history-list mit

```
% history
1 pwd
2 ls
3 vi test
```

- n-ten Befehl wiederholen

```
% !n
```

- history-länge auf n setzen

```
% set history=n
```

- wiederhole letzten Befehl

```
% !!
```

- oder wiederhole letzten Befehl, der mit Buchstaben x begonnen hat

```
% !x
```

history

- tcsh erlaubt Auswahl und Editieren von Kommandos aus "history list"

Taste	Funktion
CTRL-p, ↑	letzten Befehl aus history-list holen
CTRL-n, ↓	nächsten Befehl aus history-list holen
CTRL-b, ←	ein Zeichen nach links
CTRL-f, →	ein Zeichen nach rechts
CTRL-a	an Zeilenanfang
CTRL-e	an Zeilenende
CTRL-h, BKSPC	zeichenweise löschen
CTRL-u	Zeile löschen

Prozeßkontrolle

- jedes laufende Programm ist in UNIX ein Prozeß
- jedes von tcsh ausgeführte externe Kommando ist ein Prozeß
- tcsh ist dann Vaterprozeß (parent)
- Kommando ist Sohnprozeß (child)
- Im Multitasking-Betrieb laufen mehrere Prozesse "gleichzeitig"
- Weiters gibt es den Begriff "JOB":
  job ist Arbeitseinheit, die aus mehreren Programmen
  oder Kommandos bestehen kann

Prozesse

- **solange ein Prozeß läuft hat er eine eindeutige Prozeßnummer PID**

- **mit** % ps

```
PID        TT        STAT      TIME      COMMAND
8596       p0        S         0:01      csh
8315       p0        R         0:00      ps
```

erhält man gerade laufende Benutzerprozesse

PID	**Prozeßnummer**	**TT**	**Terminal**
STAT	**Status**	**TIME**	**CPU-Zeit mm:ss**

- **möglicher Prozeßstatus**

R unning	**T stopped**	**D defunct**
S Active (wartet <20s)	**I idle (wartet >20s)**	**Z zombie**

Prozeß abbrechen

- **Prozeß kann mit**

  % kill PID **abgebrochen werden**

- **Bsp:** **% ps***

```
PID      TT      STAT    TIME    COMMAND
8596     p0      S       0:01    csh
9756     p0      R       0:02    sort
9899     p0      R       0:00    ps
```

```
% kill 9756
[2]Done sort
```

- **Notbremse: kill -9 PID**

Job - control

- job im Vordergrund (foreground) reserviert Tastatur
- job im Hintergrund (background) läuft ohne Tastatur
  - ➔ mehrere background-jobs können gleichzeitig laufen
- laufender foreground-job kann mit CTRL-z gestoppt werden
  - ➔ es erscheint csh-Prompt und erlaubt neue Kommandos
- für einfache Verwaltung, erhält jeder job von csh eine Jobnummer (zusätzlich zur PID)

Job - control

- Anzeigen der gestarteten jobs

```
% jobs
[1]  +          Suspended programm.c
[2]  -          Running   out.c
[3]             Stopped   vi memo
[4]             Running   ls -la >listing
jobnummer
     + aktueller Status   Jobname
     - nächster
```

- der aktuelle job ist immer der zuletzt gestoppte Job

- Job abbrechen:

  % kill %jobnummer oder %kill PID

Job - control

- **neuen job in background schicken durch & am Kommandoende:**

  Bsp: % who | sort >current &

  [1] 01299

- **will man aktuellen job in background stellen:**

  % CTRL-z stoppe job

  % bg in background

- **gestoppten job in background mit:**

  % %jobnummer& oder

  % bg %jobnummer oder

  % bg PID

Job - control

- **gestoppten job in foreground mit:**

  % %jobnummer **oder**
  % fg %jobnummer **oder**
  % fg PID

- **laufenden backgroundjob stoppen mit:**

  % stop %jobnummer **oder**
  % stop PID

UNIX Texteditoren

- erstellen Dateien in ASCII Format
- sind keine Textverarbeitungsprogramme
- alle Editoren arbeiten im Speicher, Änderungen müssen explizit gespeichert werden
- mehrere Varianten
  - zeilenorientiert
  - bildschirmorientiert
  - batchgesteuert

