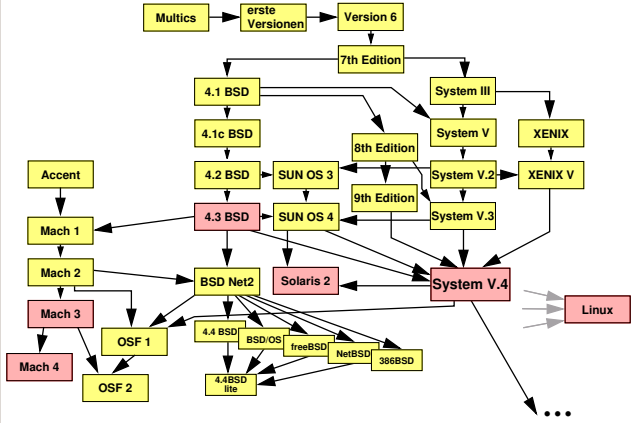
D UNIX-Überblick

- Historische Entwicklung
- Grundkonzepte
  - ◆ Dateien und Dateisystem
  - ◆ Prozesse, Prozeßkommunikation
  - ◆ Betriebsmodi
  - ◆ Interrupts, Traps, Exceptions
- Systemkomponenten
  - ◆ Dateisystem
  - ◆ Prozeßverwaltung
  - ◆ Treiber

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

D.1 Historische Entwicklung (2)



Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

D.1 Historische Entwicklung

1969	erste Implementierung auf einer PDP7 *(Ken Thompson)*
1971	Reimplementierung in der Sprache B
1973	Erste Version in C
1976	UNIX Version 6
1978	UNIX Version 7 - erste portable Systemversion
ab 1980	Entstehung der Berkeley-Linie
	1981 4.1BSD — 1992 BSD Net2
	1983 4.2 BSD — 1994 4.4 BSD
	1986 4.3 BSD
1982	erste kommerziell vertriebene UNIX-Version von AT&T (System III)
1984	UNIX SystemV / Release 2
1986	UNIX SystemV / Release 3
1992	UNIX SystemV / Release 4

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

D.2 Grundkonzepte

1 Dateien und Dateisystem

Verwaltung persistenter Daten — Abstraktion vom Plattenspeicher

- Datei
  - ◆ einfacher, unstrukturierter Bytestrom
  - ◆ kann beliebige Text- oder Binärinformation enthalten
  - ◆ kann dynamisch erweitert werden
- Dateiattribute
  - ◆ Name
  - ◆ Typ
  - ◆ Ortsinformation
  - ◆ Größe
  - ◆ Zeitstempel
  - ◆ Rechte
  - ◆ Eigentümer, Gruppe

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Dateien und Dateisystem (2)

- Dateisystem
  - hierarchisch, baumförmig strukturiert
  - Knoten sind **Dateikataloge *(Directories)***, "Blätter" sind Verweise ***(Links)*** auf Dateien ***(Files)***
  - der oberste Dateikatalog ist die **Wurzel *(Root)*** des Dateibaums
  - jedem Benutzer ist zu jeder Zeit eine **aktuelle Wurzel *(current root)*** und ein **aktueller Dateikatalog *(current working directory)*** zugeordnet
  - Peripheriegeräte (Terminals, Magnetbänder, ...) ***(special files)*** werden analog zu normalen Dateien ***(regular files)*** behandelt
  - Dateien und Kataloge sind mit Zugriffsrechten versehen

© Jürgen Kleinöder, Universität Erlangen-Nürnberg, IMMD IV, 1998
Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

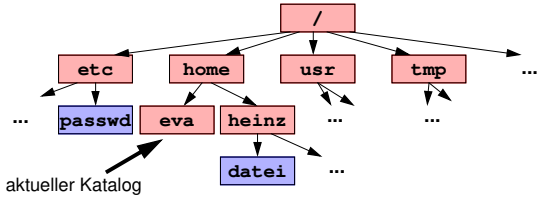
2 Programme und Prozesse

- Programm
  - eine Reihe von Maschinenbefehlen, die in einer Datei abgespeichert sind
- Prozeß
  - die Ausführung eines Programms in der durch die Prozeßdatenstrukturen definierten Umgebung
    - Speicherbereiche
    - Zugriffsrechte
    - Prioritäten
  - mehrere Prozesse können (quasi-)gleichzeitig ablaufen ***(Multitasking)***
  - mehrere Prozesse können das gleiche Programm ausführen, ohne sich gegenseitig zu beeinflussen
  - Prozesse können neue Prozesse (identische Abbilder ihrer selbst) erzeugen ***(fork)***
  - Prozesse können das ablaufende Programm durch ein anderes ersetzen ***(exec)***

