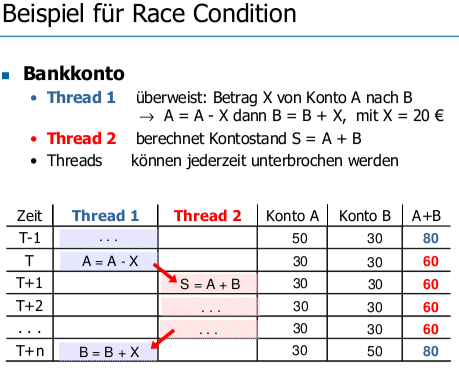
**ZUF 2 BSY**

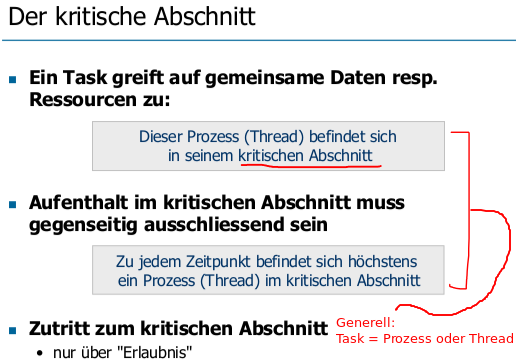
*René Bernhardsgrütter, 01.05.2013*

Betirebssystem / Computer



Race Condition, wenn…

* …Prozesse und /oder Treads gemeinsam Daten lesen und schreiben.
* …das Resultat von der Ausführungsreihenfolge der betiligeten Prozesse resp. Threads abhängt.



Lösung:

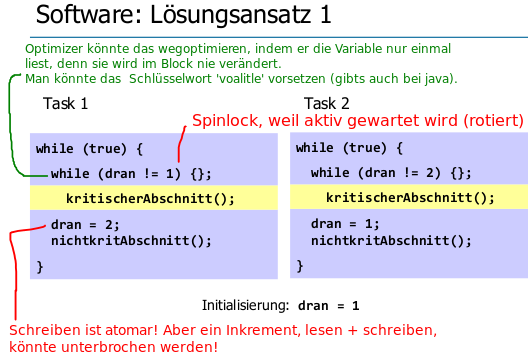
Entry Section: Sicherstellen, dass nur einer arbeitet

Working section: Einer arbeitet

Exit Section: Wieder allen freigeben

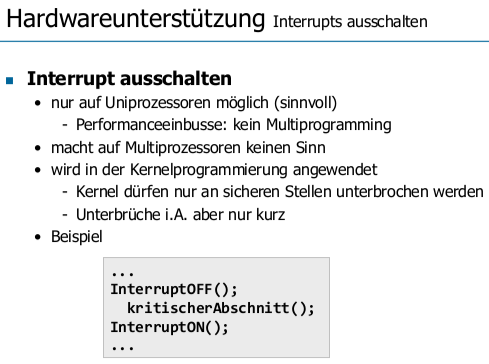
Anforderungen des Kritischen Abschnitts (=KA):

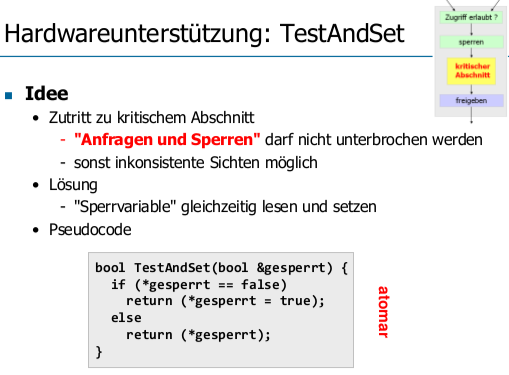
1. Nur ein Prozess gleichzeitig
2. Muss unabhängig von Prozessgeschwindigkeiten oder Anz. CPUs sein
3. Prozesse ausserhalb des KA dürfen andere nicht beeinflussen (sonst müssten sie in KA)
4. Prozesse, die in KA wollen, dürfen nicht verhungern können
5. Wenn KA frei: Ein anfragender Prozess muss umgehend rein dürfen
6. KA darf nur endlich lange von Prozess besetzt werden

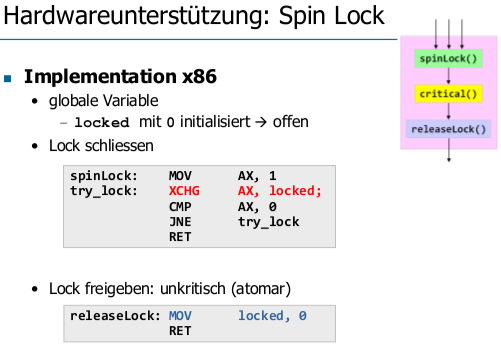


Problem: Busy Wait!