Zeilen- und Batcheditoren

- **Zeileneditoren**
  - **immer nur eine Zeile wird bearbeitet**
  - **Bewegen zwischen Zeilen mit Befehlen**
  - **Editieren in Zeile mit Befehlen**
  - **UNIX: ed, edit, ex**
- **Batcheditoren**
  - **editieren Dateien über Kommandodatei**
  - **für wiederholte, gleichartige Bearbeitung von Dateien**
  - **UNIX: sed**

Bildschirmeditoren

- zeigen Datei immer schirmweise an
- Bewegung in Dateien beliebig (scrolling)
- Cursor zeigt aktuelle Bearbeiteposition an
- Cursor kann auf Bildschirm frei bewegt werden
- nur auf Full-Screen-Video-Terminals lauffähig
- erfordern Umgebungs-Variable TERM
- UNIX: vi
- SGI: jot (X-Windows)
- Andere: emacs (public domain, eigenes Handbuch)

vi

- **"der" UNIX Editor**
- **ausführlichst in man-page beschrieben**
- **2 Betriebszustände**

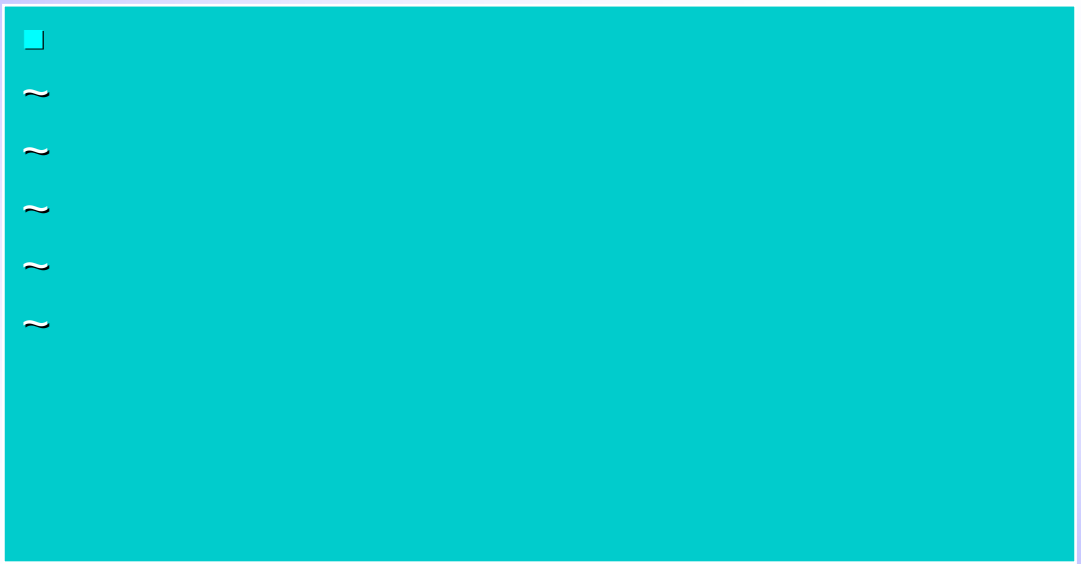
  Kommandomodus: **Cursorbewegung, Suchen, Ändern von Textteilen, Löschen, Undo**

  Insertmodus: **Text eingeben, Tippfehlerkorrektur**

  - ⇨ **durch ESC kommt man in Kommandomodus**
  - ⇨ **im Kommandomodus erzeugt ESC einen Piepton**
  - ⇨ **nach vi-Start immer im Kommandomodus**

vi-Aufruf

- vi dateiname
- ist Datei vorhanden wird ihr Inhalt angezeigt
- bei neuer Datei erscheint:



- Statuszeile am unteren Bildrand: Auskunft über Datei (Name, Zeilen, Zeichen,...)

