Fachhochschule Nordwestschweiz
Hochschule für Technik

Modul Betriebssysteme (bsys-iC)

Feedback aus der Hausaufgabe

- Was ist Ihnen aufgefallen?
- Gab es grundlegende neue Erkenntnisse?
- Was hat gefehlt?
- Wieviel Zeit haben Sie aufgewendet?

Lektion 4: Dateisystem, Datensicherung und Ein-/Ausgabesystem

Inhalt

- Anforderungen an ein Dateisystem und schematischer Aufbau eines Dateisystems am Beispiel Unix UFS
- Benutzungs- und Administrationssicht auf ein Dateisystem am Beispiel Unix UFS (Navigation, Zugriffsrechte, SetUID-Mechanismus, Mounting, Geräte-Spezialdateien, Backup/Restore)
- Design-Kriterien für ein Dateisystem (Block-grösse, Anzahl Dateien / Inodes, Verteilung auf Partitionen etc.)
- Das Ein-/Ausgabesystem

Motivation

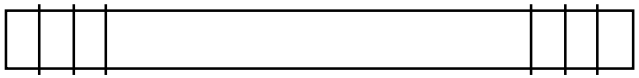
Anforderungen an ein Dateisystem

- Konsistente, permanente Ablage von Dateien
- Strukturierung der Ablage (Hierarchie, Extensions, …)
- Zugriffssicherheit (Rechte/Rollen, Attribute)
- Basisoperationen (Benutzer-Navigation, Datei-Manipulationen)
- Programmierbarer Zugriff
- Mehrbenutzer/-prozessfähigkeit
- Performance
- Standardisierung
- Weitere?

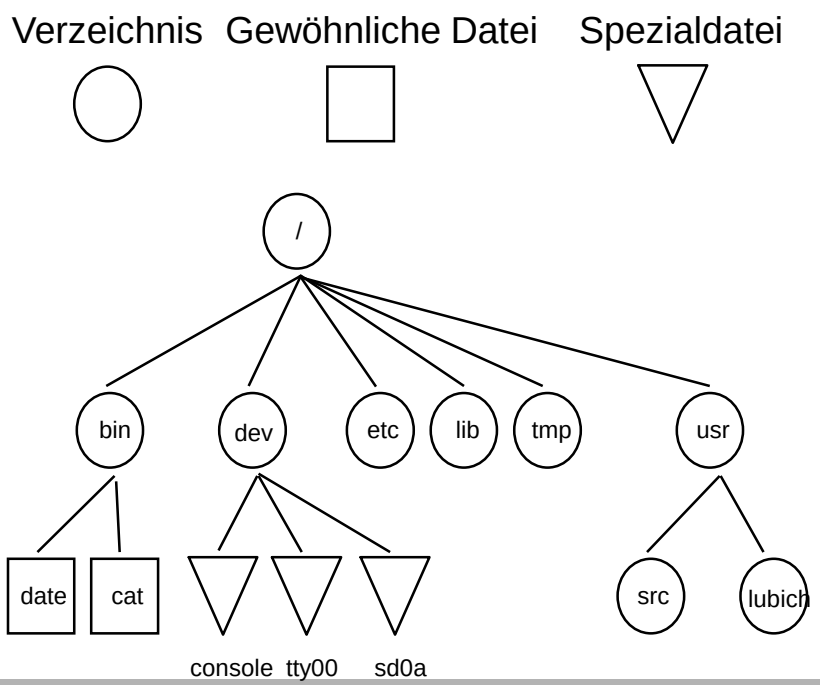
Schematischer Aufbau eines Dateisystems am Beispiel Unix UFS

Fachhochschule Nordwestschweiz
Hochschule für Technik

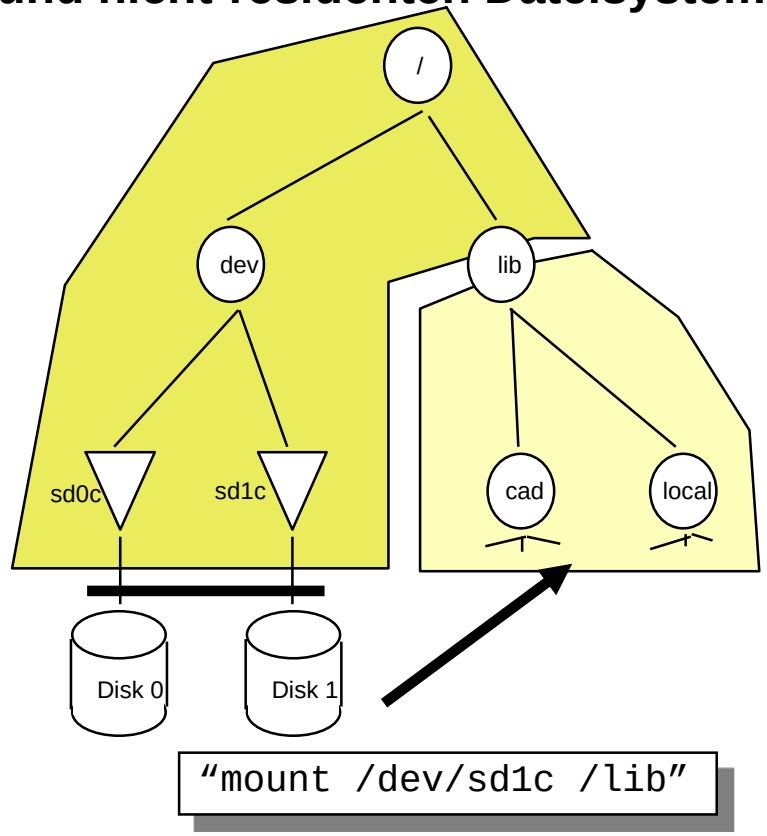
gewöhnliche Datei



Verzeichnisse



Virtuelles Dateisystem mit Wurzel- und nicht-residenten Dateisystemen



Nach: R. Adamov, VL: UNIX-Betriebssystem und Werkzeuge, Univ. Zürich

Beispiel eines Directory-Layout (/etc)