Heisst auch **Peterson’s Algorithm**









Der Semaphor ist eigentlich eine Datenstruktur

typedef struct {

int count; //Wert des Semaphors

ProcList queue; //Liste wartender Prozesse

} semaphor;

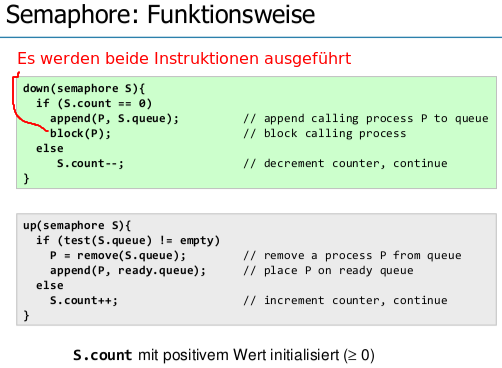
semaphor S;

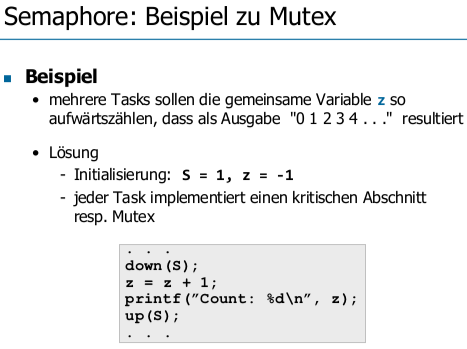
**Dijktras** Bezeichnungen für Semaphoroprationen

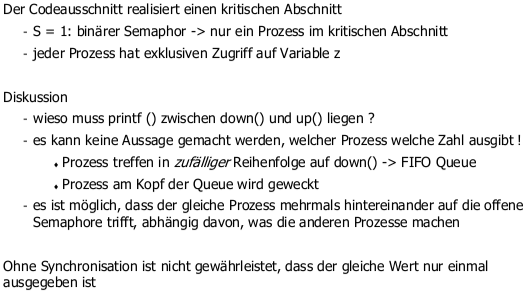
* P(S) proberen -> testen
* V(S) verhogen -> erhöhen (freigeben)

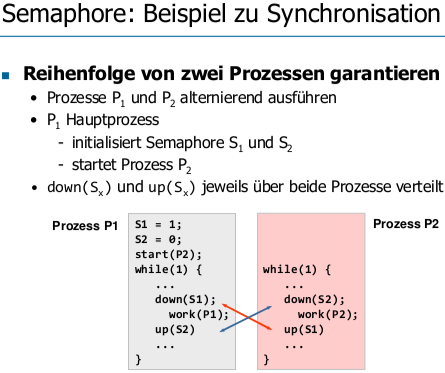
Oft werden auch folgende Bezeichnungen verwendet

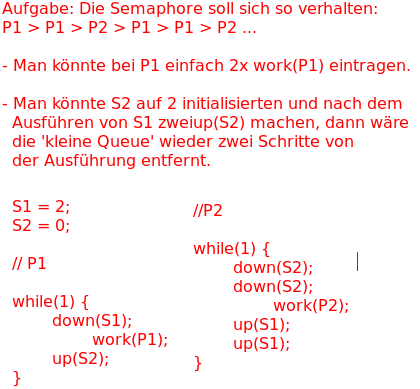
* semWait(S) (down)
* semSignal(S) (up)