vi Beenden

- 2 Möglichkeiten
  - Änderungen speichern und Ende (exit)
  - Änderungen verwerfen und Ende (quit)
- Befehl im Kommandomodus
  - ZZ Speichern und Ende
- Befehle im ex-Modus
  - :x abspeichern und Ende (exit)
  - :wq Speichern (write) und Ende (quit)
  - :q! Änderungen verwerfen und Ende (quit)

vi Kommandos

- **vi Kommandos haben folgenden Aufbau**

  [anzahl]operator[anzahl]objekt

  operator: **auszuführende Operation**

  objekt: **Objekt worauf Operation wirkt**

  [anzahl]: **gibt an wie oft Operation auf wieviele Objekte wirken soll**

- Wichtig**: vi-Kommandos werden sofort, ohne RETURN, ausgeführt!**

Bsp:

dw **lösche Wort**

12dw = 3d4w = 4d3w = d12w **lösche 12 Worte**

vi-Cursorbewegung

COMMAND - MODUS

➢ zeichenweise	h	ein Zeichen nach links
	k	eine Zeile nach oben
	l	ein Zeichen nach rechts
	j	eine Zeile nach unten
➢ wortweise	w	nächster Wortanfang (stoppe bei Satzzeichen)
	e	nächstes Wortende (stoppe bei Satzzeichen)
	b	vorhergehender Wortanfang (stoppe bei Satzzeichen)
	W	nächster Wortanfang (ignoriere Satzzeichen)
	E	nächstes Wortende (ignoriere Satzzeichen)
	B	vorhergehender Wortanfang (ignoriere Satzzeichen)
➢ zeilenweise	0	(Null) Zeilenanfang
	$	Zeilenende
	Return	Anfang nächste Zeile

vi-Cursorbewegung

COMMAND - MODUS

andere	(	vorhergehender Satz
	)	nächster Satz
	{	vorhergehender Absatz
	}	nächster Absatz
	nG	gehe zu Zeile n, fehlt n gehe ans Textende
	CTRL-f	eine Seite vor
	CTRL-b	eine Seite zurück
	CTRL-d	halbe Seite vor
	CTRL-u	halbe Seite zurück
	CTRL-e	eine Zeile vor rollen (neue Zeile unten am Schirm)
	CTRL-y	eine Zeile zurück rollen
	z-	aktuelle Zeile an unteren Bildschirmrand bringen
	z.	aktuelle Zeile in Bildschirmmitte bringen

vi-Texteingabe

INPUT - MODUS

- Eingabe mit

a	füge Text rechts von Cursor ein (append)
A	füge Text am Zeilenende an
i	füge Text an Cursorposition ein (insert)
I	füge Text am Zeilenanfang ein
o	füge unterhalb neue Textzeile ein (open line)
O	füge oberhalb neue Textzeile ein

- Befehle schalten in Input-Modus:

BACKSPACE	lösche Zeichen links vor Cursor
RETURN	neue Zeile
ESC	Ende der Eingabe, Kommandomodus

vi-Text löschen und Undo

COMMAND - MODUS

- **Löschen mit**

x	lösche Zeichen unter Cursor
D	lösche von Cursorposition bis ans Zeilenende
d Objekt	lösche angegebenes Objekt
dd	lösche aktuelle Zeile
dw	lösche bis Wortende
db	lösche bis Wortanfang
dG	lösche bis Dateiende
dnG	lösche von aktueller bis n-te Zeile (d1G bis Dateianfang)

- **Undo von Änderungen mit**

u	letztes Kommando rückgängig machen
U	stelle alten Zeilenzustand wieder her

- **WICHTIG: wird Zeile verlassen, ist kein Undo mehr möglich!**

vi-Textsuche

COMMAND - MODUS

- **Suchen mit**

/ausdruck	suche vorwärts nach ausdruck
?ausdruck	suche rückwärts nach ausdruck
n	wiederhole letztes / oder ?
N	wiederhole / oder ? in umgekehrter Richtung

- **Ausdrücke gebildet aus Kombinationen von**

c	beliebiges Zeichen c
\c	spezielles Zeichen c "\[]*$^.?/"
^	Zeilenanfang
$	Zeilenende
[c1c2c3...]	Zeichen aus angegebener Menge
[^c1c2c3...]	kein Zeichen aus angegebener Menge
.	Platzhalter für ein beliebiges Zeichen

vi-Ausdrücke Beispiele

LINUX	einfaches Wort
^LINUX	Wort muß am Zeilenanfang stehen
LINUX$	Wort muß am Zeilenende stehen
LINUX.$	Zeile die am Ende beliebiges Zeichen, davor LINUX hat
^LINUX$	Zeile die nur LINUX enthält
^$	leere Zeile
[LI]inux	Linux, linux
LINUX\.$	Zeile die mit LINUX gefolgt von "." endet

vi-Textänderungen

COMMAND - MODUS

- Einzelzeichen ändern mit

rx	ersetze ein Zeichen unter Cursor mit Zeichen x und kehre in Kommandmodus zurück