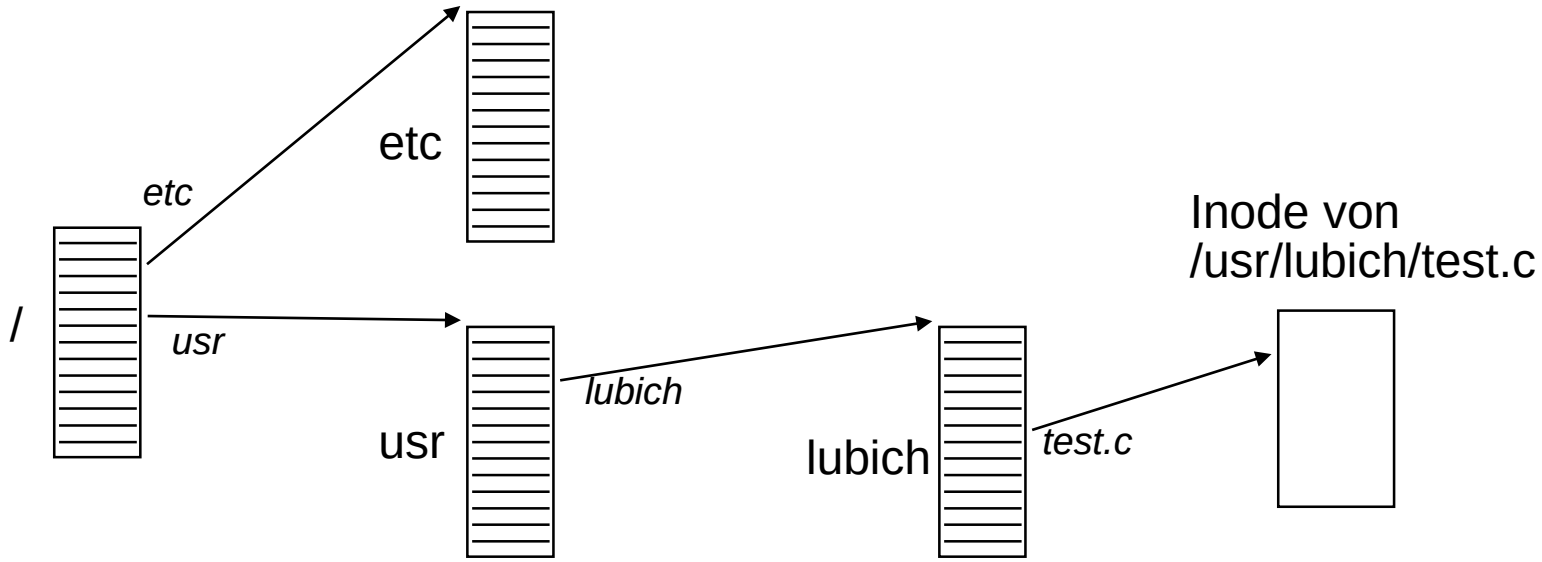
„Hard Link“ →

„Symbolic Link“ / „Soft Link“, ggf auch Cross-Device →

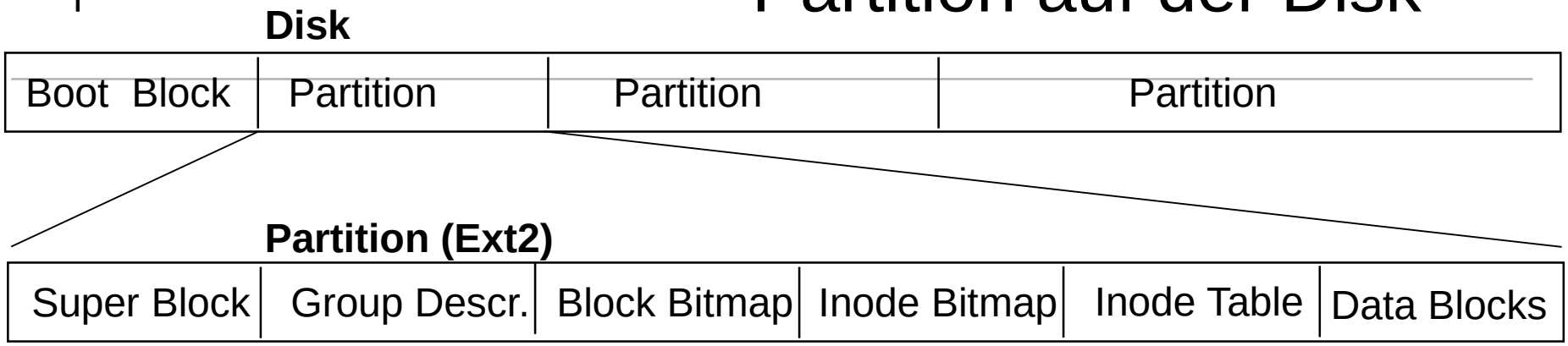
Byte-Offset in Directory	Inode Number (2 bytes)	File Names
0	83	.
16	2	..
32	1798	init
48	1276	fsck
64	85	clri
80	1268	motd
96	1799	mount
112	88	mknod
128	2114	passwd
144	1717	unmount
160	1851	checklist
176	2004	msg -> motd
192	84	config
208	1432	getty
224	0	crash

Nach M.J. Bach, The Design of the UNIX Operating System, Figure 4.10

Aufbau eines Dateibaums aus Directories



Ein Dateisystem / Partition auf der Disk



Block-Allokationsmethoden:

- Contiguous (ganze Dateien)
- Verlinkte Blöcke
- File Map (Landkarte)
- Index-Allokation (z.B. in Linux)

Super Block Inhalt:

- Anzahl inodes & Datenblöcke
- Adresse des 1. Datenblocks
- Anzahl freie Blöcke & inodes
- Grösse eines Datenblocks
- Blöcke / inodes pro Gruppe
- Anzahl Bytes pro inode