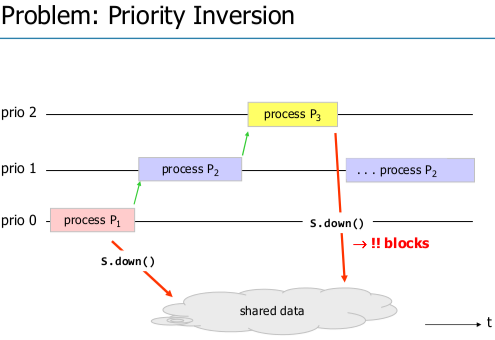


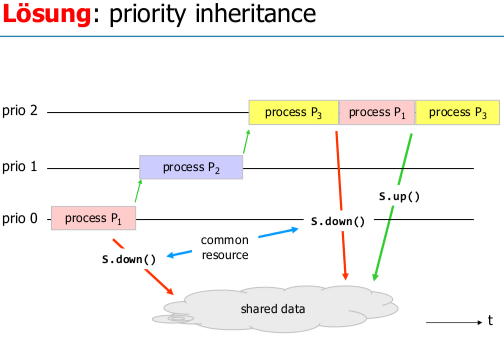




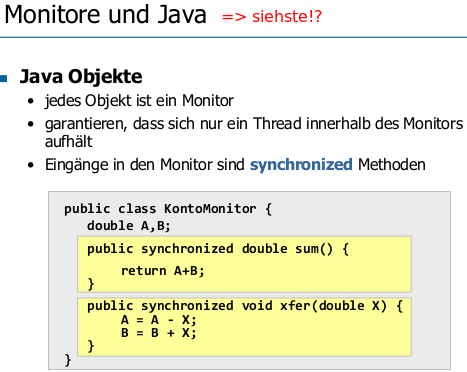


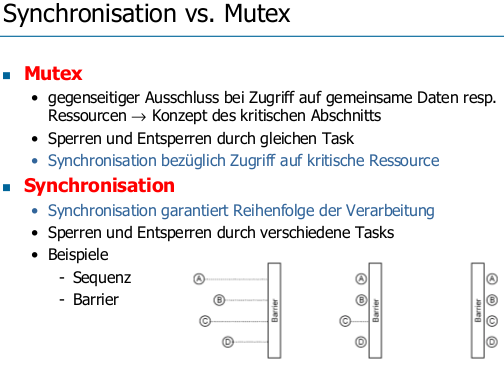
Probleme mit Semaphoren: Kompliziert => Fehleranfällig, Anwendung muss in allen Prozessen korrekt sein, Böswilliger Code kann die Zusammenarbeit schädigen.

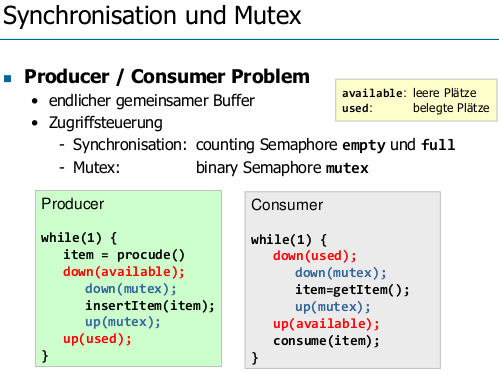






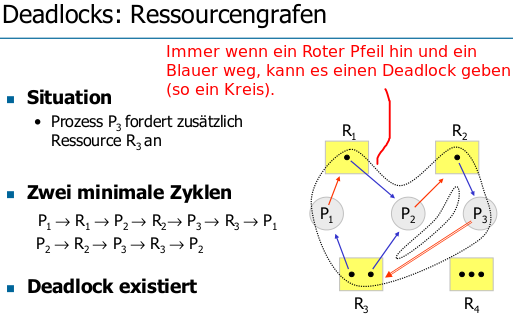






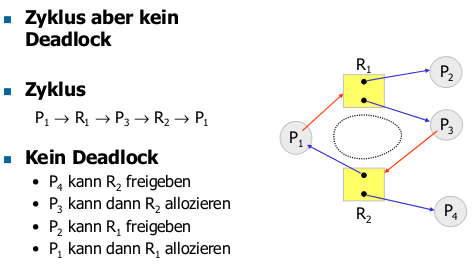
Deadlocks

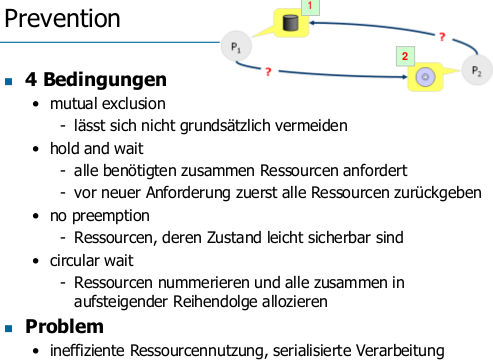
1. **Mutual exclusion**; mindestens eine Ressource ist exklusiv reserviert
2. **Hold and wait**: mindestens ein Task hat eine Ressource exklusiv reserviert und wartet auf weitere Ressourcen
3. **No preemption**: reservierte Ressourcen können dem Task nicht entzogen werden (freiwillige Rückgabe nur, wenn Aufgabe gelöst)
4. **Circular wait**: geschlossene „Kette “ von Tasks existiert, in der jeder Prozess mindestens eine Ressource reserviert hat, die auch von einem Nachfolger in der Kette benötigt wird



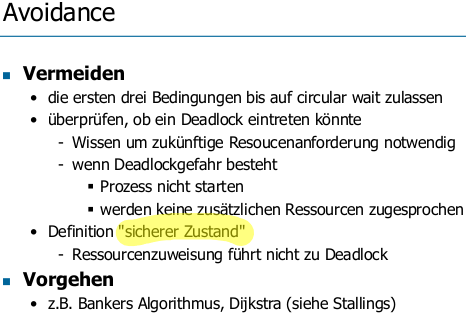
Deadlock existiert:

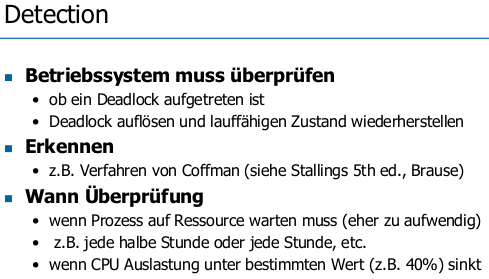
* Prozess P 2 wartet auf Ressource R2 , die von Prozess P3 gehalten wird
* Prozesse P3 wartet auf R3, die von den Prozessen P1 und P2 gehalten wird
* Prozess P wartet auf Ressource R , die von Prozess P gehalten wird

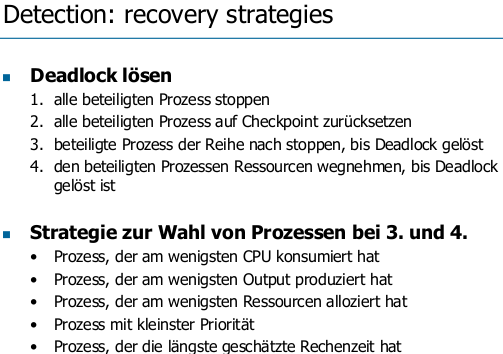


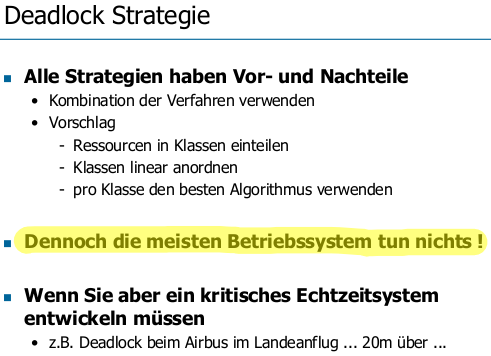


=> Unwichtig

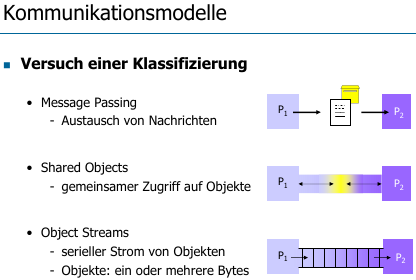


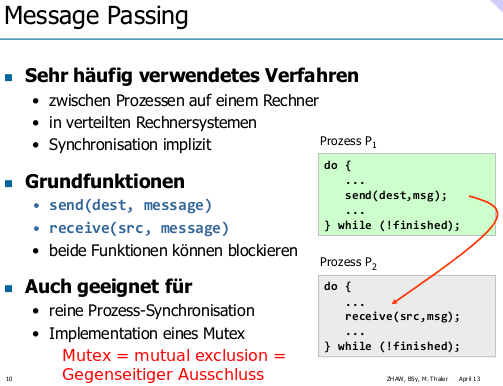


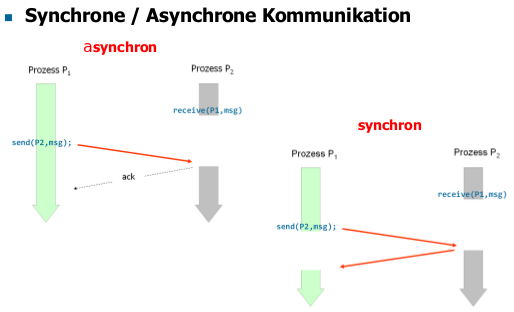


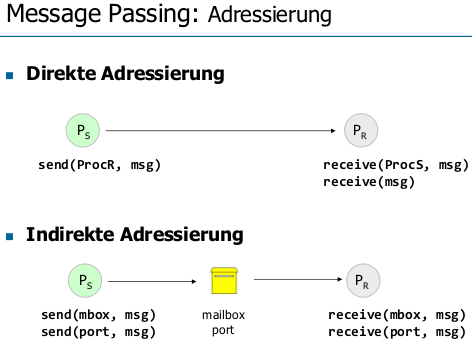


Interprocess Communication

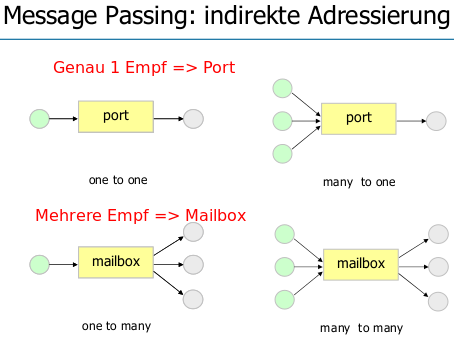


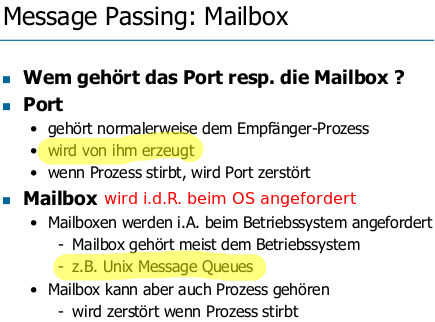


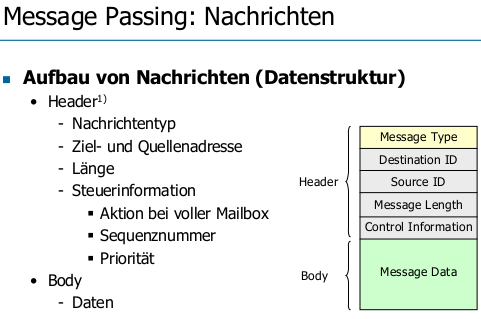


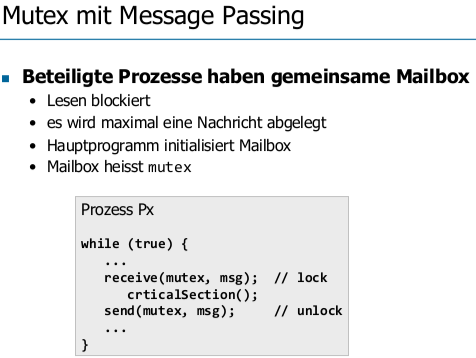


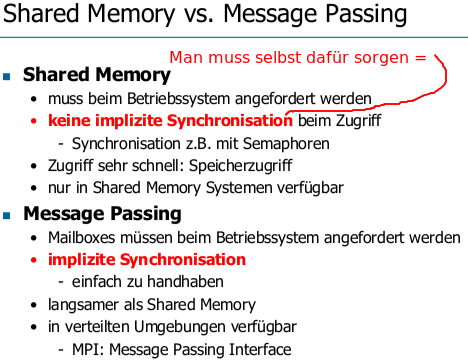
Mailbox, Port: Im Wesentlichen eine Queue.