- Textteile ändern mit

Rtext	ersetze ab Cursorposition mit text
stext	ersetze Zeichen unter Cursor durch text
Stext	ersetze Zeile durch text
cobjekttext	ersetze objekt durch text (objekt wie oben)
cctext	ersetze aktuelle Zeile durch text
Ctext	ersetze von Cursorposition bis Zeilenende
cwtext	ersetze Wort durch text

  Diese Änderungsbefehle müssen mit ESC nach text abgeschlossen werden!
  vi kehrt dann in den Kommandomodus zurück.

vi-Textteil bewegen und kopieren

COMMAND – MODUS

- vi verwaltet "yank"-Buffer, dessen Inhalt an beliebiger Stelle im Text eingefügt werden kann.
- Befehle

d*objekt*	speichert gelöschten Text in Buffer
y*objekt*	kopiert Text in Buffer
Y	kopiere aktuelle Zeile in Puffer
p	füge Bufferinhalt nach Cursor ein
P	füge Bufferinhalt vor Cursor ein
J	hänge nächste Zeile ans Ende der aktuellen
r	schreibt Text ab Cursor in nächste Zeile

- Wichtig:

7dw	löscht 7 Worte und stellt diese in Buffer
dw	7-mal eingegeben löscht 7-mal ein Wort und stellt daher nur das letzte gelöschte Wort in Buffer

vi - weitere Eigenschaften

- spezielle Funktionen von vi:
  - automat. Einrückung, Tabulatoren, Ränder
  - Makros für wiederkehrende Befehlsfolgen
  - Editieren mehrere Dateien
  - Suchen/Ersetzen
  - Standardeinstellungen (.exrc) z.B. :set nu und :set showmode setzen
  - Anzeigen der Einstellungen
    :set all
  - ex-Kommandos

ex-Kommandos

- **Suchen und Ersetzen eines Ausdrucks in ganzer Datei**

  :g/ausdruck/s//neuer Ausdruck/g

- **Anwenden eines „ex“-Befehles auf bestimmte Zeilen**

  :von,bis Kommando [nach] **(:set number)**

Löschen	**:1,10 d**	
Kopieren	**:5,30 co 100**	**Kopiere Zeile 5 bis 30 nach Zeile 100**
Bewegen	**:7,21 m 28**	**Verschiebe Zeile 7 bis 21 nach Zeile 28**
Speichern	**:3,78 w Datei**	**Schreibe Zeile 3 bis 78 in Datei**

- **spezielle Positionszeichen:**

.	**aktuelle Zeile**
$	**letzte Zeile der Datei**
0	**Dateianfang (vor 1. Zeile)**

Beispiel vi

- **Beispielprogramm potenz mit vi erstellen**

```
#!/bin/csh
if ($#argv == 0) then
echo -n "Geben Sie eine Zahl ein:‘‘
set zahl = $<
else
@ zahl = $argv[1]
endif
@ pow = $zahl * $zahl
echo "Das Quadrat der Zahl ist: $pow"
```

- **Testen**

Kommunikation in UNIX

- Information über Benutzer einholen
- aktive Benutzer ausgeben (Kurzform)

```
% users
root susi otto  ...
```

- aktive Benutzer ausgeben (lange Form)

```
% w
5:36 pm up 4 days, 5 users, load average: 5.29, 5.21, 5.31
User     tty     login@    idle   JCPU  PCPU    what
root     ttyp0   1:28pm    2:55     33     1    -csh
otto     ttyp1   8:05am       1   2:23  1:16    fortran
```

idle	Zeit seit letzter Eingabe (hh:mm)
JCPU	Zeit die alle Prozesse von Benutzer verbraucht haben (hh:mm)
PCPU	Zeit die laufender Prozeß verbraucht hat (hh:mm)

Mail

- **UNIX-Mail erlaubt**
  - **mail an einen/mehrere Benutzer zu schicken**
  - **mail an Benutzer auf anderen Rechnern zu schicken**
  - **mail zu empfangen**
  - **mail weiterzuleiten**
  - **mail zu speichern**
- **Meldungen von csh:**
  - **Neue Mail seit letztem Login**