Design-Kriterien für ein Dateisystem

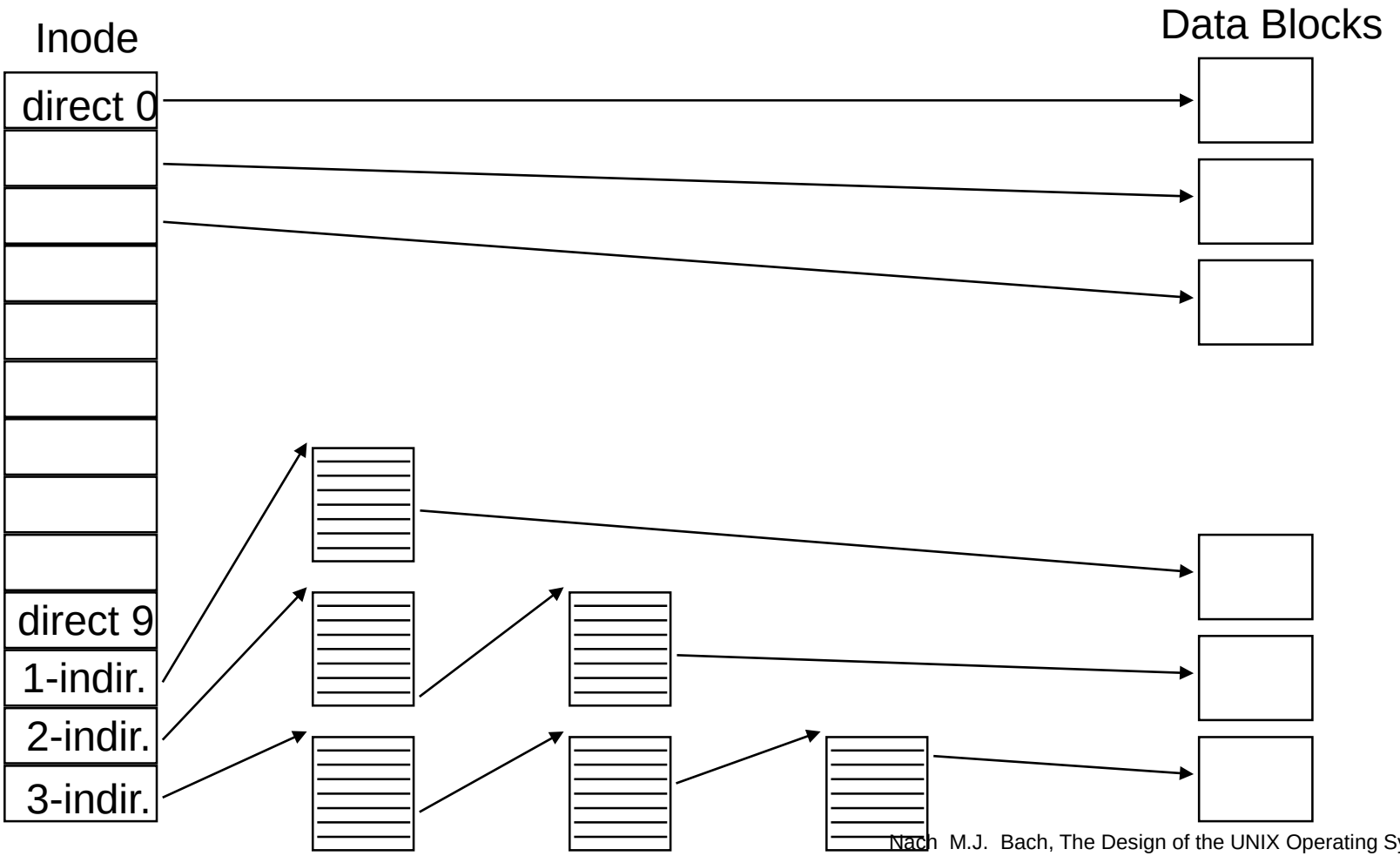
- Anzahl Disks & Disk Controller
- Verteilung auf Partitionen
- Blockgrösse pro Partition
  - Grosse Blöcke: schneller Zugriff auf Dateien (Datei-Durchschnittsgrösse < 4 kB)
  - Kleine Blöcke: weniger interne Fragmentierung
- Anzahl Dateien / Inodes pro Partition
  - Wenige Inodes: mehr Platz für Dateiblöcke
  - Viele Inodes: mehr Dateien pro Partition mgl.

Design eines Inode (Linux)

- **Auf der Disk:**
- Inode Nummer
- Anzahl hard links
- Typ (-, l, d, b, c, p)
- Rechte (-, r, w, x, s)
- Besitzer
- Gruppe
- Grösse
- Letzte “access time”
- Letzte “content change time”
- Letzte “inode modification time”
- Datenblock Pointer

- **Zusätzlich im Memory:**
- Link auf die “hash list”
- Link auf die “inode list”
- Benutzerzähler
- Gerätenummer
- “Device special file” Indikator
- Grösse eines Blocks
- Anzahl Blöcke
- Lock auf den inode
- “Mount point” Indikator
- Warteschlange wartender Prozesse
- Locks auf die Datei
- Hauptspeicher-Region für “Memory-mapped file I/O”
- Belegte Seiten im Hauptspeicher

Referenzierung von Datenblöcken eines Inode



Nach M.J. Bach, The Design of the UNIX Operating System, Figure 4.6

Maximale File-Grössen (1 Block = 1 KBytes)

10 direkte Blöcke mit je 1 Kb	10 Kilobytes
1 indirekter Block mit 265 direkten Blöcken	256 Kilobytes
1 doppelt indirekter Block mit 256 indirekten Blöcken	64 Megabytes
1 dreifach indirekter Block mit 256 doppelt indirekten Blöcken	16 Gigabytes

Nach M.J. Bach, The Design of the UNIX Operating System, Figure 4.7

Gibt es wirklich 16 GB grosse Files?
Welche Limitierungen bestehen?

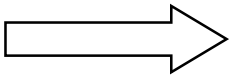
Finden eines Disk Block und eines Inodes

Blocknummer =
((Inode-Nummer - 1) DIV (Anzahl Inodes pro Block))
+ Adresse des ersten Blocks der Inode-Liste

Inode-Byte-Offset =
((Inode-Nummer -1) MOD (Anzahl Inodes pro Block))
* Grösse eines Disk-Inode

Typen von Unix-Dateisystemen

- spec — Special (/dev) FS
- vxfs — Veritas FS
- cdfs — CD ROM
- ufs — Unix (fast) FS
- nfs — Network FS
- fd — Filedescriptor FS
- fifo — Pipe FS
- proc — Prozess FS
- s5 — Traditional System 5 FS
- rfs — Remote File Sharing (AT&T)
- bfs — Boot FS
- dos — DOS FS
- ... — Es kommen ständig neue FS-Typen dazu, z.B. im Linux Kernel 2.6.30 neu exofs (extended object fs) und nilfs2 (log-structured fs v2 für „continuous snapshotting“), neu ist btrfs der Kandidat als neues Standard-Linux-FS



Benötigt:
einheitliche
Abstraktion
für File-Handling
→ VFS

Pause

Übung
(ca. 30 min.)

- Aufgabe(n) gemäss separatem Aufgabenblatt
- Lösungsansatz: Einzelarbeit oder Gruppen von max. 3 Personen
- Hilfsmittel: beliebig
- Besprechung möglicher Lösungen in der Klasse (es gibt meist nicht die eine «Musterlösung»)

Übungsbesprechung (ca. 15 min.)

- Stellen Sie Ihre jeweilige Lösung der Klasse vor.
- Zeigen Sie auf, warum ihre Lösung korrekt, vollständig und effizient ist.
- Diskutieren Sie ggf. Design-Entscheide, Alternativen oder abweichende Lösungsansätze.
- Gibt es Unklarheiten? Stellen Sie Fragen.

Pause

Benutzungs- & Administrationssicht auf ein Dateisystem am Beispiel Unix UFS

- Navigation
  - Darstellung von Datei- und Pfadnamen
  - Absolute & relative Adressierung
  - Benennung des aktuellen und des direkt übergeordneten Directories
  - Anlage von Directories
  - Hard und Soft Links
  - Kommandos: cd, pwd, mkdir, rmdir, …

User-ID's in Linux und zugehörige Privilegien I

- Unix/Linux unterscheidet verschiedene Benutzer und kann diesen Zugriffs- und Ausführungsrechte zuordnen.
- User-ID‘s sind numerisch, werden aber als Gründen der Benutzerfreundlichkeit auf Benutzernamen abgebildet.
- Zusätzlich verwaltet Unix/Linux Benutzer-gruppen, denen ebenfalls Zugriffs- und Ausführungsrechte zugeordnet werden.