© Jürgen Kleinöder, Universität Erlangen-Nürnberg, IMMD IV, 1998
Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Dateien und Dateisystem (3)

- Baumstruktur & Pfade im Dateibaum



  - z.B. „**`/home/heinz/datei`**", „**`/tmp`**", „**`../heinz/datei`**"
  - jeder Katalog enthält Verweise auf sich selbst **(.)** und auf den darüberliegenden Katalog **(..)**
  - mehrere Dateisysteme können zu einem homogenen Baum zusammenmontiert werden

© Jürgen Kleinöder, Universität Erlangen-Nürnberg, IMMD IV, 1998
Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Prozeßkommunikation

- Prozesse können mit anderen Prozessen Daten austauschen
- UNIX stellt dazu folgende Kommunikationsmechanismen zur Verfügung:
- ursprüngliche Mechanismen (UNIX V7)
  - ➥ **Signale**
    - Auslösung einer Unterbrechung bei einem anderen Prozeß
    - Empfänger kann auf die einzelnen Signale gezielt reagieren
  - ➥ **Pipes**
    - gerichtete FIFO-Puffer zur Übertragung beliebiger Daten
    - nur zwischen dem Erzeuger-Prozeß der Pipe und von ihm erzeugten Prozessen, bzw. zwischen seinen Sohn-Prozessen
  - ➥ exit-Status
    - ein Byte, das durch den Vater-Prozeß abgefragt werden kann
  - ➥ Dateien

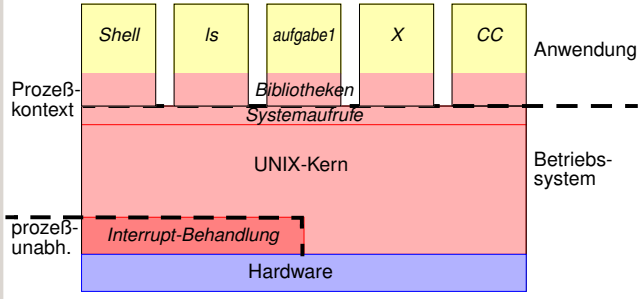
© Jürgen Kleinöder, Universität Erlangen-Nürnberg, IMMD IV, 1998
Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Prozeßkommunikation (2)

- System V - Erweiterungen
  - ➥ **named Pipes**
  - ➥ **Messages**
  - ➥ **shared Memory**
  - ➥ **Semaphore**
- Berkeley-Erweiterungen
  - ➥ **Sockets**
    - Endpunkte von FIFO-Speicher-Paaren — im Gegensatz zu Pipes bidirektionale Kommunikation möglich

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

D.3 UNIX Systemstruktur



Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Benutzermodus

- Programmcode wird von einem Prozeß im Benutzermodus *(user mode)* ausgeführt (nicht privilegiert!)

2 Systemkern

- Alle Funktionen des Betriebssystems sind in einem **monolithischen Kern** zusammengefaßt
- Der Systemkern wird im **privilegierten Modus** (supervisor) ausgeführt
- in der Kern-Umgebung gibt es für jeden Prozeß einen eigenen Stack (lokale Daten), jedoch nur ein Datensegment (globale Daten) für alle Prozesse gemeinsam

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Systemkern (2)

- **Prozeßwechsel** im Systemkern sind nur an fest definierten Stellen möglich, an denen die globalen Datenstrukturen in konsistentem Zustand sind
  - ➥ Probleme bei Multiprozessorsystemen!!!
- Hardware-Interrupts bewirken den Aufruf einer *Interrupt Service Routine*, die die Daten von der Hardware annimmt und ggf. Prozesse aufweckt
- Interrupt Service Routinen laufen in der Kern-Umgebung ab, haben aber keinen Prozeßkontext
  - ◆ Interrupt Service Routinen können auf die Datenstrukturen des Kerns zugreifen
    - ➥ während Manipulationen an Kern-Datenstrukturen, die nicht unterbrochen werden dürfen, müssen Interrupts verhindert werden