    `You have new mail`
  - **Ungelesene Mail seit letztem Login**

    `You have old mail`

mail-Aufruf

- **Aufruf von mail mit Ausgabe der aktuellen Mails**

  % Mail **oder:** %mail **(je nach Unix)**

```
"/usr/spool/mail/susi": 2 messages  1 new  1 unread
>U 1 susi        Wed Aug 19 17:03 12/316 "Mail schicke
 N 2 otto        Mon Aug 17 10:05 35/978 "Tape-Hilfe"
&
```

- **Statuszeile zeigt Mail-Systemdatei von Benutzer, Nachrichtenanzahl,... an**
- **& ist Kommandoprompt von mail**
- **Mail verlassen mit**

  CTRL-d, q **führe Veränderungen in Mail aus (löschen,...), Ende**

  x **ändere nichts, Ende**

mail-Kommandos

?	zeige alle mail-Kommandos an
RETURN	zeige nächste Nachricht
-	zeige vorhergehende Nachricht
p	zeige Nachricht nocheinmal
h	zeige Headers (wie nach mail-Aufruf)
R	antworte an Sender einer Nachricht
r	antworte an alle Sender und Empfänger einer Nachricht
d	lösche Nachricht
u	stelle Nachricht wieder her (undelete)
m user	schicke Nachricht an Benutzer user
s datei	speichere Nachricht in datei
w datei	speichere Nachricht ohne Kopfzeilen in datei
x	verlasse mail, stelle gelöschte Nachrichten wieder her
q	verlasse mail, führe Löschungen aus

Alle Kommandos müssen mit RETURN abgeschlossen werden!

Mail versenden

- **aus Kommandozeile**
  - **auf gleichem Rechner**
    - mail benutzer [...]
    - **Bsp: mail root**
  - **an Benutzer auf anderem Rechner**
    - mail benutzer@host [...]
    - **host ist hostname oder ip-Adresse**
    - **Bsp: mail root@cx.ooe.wifi.at.at**
- **innerhalb mail**

  m benutzer[@host] [...]

Mail versenden

% mail root
Subject: Mail an root
Textzeilen werden mit RETURN abgeschlossen.
Abbruch des Mail-Vorganges mit CTRL-c CTRL-c
Folgenden Befehlen (ab 1. Spalte in Zeile) existieren (LINUX)

~t [benutzer ...]	**füge weitere Empfänger hinzu**
~s text	**ersetze Subject durch text**
~r datei	**füge Datei ein**
~p	**zeige gesamten Text an**
~v	**rufe vi zum Maileditieren auf**

Texteeingabe durch "." in der 1.Spalte einer Zeile oder CTRL-d beenden.
Es wird dann die erstellte Mail abgeschickt.
.
Cc: susi otto
%

Mail lesen

% mail

```
"/usr/spool/mail/susi": 2 messages  1 new  1 unread
>U 1 susi  Wed Aug 19 17:03 12/316          "Mail schicken"
 N 2 otto           Mon Aug 17 10:05 35/978          "Tape-Hilfe"
   # Sender         Empfangsdatum    Zeilen/Zeichen Subject
```

U ungelesene Nachricht

N neue Nachricht (seit letztem mail)

> aktuelle Nachricht (wird als nächste gelesen)

Aktuelle Nachricht wird mit RETURN gelesen, andere Nachricht durch Eingabe von Mail-Nummer (#) und RETURN

Mail speichern

- **mail Kommandos**

&s datei	**speichere aktuelle Nachricht in datei**
&s n datei	**speichere Nachricht n in datei**

  Wird Nachricht in bestehende Datei gespeichert, wird sie an das Dateiende angehängt.

  Beim Beenden von mail werden gelesene Nachrichten im home-directory des Benutzers in die Datei "mbox" gestellt.

- **mit Kommando**

  mail -f [datei]

  kann "mbox" oder datei mit mail erneut verarbeitet werden

Antwort auf Nachricht

- **Antwort auf gerade gelesene Nachricht senden**

  &r schicke Antwort an Sender und alle Empfänger

  &R schicke Antwort nur an Sender

  Es werden von mail die Empfänger und Subjectfelder automatisch eingefüllt.

  Dann wird, wie bei m-Kommando, der Nachrichtentext erwartet.