User-ID's in Linux und zugehörige Privilegien II

- Die zentrale Datenbank für User-IDs, Benutzer-namen und andere Information für den Benutzer ist die Datei /etc/passwd.
- Jede Zeile in /etc/passwd beschreibt einen Benutzer und seine Attribute:

```
root:asd%sZsd:0:1:Super-User:/root:/sbin/sh
daemon:x:1:1::/:
bin:x:2:2::/usr/bin:
lubich:Hx&wrt%d:20:10:Hannes Lubich, FHNW, 4.317:/usr/lubich:/bin/csh
nobody:x:60001:60001:Nobody:/:
```

- Jede Zeile in /etc/group beschreibt eine Gruppe und nennt ihre Mitglieder:

```
root::0:root
bin::2:root,bin,daemon
sys::3:root,bin,sys,adm
adm::4:root,adm,daemon
usr::10:lubich
```

User-ID's in Linux und zugehörige Privilegien III

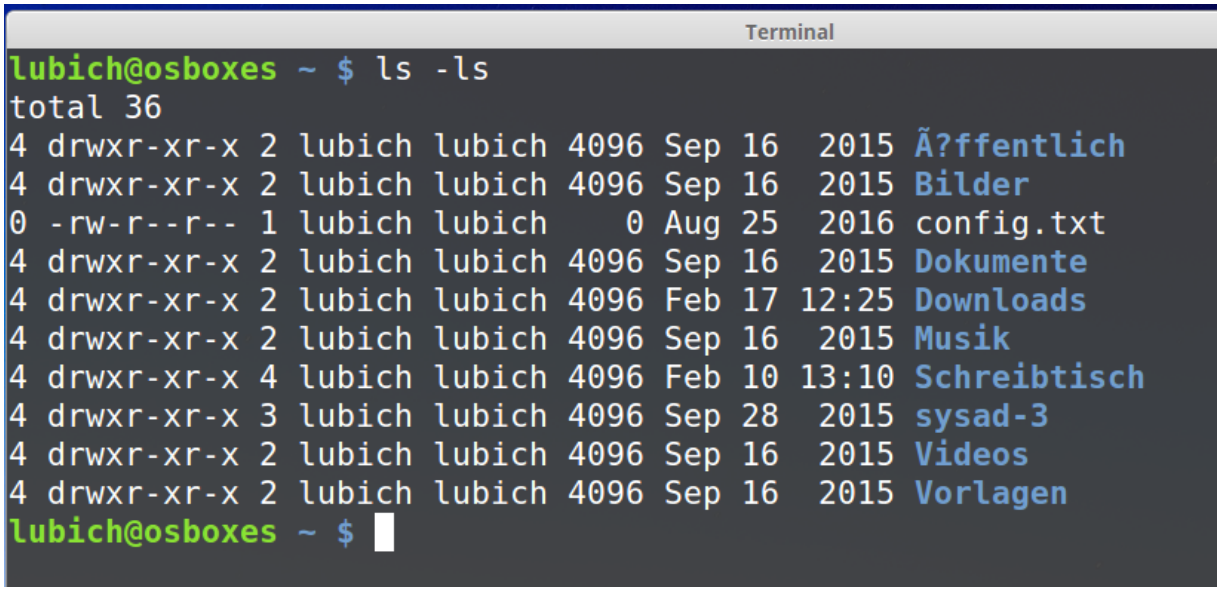
- Benutzerrechte sind an die Attribute von Dateien geknüpft. Diese Rechte entscheiden, welche Benutzer oder Grupenmitglieder eine Datei lesen, verändern oder ausführen können.
- Loggt sich ein Benutzer ein, läuft die Start-Shell mit der User-ID des Benutzers ab – alle weiter durch den Benutzer gestarteten Prozesse haben die gleiche User-ID.
- Wird eine Datei ausgeführt (d.h. es wird ein Prozess erstellt), läuft dieser mit den Rechten des Benutzers/Aufrufers, nicht des Besitzers.

User-ID's in Linux und zugehörige Privilegien IV

- Jeder Benutzer ist selbst verantwortlich für die Zuteilung von Zugriffsrechten für seine oder ihre Dateien (discretionary access control, DAC) – eine Shell-Variable (die „umask“) legt einen vom Benutzer veränderbaren Defaultwert fest.
- Manche hochsicheren Unix/Linux-Varianten erzwingen stattdessen eine Daten-Klassifikation, aus der dann die Zugriffsrechte automatisch abgeleitet werden (mandatory access control, MAC).

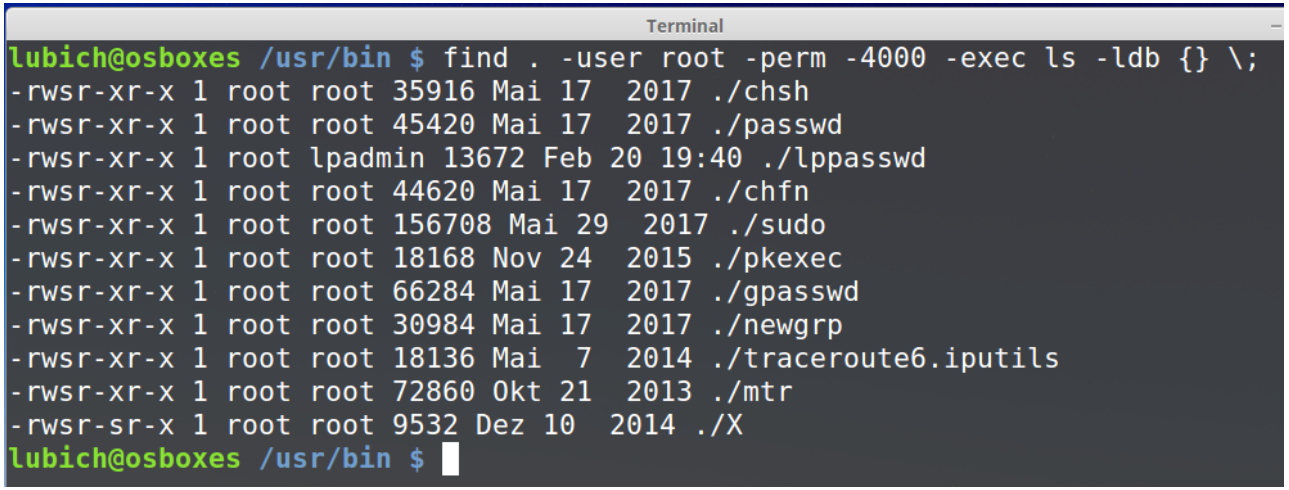
Benutzungs- & Administrationssicht auf ein Dateisystem am Beispiel Unix UFS

- Zugriffsrechte
  - Benutzer
  - Gruppen
  - „Rest der Welt“
  - Rechte
    - Read
    - Write
    - Execute
    - Warum kein „delete“?
    - Warum kein „create“?
- Änderung: chown, chgrp, chmod (u+x oder 755)