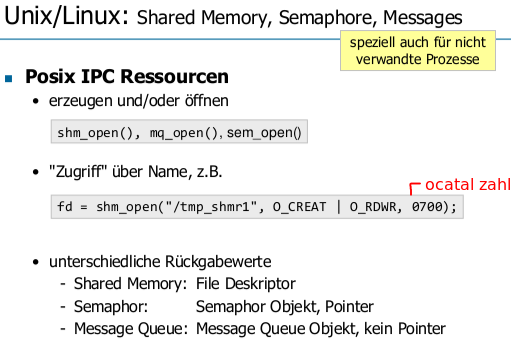


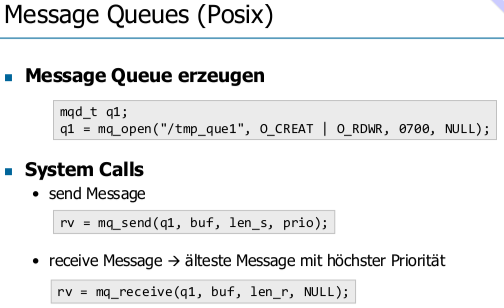


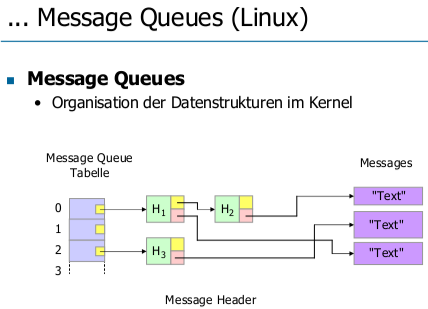


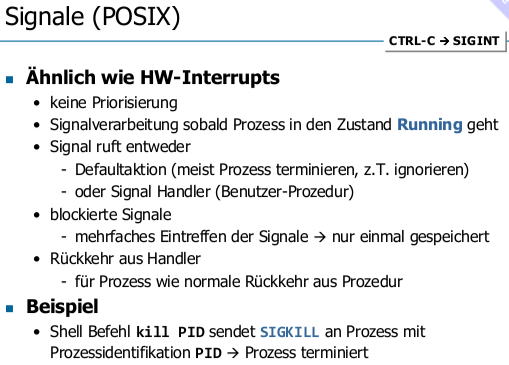


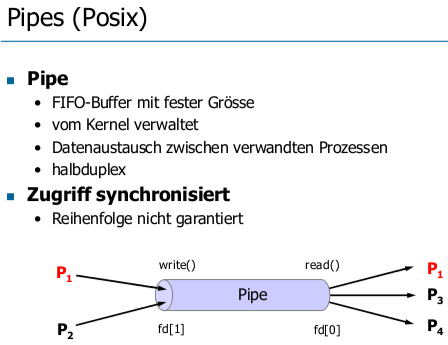


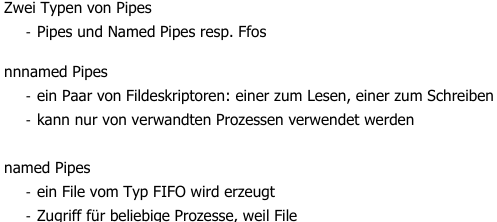


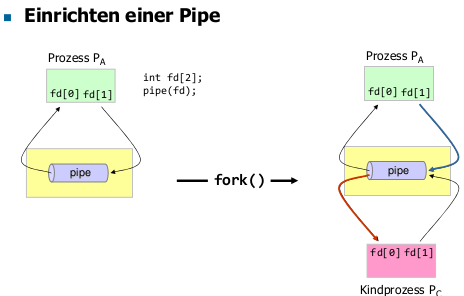