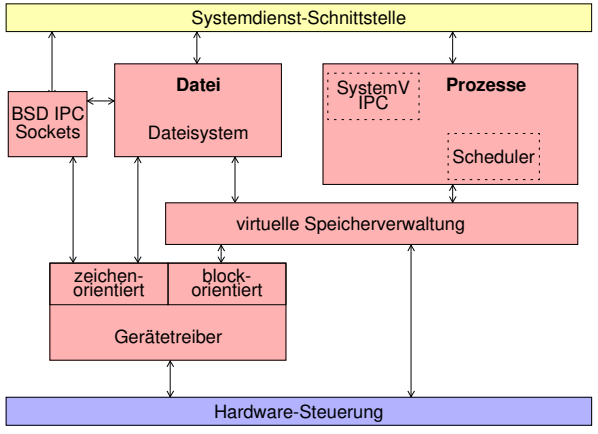
Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Wechsel zwischen Benutzermodus und Systemkern

- Betriebssystemdienste führt ein Prozeß durch
  - Wechsel in den **System-Modus *(kernel mode)*** (privilegiert!)
  - und Aufruf von Funktionen des UNIX-Kerns aus
    - ➥ ein Prozeß fordert nicht einen Betriebssystemdienst von einem Kern-Prozeß an, sondern sein Aktivitätsträger wechselt in den Systemkern und führt dort den Systemdienst selbst aus!
- der Wechsel zwischen Benutzer- und Kernelmodus geschieht durch ***Traps* (*system calls*, *SVCs*)**
- ***Exceptions*** (z. B. Division durch 0) bewirken ebenfalls einen Wechsel in den Systemkern, wo die Fehlerbehandlung veranlaßt wird

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

D.4 Struktur des Systemkerns



Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

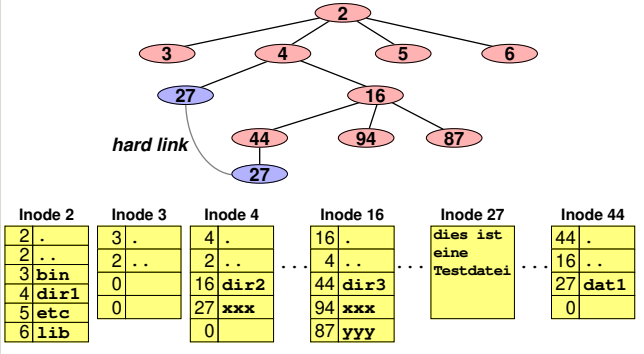
D.5 Dateisystem

- jede Datei (*Directory*, *special file*, *named pipe*, ...) wird durch einen **Inode** vollständig beschrieben
- die **Directories** enthalten nur die Zuordnungen ***Dateiname* ⇒ *Inode* *(Link)***
- es können mehrere Dateinamen auf den gleichen Inode verweisen (nur innerhalb eines Dateisystems!): ***Hard-Links***
- die Zuordnung ***Pfadname* ⇒ *Inode*** erfolgt durch Absuchen des Pfads:
  - *Directory einlesen*
  - *Pfadkomponente suchen und Inodenummer feststellen, ...*
  - Kernel-Funktion ***lookup***

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

D.5 Dateisystem (2)

- Directory-Baum als Namensverzeichnis für das (eigentlich flache) Dateisystem



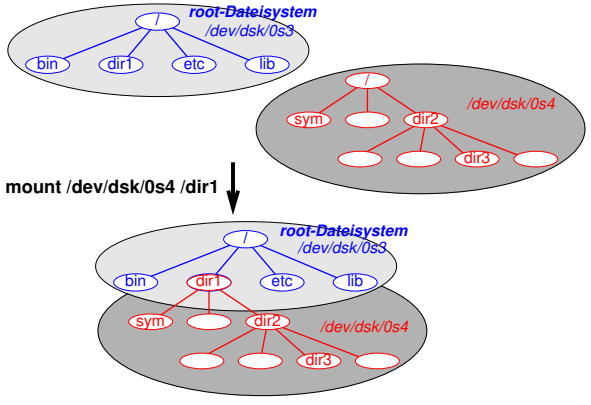
Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

D.5 Dateisystem (3)

- der UNIX-Dateibaum kann aus mehreren Dateisystemen - auch auf verschiedenen physikalischen Platten - bestehen
- jedes Dateisystem besteht aus einem Dateibaum mit einer ***Root***
- durch ***Mounten*** kann die Root eines Dateibaums an einen beliebigen Knoten eines anderen (bereits gemounteten) Dateibaums angebunden werden
- ***Network File System (NFS)*** erlaubt auch das Mounten von beliebigen Directories anderer Rechner

© Jürgen Kleinöder, Universität Erlangen-Nürnberg, IMMD IV, 1998
Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

D.5 Dateisystem (4)