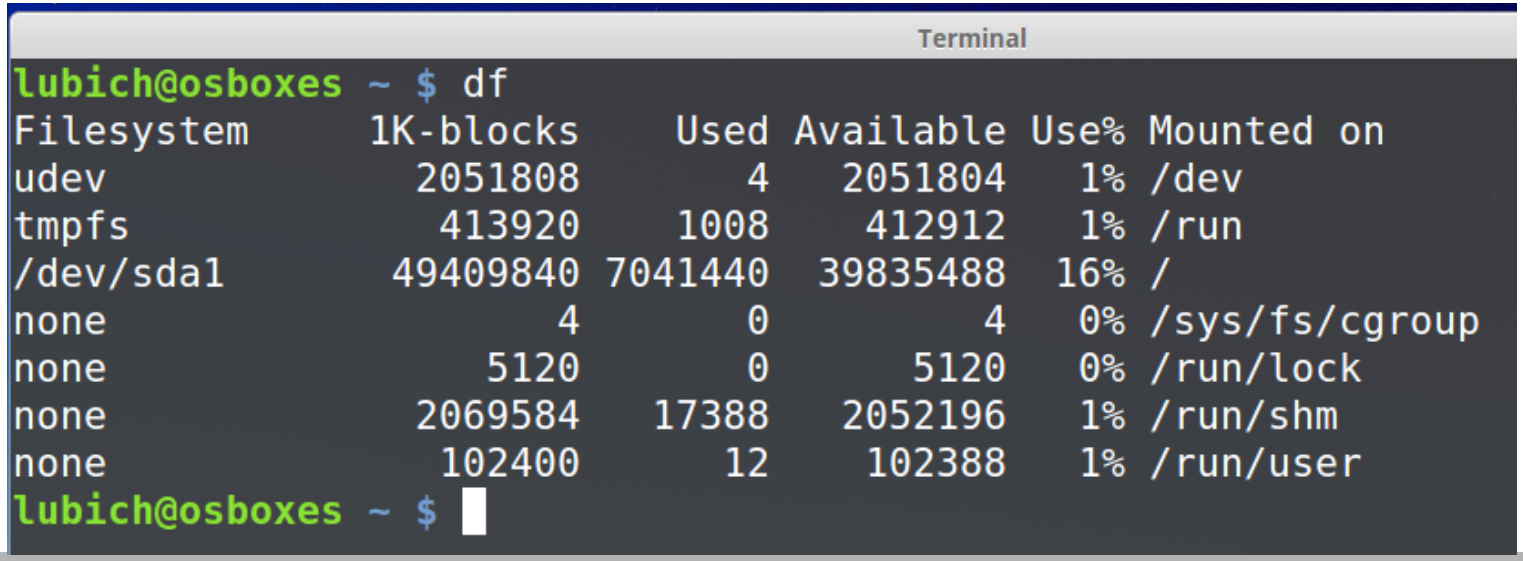
Benutzungs- & Administrationssicht auf ein Dateisystem am Beispiel Unix UFS

- SetUID/SetGID
  - Ersetzt die Berechtigungen des Benutzers oder der Gruppe während der Ausführung eines Programms durch die Rechte des Programm-besitzers.
  - Beispiel: „passwd“ zum Wechseln des Passwortes (setuid).
  - Mächtiger Mechanismus mit hohen Risiken bei unvorsichtiger Benutzung.

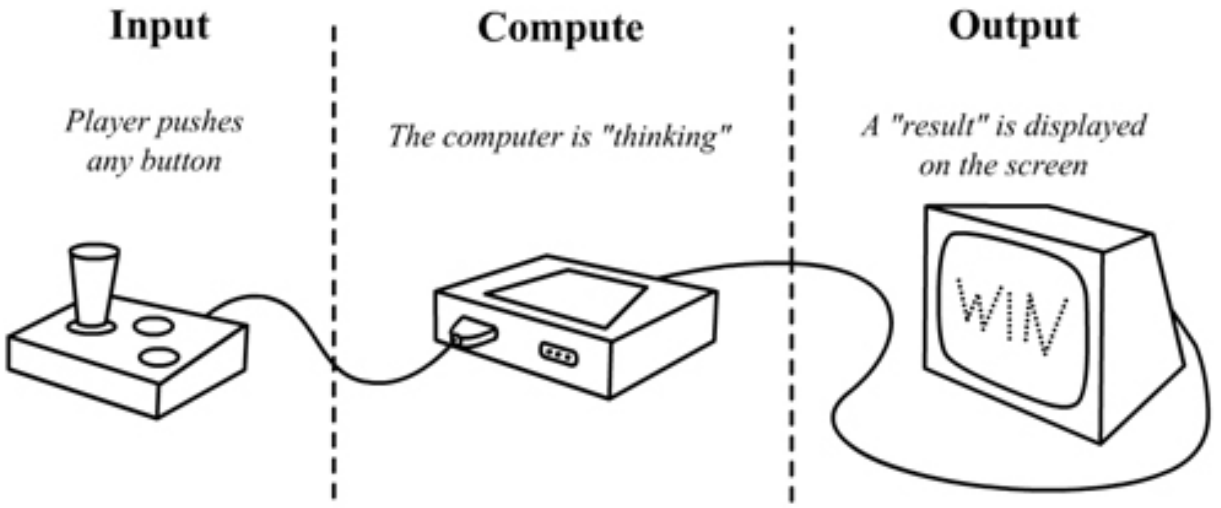


Benutzungs- & Administrationssicht auf ein Dateisystem am Beispiel Unix UFS

- Mounting
  - Zusammensetzen eines logischen Dateibaums aus mehreren Dateisystemen auf verschiedenen Disks / Partitionen.
  - Mount Point = Directory (muss nicht leer sein, aber bei Verwendung als Mount Point andere Einträge im Directory nicht mehr sichtbar).
  - Logische Wurzel bei Navigation im Baum erreicht, wenn „.“ = „..“ Eintrag.
  - mount, umount, /etc/mounttab, /etc/fstab



Einbindung und Verwaltung von Peripherie-Geräten in das Betriebssystem



- Zentrales Element: Gerätespezialdateien im /dev bzw. /devices Dateisystem als einheitliche Schnittstelle (oft ein eigenes Dateisystem)
- Major / Minor Device Number zur Identifikation
- Zugriffsrechte auf die Gerätespezialdateien sind relevant
- Geräte in verschiedenen Betriebs-Modi: block- oder zeichenweiser Zugriff
- Echte Geräte und „Pseudo-Geräte“ (z.B. virtuelle Terminals, Netzwerkprotokolle oder /dev/null)

Anforderungen von Peripheriegeräten an das Betriebssystem

- Ressourcenverwaltung:
  - Hauptspeicher für Zwischenpufferung von Daten beim Transfer
  - CPU-Zeit für das Behandeln asynchroner Events (z.B. Ankunft von Daten an der Netzwerkschnittstelle)
- Zugriffssteuerung und –synchronisation
  - Einheitliche Schnittstelle
  - Synchronisation von Zugriffen durch die Prozesse
  - Signalisierung
- Scheduling

Aufgaben und Funktionsweise des I/O-Subsystems am Beispiel Unix / Linux

- **Aufgabe**: schneller und zuverlässiger Datentransfer zwischen Geräten (Disk, Drucker, Tastatur, Bildschirm, Maus etc.) und Prozessen, d.h. Datenstrukturen im Prozess-Adressraum (read und write Operationen)