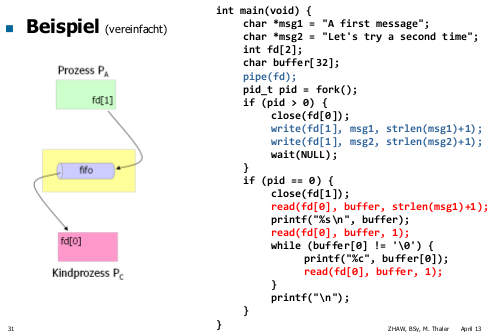


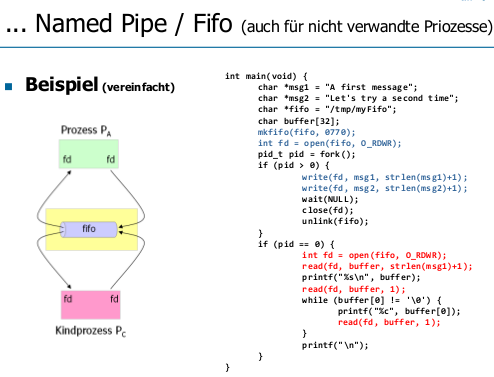


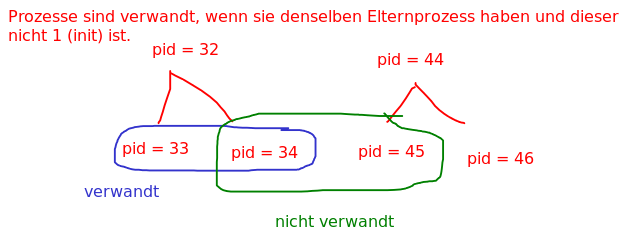


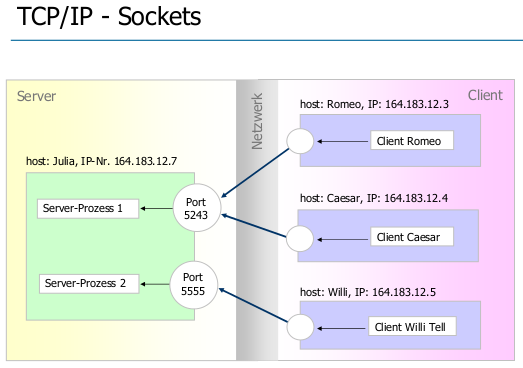












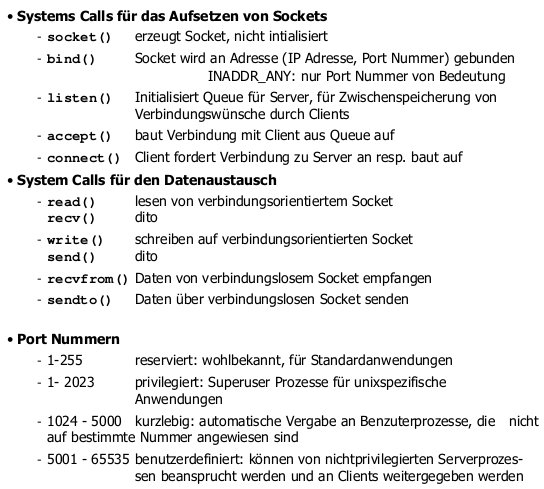
Typen:

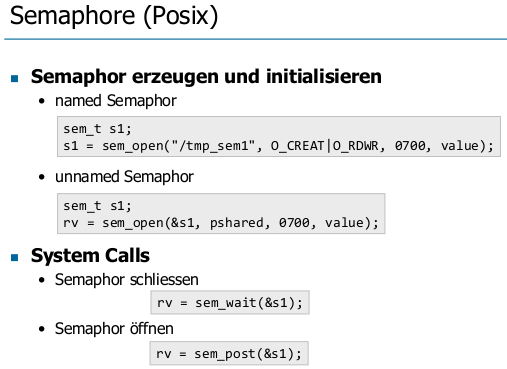
* TCP/IP Sockets: Netzwerk
* Unix Domain Sockets: Lokal