- **Bsp:&R**

```
To: root@cx
Subject: Test-Mail
Hier den Antworttext eingeben
.
&
```

Nachrichten löschen und undo

- **Löschen**

  &d **löscht zuletzt gelesene/aktuelle Nachricht**

- **Undo**

  &u **macht Löschung rückgängig (undo)**

- **d und u können auch mehrere Nachrichten behandeln**

  &d 2 **löscht Nachricht 2**

  &u 2- **hole Nachrichten 2,3 und 4 zurück**

  &d 2 3 9 **lösche Nachrichten 2,3 und 9**

talk

- Ermöglicht durch
  % talk benutzername@host
- 2 Bildschirmteile
  Eingabefenster für Benutzer
  Ausgabefenster für Partner
- Es erscheint `[waiting for your party to respond]`
- Der Partner erhält am Schirm

```
Message from Talk_Daemon at 10:56...
talk: connection request by otto
talk: respond with: talk otto
```

- Will Partner sprechen muß er nach obiger Nachricht
  % talk otto@spp eingeben.
- talk kann von jedem der Gesprächspartner mit CTRL-c abgebrochen werden.
- Will man keine talks führen, kann der Benutzer talk durch
  % mesg n deaktivieren.

TELNET und Remote Login

- **telnet: `telnet [-l username] hostname/ip-adresse`**
  erlaubt Terminalemulation (z.B.:VT100, 3270)

- **rlogin: `rlogin hostname/ip-adresse [-l username]`**
  Von UNIX-System zu UNIX-System;sendet Benutzername mit

- **Um fehlende Angaben werden sie gefragt (Username, Passwort)**

File Transfer Program

- ftp [options] host
  - **erlaubt Filetransfer von und zu anderen Rechnern no secure !**
- **Options:**
  - **-v anzeigen aller Tätigkeiten**
  - **(Weitere Options siehe man-Page)**
- **Beispiel:**

%ftp lilli.wifi.uni-linz.ac.at

```
Connected to lilli.wifi.uni-linz.ac.at
220 lilli FTP sertver ........
Name(lilli.chris): Userid
331 Password required for chris
Password: xxxxxxxx
ftp> Befehlseingabe
```

File Transfer Program

- **Befehle im ftp:**
  - **help**
  - **“Standard” Unix Befehle: dir, cd, mkdir, delete, ...**
  - **Für Filetransfer:**