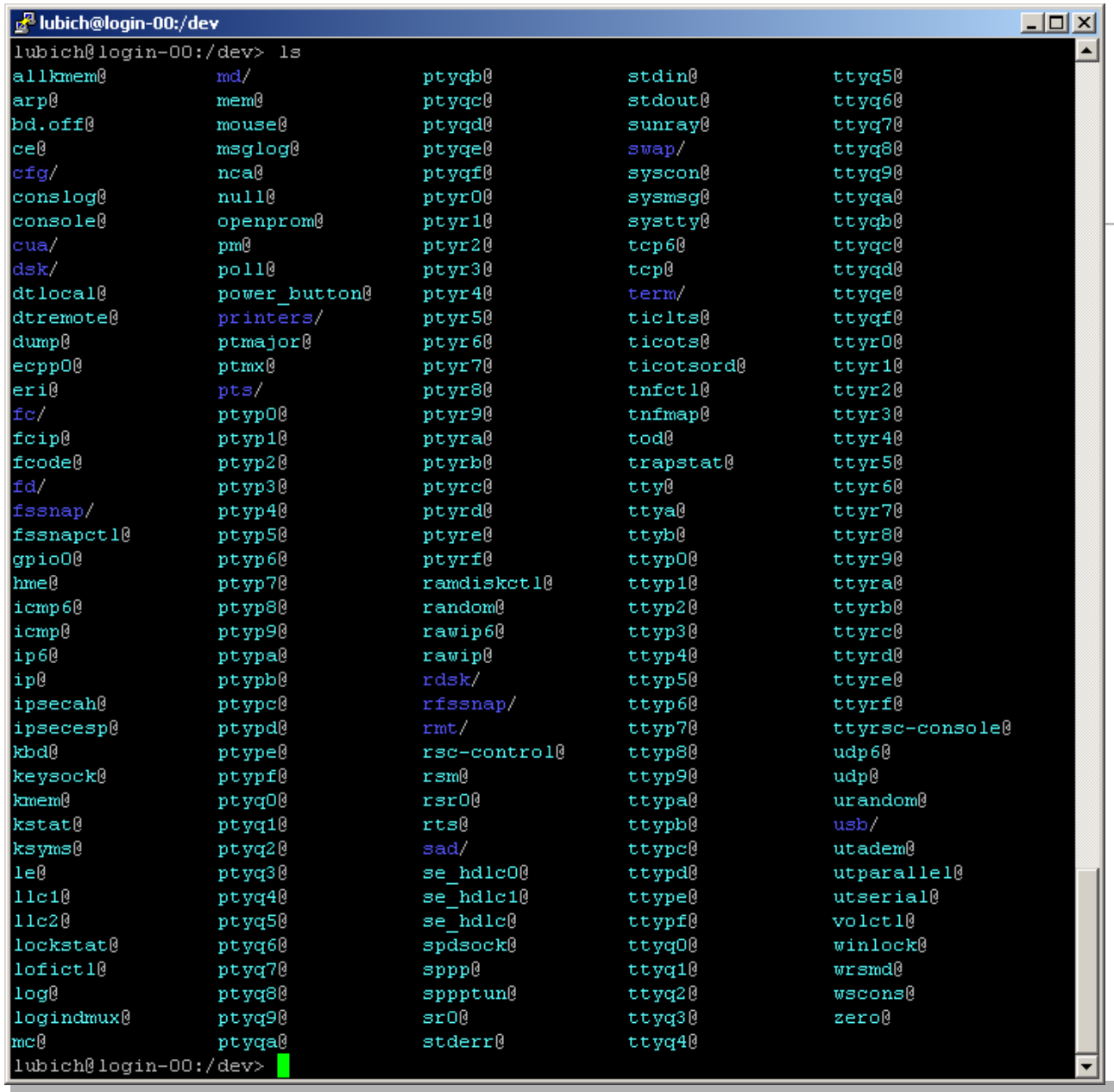
- **Design**: Das I/O Subsystem besteht aus einer oberen Schicht, die Daten zwischen dem Benutzer- und dem Kernel-Adressraum bewegt, und einer unteren Schicht, die Daten zwischen dem Kernel-Adressraum und den Geräten bewegt.

- Standardisiertes I/O (Geräteunabhängigkeit bezüglich Programmierung und Benutzung)
- Optimiertes I/O (abhängig vom Gerät und dessen Eigenschaften, Durchsatz, Sicherheit usw.)
- Konsistenz trotz Unterbrechbarkeit der Operationen und Zugriffssicherheit wenn Daten zwichen Kernel und Benutzer-Prozess bewegt werden
- Drei Typen von I/O: Datei-basiert, Zeichen-basiert, STREAM-basiert