- vor allem Client-Server Kommunikation

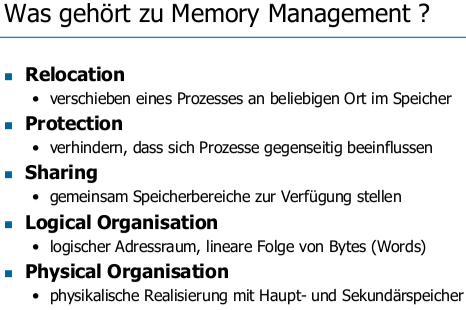
- vielfältig und flexibel (dafür auch komplizierter)

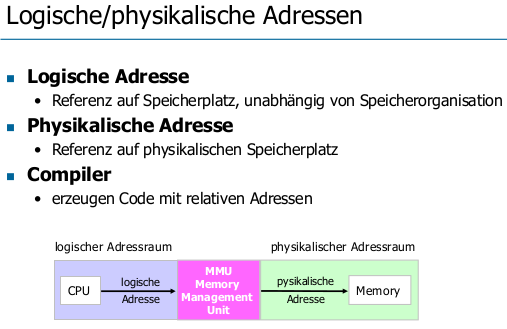
- verbindungsorientierte und -lose Kommunikation

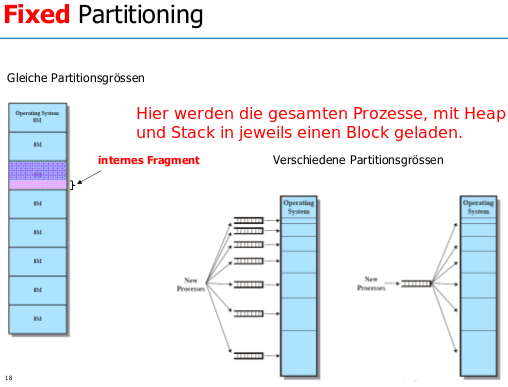




Memory Management

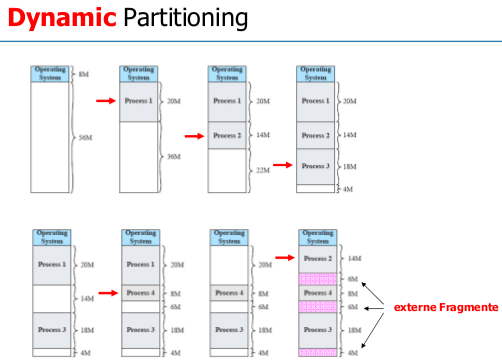






Gleiche Partitionen

* jedes Programm, egal wie gross, belegt eine Partition
* Partiotion nicht vollständig gefüllt: wird internal fragmentation genannt
* Nutzung des Hauptspeichers ineffizient



Grösse und Anzahl der Partitionen variabel

* jedem Prozess soviel Speicher zuweisen, wie er benötigt

Problem: **externe Fragmentatierung**

* im Hauptspeicher bilden sich mit der Zeit Löcher
* Grund: Prozesse werden ausgelagert und können nicht immer durch gleich grosse Prozesse ersetzt werden
* **compaction** Algorithmus notwendig, um Prozesse zu verschieben



first-fit

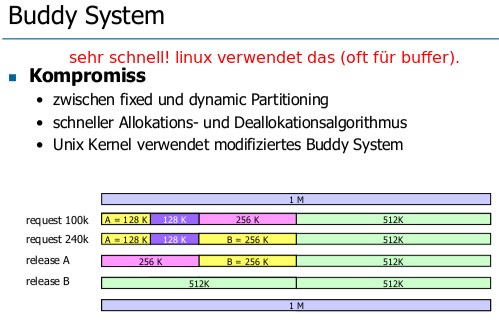
* einfachster Algorithmus, i.A. auch am schnellsten und besten
* alloziert Blöck eher in Nähe des Speicheranfangs
* Tendenz: weniger Fragmentatierung als bei next-fit

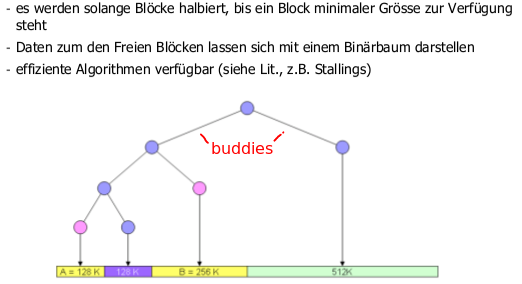
next-fit

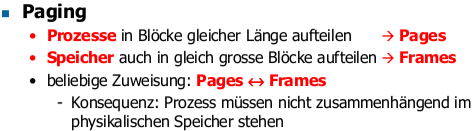
* alloziert oft freien Block am Schluss des Speichers
* Blöcke am Ende des Speichers oft am grössten
* Tendenz: compaction öfter notwendig als bei first-fit