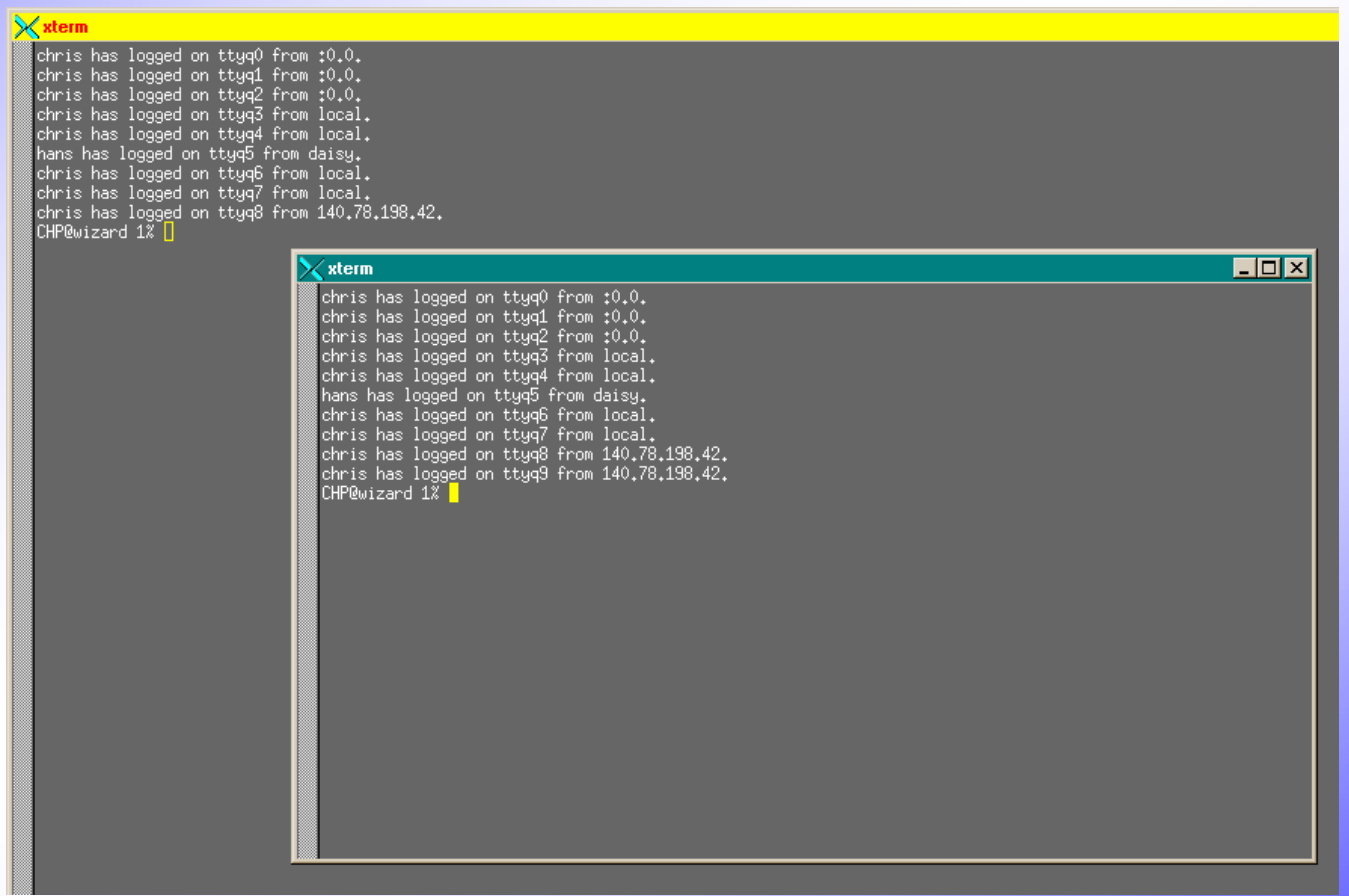
binary	**Binary-Daten-Transfer**
prompt on | off	**Ein-/Ausschalten von Bestätigung bei multiple-file Kommandos**
get remotefile [localfile]	**hole File**
mget remotefiles	**hole mehrere Files**
put localfile [remotefile]	**schicke File**
mput localfile	**schicke mehrere Files**

  - **ftp beenden:**
    quit

(Weitere Befehle siehe man-Page)