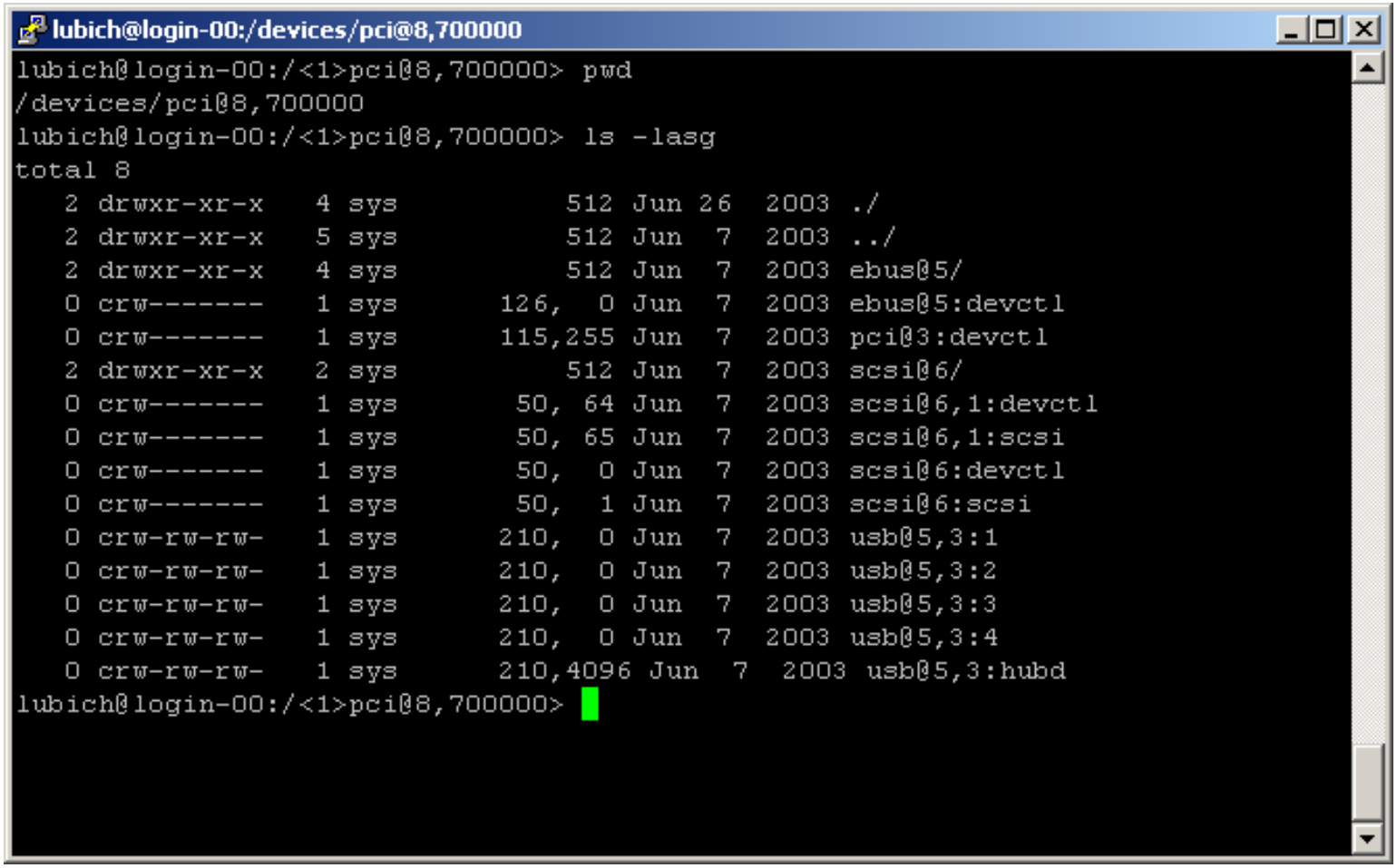
Der Buffer Cache

- **Ziel**: Optimierung des Zugriffs auf block-orientierte geräte, maximale Menge Daten im Speicher behalten.
- **Strategie 1**: vorausschauendes Lesen (read ahead)
  - Vorteil: beschleunigt das sequentielle Lesen (z.B. einer Datei)
  - Risiko: potentielle Verschwendung von Hauptspeicher
- **Strategie 2**: verzögertes Schreiben (delayed write)
  - Vorteil: Bündeln von Daten in Blöcke für das Schreiben auf langsame Geräte (z.B. Disk)
  - Risiko: nach einem erfolgreichen `write()` Systemaufruf sind die Daten noch nicht auf der Disk gespeichert (Verlustrisiko)
- Optimiert für die Arbeit mit “dummen” Peripherie-geräten