© Jürgen Kleinöder, Universität Erlangen-Nürnberg, IMMD IV, 1998
Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

D.6 Prozeßverwaltung

1 Prozeß

- Ausführung eines Programms in der durch die Prozeßdatenstrukturen definierten Umgebung

2 Datenstrukturen zur Ausführung eines Programms

- Programmkode und Programmdaten werden in **Segmenten *(Regions)*** organisiert
- abhängig vom gerade auszuführenden Programm, können diese während der Lebenszeit eines Prozesses gewechselt werden

© Jürgen Kleinöder, Universität Erlangen-Nürnberg, IMMD IV, 1998
Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Datenstrukturen zur Ausführung eines Programms (2)

- **Text-Segment**
  Maschineninstruktionen des Programms
  (in der Regel schreibgeschützt, wird von mehreren Prozessen gemeinsam benutzt)
- **Daten-Segment**
  Daten des Programms (global und *static*), dynamisch erweiterbar (*malloc(3), sbrk(2)*)
- **Stack-Segment**
  lokale Daten und Aufrufparameter von Funktionen
  sowie Sicherungsbereiche für Registerinhalte und Rücksprungadressen
  — wächst bei Bedarf
- **shared Daten-Segment (optional)**
  gemeinsamer Datenbereich für mehrere Prozesse

© Jürgen Kleinöder, Universität Erlangen-Nürnberg, IMMD IV, 1998
Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Datenstrukturen zur Abwicklung eines Prozesses

weitgehend unabhängig vom gerade abzuwickelnden Programm

- ***proc*-Struktur**
  Prozeßdaten, die während der Lebenszeit eines Prozesses ständig im Hauptspeicher benötigt werden
  z. B.
  - Hauptspeicher/Disk-Adressen der Segmente
  - Prioritäten
  - Signale

- ***user*-area**
  Daten, die zur Ausführung eines Prozesses benötigt werden
  z. B.
  - Segment-/Seiten-Kachel -Tabellen
  - *signal-handler*
  - Zugriffsrechte
  - Laufzeiten

  außerdem Stack für Abwicklung von Funktionen des Systemkerns *(kernel-stack)*

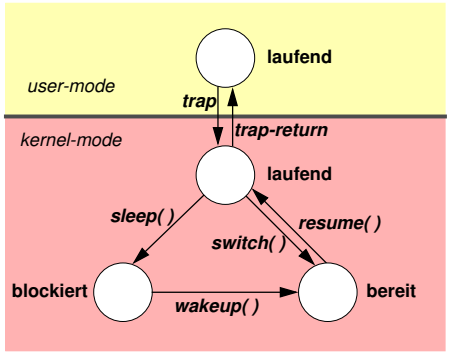
© Jürgen Kleinöder, Universität Erlangen-Nürnberg, IMMD IV, 1998
Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Multiprogramming, Scheduling

- UNIX erlaubt die (quasi-)gleichzeitige Abwicklung mehrerer Programme (Prozesse)
- Umschaltung zwischen Prozessen durch **Prozeßwechsel *(context switching)***
- Prozeßwechsel erfolgen
  - ◆ wenn Prozesse warten müssen (z. B. auf E/A), oder
  - ◆ nach einer bestimmten Laufzeit — **Zeitscheibe *(time slice)***
- die Entscheidung, welcher Prozeß als nächstes den Prozessor zugeteilt bekommt ***(Scheduling)*** erfolgt auf der Basis **dynamischer Prioritäten *(multi-level feedback)***

© Jürgen Kleinöder, Universität Erlangen-Nürnberg, IMMD IV, 1998
Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

5 Prozeßzustände



© Jürgen Kleinöder, Universität Erlangen-Nürnberg, IMMD IV, 1998
Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

D.7 Treiber

- Treiber erzeugen die Verbindung zwischen dem Systemkern und Peripheriegeräten (Terminals, Drucker, Platten, Magnetbänder, ...)

1 *raw-* oder *character-device driver*

- E/A erfolgt zeichenweise oder in beliebigen Blöcken
- Kommunikation mit der Peripherie erfolgt in der gleichen Reihenfolge, in der die Daten angefordert/geliefert werden

2 block-device driver

- E/A erfolgt in Blöcken (512 Byte, 1K, 4K oder 8K).
- die Blöcke werden im System gepuffert und in möglichst günstiger Reihenfolge von/zur Peripherie übertragen

© Jürgen Kleinöder, Universität Erlangen-Nürnberg, IMMD IV, 1998
Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.