Graphische Oberfläche - Xterm

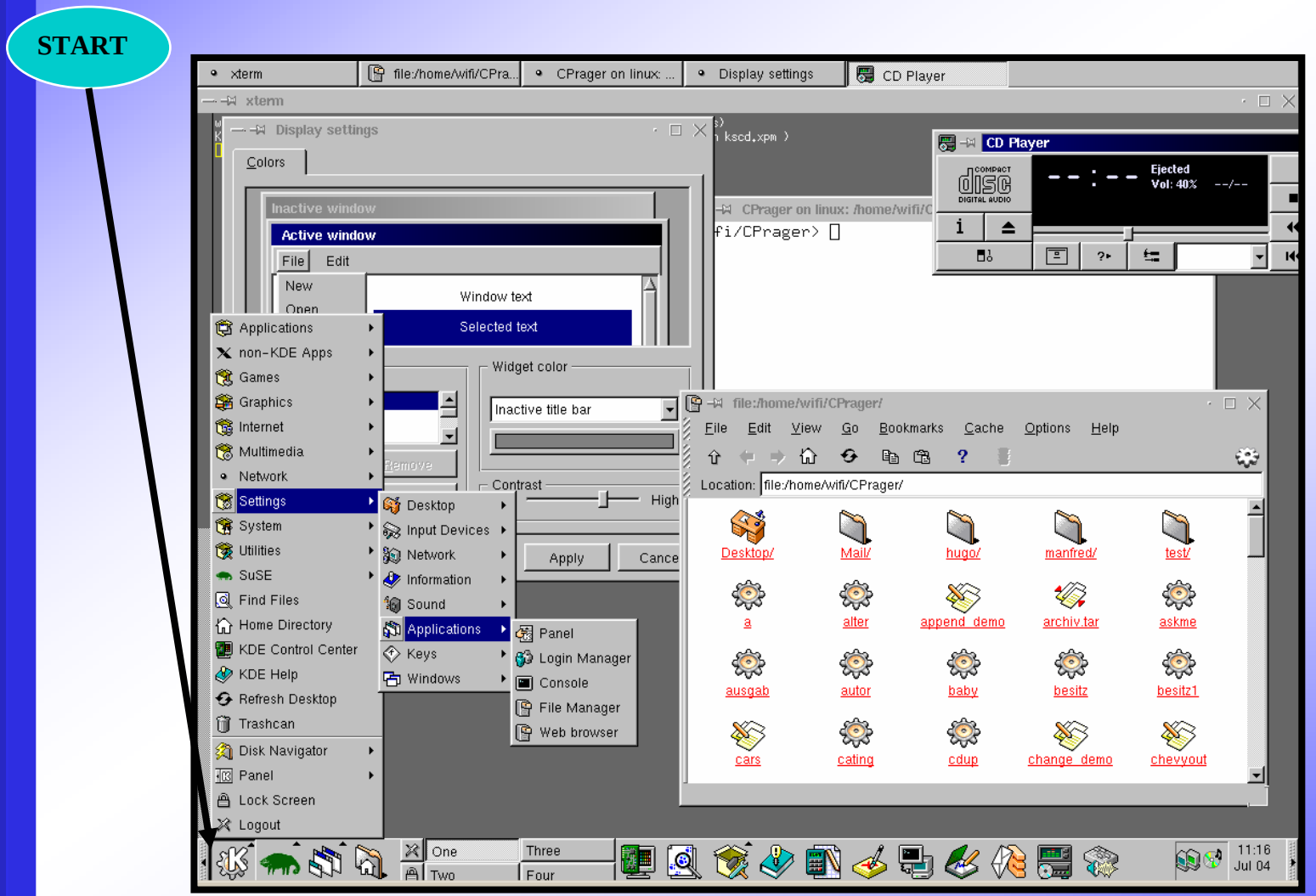
- **Xterm – Fenster:**



Graphische Oberfläche - KDE

- **Beispiel:**
  Graphische Benutzeroberfläche KDE von SUSE
- **Bedienung mit der Maus**
- **Windows ähnlich**
- **„Start“ mittels Symbol links unten**
- **4 Desktops verfügbar (einstellbar)**
- **Fenstertechnik**
- **Office Programme (Star – Office) ausführbar**
- **Viele „public domain“ Programme verfügbar**
- **Mehr System-Overhead**

KDE



DESKTOPs

Installations - Überblick

- **Notwendige Software:**
  Linux-CDs (+ Start-Diskette sofern noch mitgeliefert!) und Handbuch

Der Installation-Ablauf ist SEHR Herstellerabhängig !!!

- **Generelle Vorgehenweise:**
  1. Plattenplatz schaffen: eigene Platte oder vorhandene neu partitionieren
     Speicherbedarf: je nach installierter Software (z.B.: Suse 6.4 umfaßt 6 CD´s)
  2. Booten von CD-Laufwerk (BIOS -Bootsequenz!) oder mittels Start-Diskette
  3. „Have a lot of fun“ Begrüßungsbildschirm und automatischer Start von YaST, dem eigentlichen Installationsprogramm.
  4. Auswahl der Sprache
  5. Konfiguration der Maus
  6. Einstellungen der Tastatur und Uhrzeit
  7. Auswahl von Neuinstallation oder Update
  8. Auswahl der Festplatte und aufteilen dieser in Linux-Partitionen

Installations - Überblick

9. Software-Auswahl: Standard - System EMPFOHLEN
10. Mittels erweiterte Auswahl kann zusätzliche Software ausgewählt werden.
11. Falls sie auf dem Rechner z.B. Windows installiert haben, meldet sich der Bootmanager LILO, der es ihnen im späteren erlaubt, wahlweise UNIX oder Windows zu starten. Falls LILO nicht installiert werden kann, wird eine Start-Diskette zum späteren booten erstellt.
12. Danach ist noch ihr Username und Passwort festzulegen.
13. ACHTUNG: Passwort für root festlegen !!.
14. Beginn der eigentlichen Installation.
15. Start vom Basis-System
16. Definition von Monitor und Grafik-Karte
17. Fertigstellen der Installation (Kernel-Generierung).
18. Login und Start von KDE.

Ausführliche Informationen dazu finden sie im Installations-Handbuch.
Nachträgliche Installation von Software – Komponenten →YaST