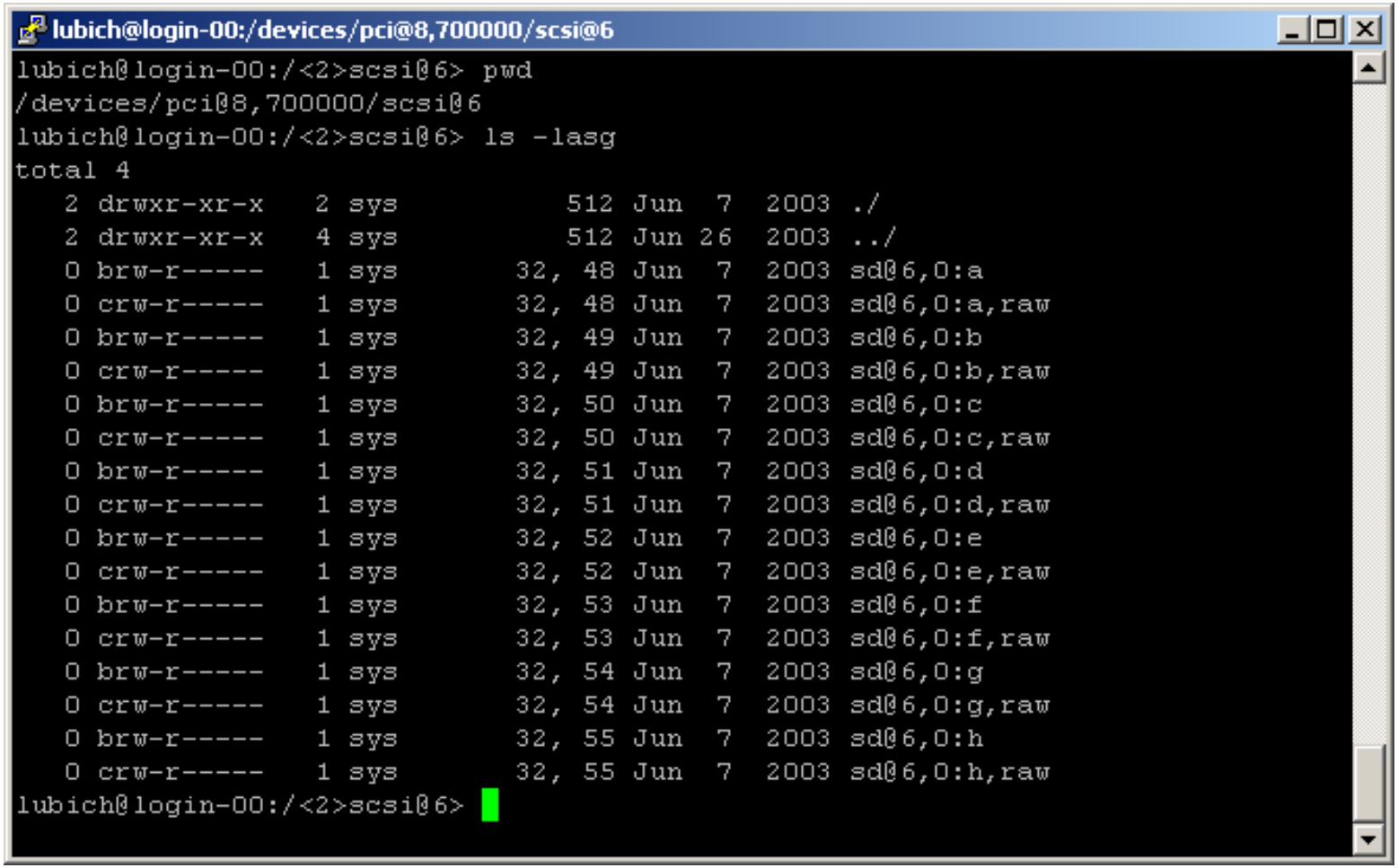
/dev Beispiel



/devices Beispiel



/devices Beispiel

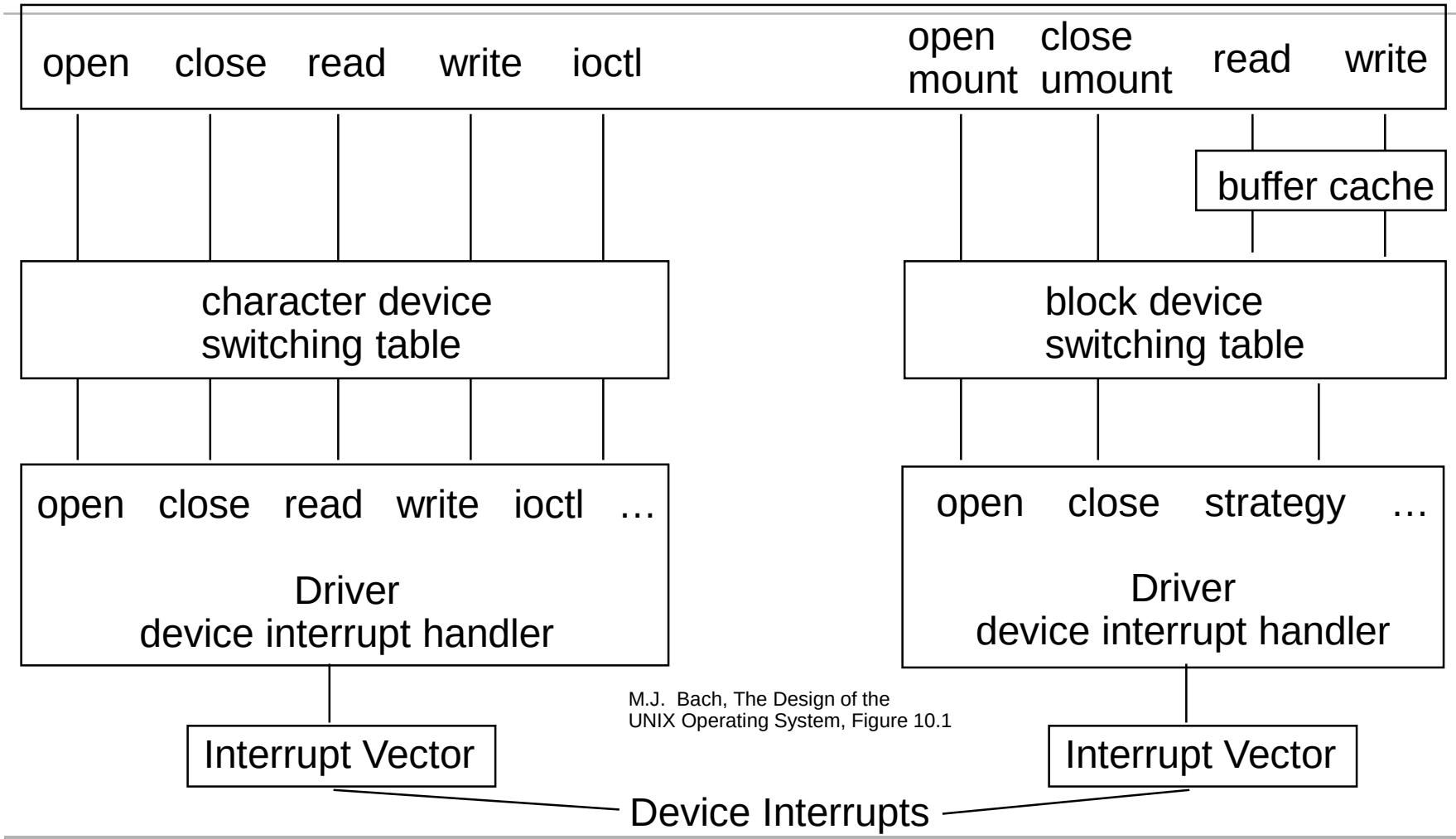


Unix Gerätetreiber

- Gerätetreiber sind die einzige Schnittstelle, über die ein Prozess mit Geräten kommunizieren kann. Sie sind Teil des Kernel-Codes des Systems, und werden entweder statisch beim Systemstart oder zur Laufzeit (in Linux: `insmod/rmmod`) geladen.

- In Unix sind Gerätetreiber Teil jedes Prozesses (über den Kernel-Code) – in anderen Betriebssystemen sind sie nur speziellen Kommunikationsprozessen zugänglich über die die anderen Prozesse dann mit Geräten kommunizieren müssen.

Zugang zu Geräten über Gerätespezialdateien



M.J. Bach, The Design of the UNIX Operating System, Figure 10.1

Gerätetreiber: Datenstrukturen

Block Device Switching Table

entry	open	close	strategy	print	…
0	gdopen	gdclose	gdstrategy		
1	gtopen	gtclose	gtstrategy		

Character Device Switching Table

entry	open	close	read	write	ioctl	…
0	conopen	conclose	conread	conwrite	conioctl	
1	syopen	*nulldev*	syread	sywrite	syioctl	
2	gdopen	gdclose	gdread	gdwrite	*nodev*	

M.J. Bach, The Design of the UNIX Operating System, Figure 10.2

```
mknod /dev/dsk/sc4d2s3 b 32 33
```

Gerätetreiber: udev

- **udev** steht für **userspace /dev** und ist ein Programm, mit welchem der Linux-Kernel Gerätedateien für die Datenein- und -ausgabe verwaltet.
- udev ersetzt seit dem Kernel 2.6 das früher genutzte devfs-Dateisystem, dessen Aufgaben es damit übernimmt. Genauso wie devfs verwaltet udev das /dev-Verzeichnis, welches die speziellen Gerätedateien enthält, um von Programmen aus auf die vom System zur Verfügung gestellten Geräte zuzugreifen.
- **udev** überwacht und wertet hotplug-Ereignisse aus. Finden sich dort Informationen über ein neu angeschlossenes Gerät, werden zu diesem Gerät vorhandene zusätzliche Informationen dem *sysfs*-Dateisystem entnommen und eine neue Gerätedatei im /dev-Verzeichnis erzeugt. Dabei ist der für die spezielle Datei verwendete Name und die Zugriffsberechtigung frei durch Regeln konfigurierbar (meist in /etc/udev oder für Ubuntu /lib/udev/rules.d)
- Siehe auch «`udevadm info –export-db`» und http://wiki.ubuntuusers.de/udev

Gerätetreiber: Interrupt Behandlung

- Routinen zur Interrupt-Behandlung sind sehr system- und hardware-spezifisch, es gibt nur wenige allgemeine Regeln für das Design.
- In Unix werden Interrupts immer im Kontext des gerade laufenden Prozesses behandelt, auch wenn der Prozess den Interrupt nicht verursacht hat oder nicht davon profitiert. Der gerade laufende Prozess muss also seine Arbeit unterbrechen, den Kontext sichern, den Interrupt behandeln, den Kontext restaurieren und kann dann weiterarbeiten.
- Die “Zeitstrafe” für das Behandeln von Interrupts verbleibt bei jedem Prozess (Annahme: etwa gleiche Verteilung über alle Prozesse → Fairness).

Zusammenfassung der Lektion 4 und Hausaufgabe

- Anforderungen an ein Dateisystem und schematischer Aufbau eines Dateisystems.
- Benutzungs- und Administrationssicht auf ein Dateisystem.
- Design-Kriterien für ein Dateisystem
- Das Ein-/Ausgabebsystem

- Hausaufgabe:
  - Repetieren Sie den Stoff dieser Lektion.
  - Studieren Sie die Web-Seite: https://de.wikipedia.org/wiki/Dateisystem
  - Experimentieren Sie mit Zugriffsrechten, Links und den Verwaltungs-Kommandos