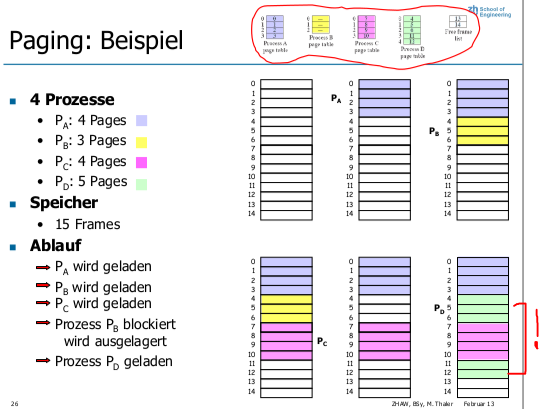
best-fit

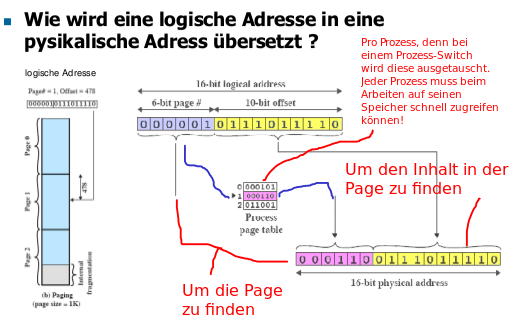
* sucht kleinsten Block passenden Block
* minimiert externes Fragmente zu nächsten Block
* Tendenz: es entstehen schnell viele kleine Fragmente
* i.A. schlechtester Algorithmus: compaction muss öfters durchgeführt werden als bei bei first-fit und next-fit

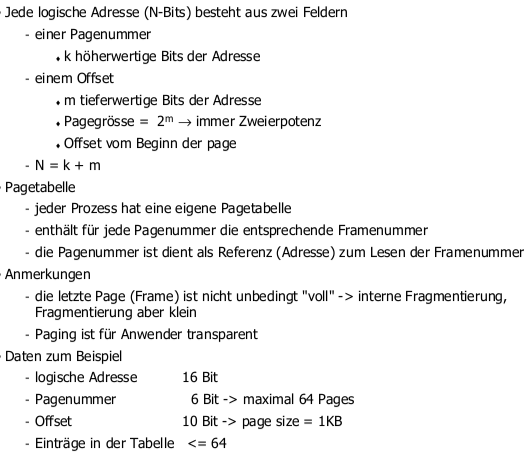


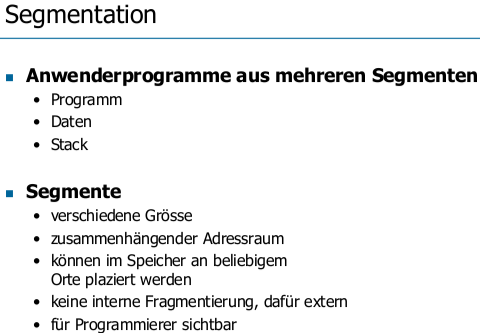


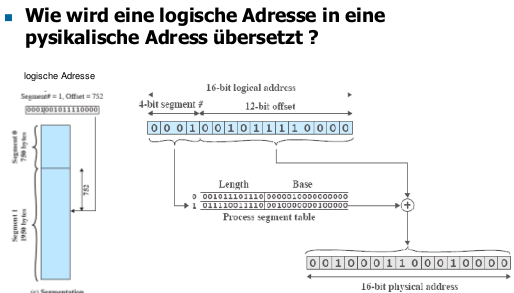


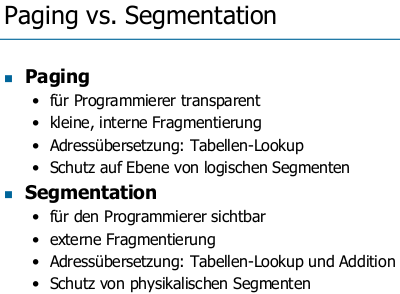




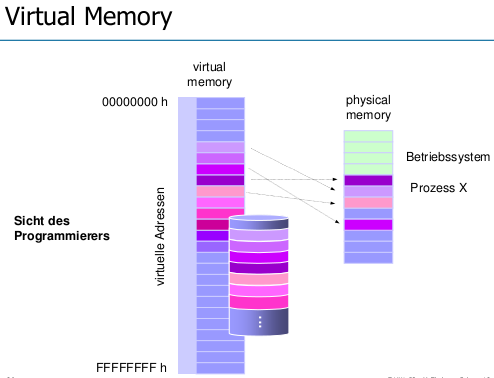


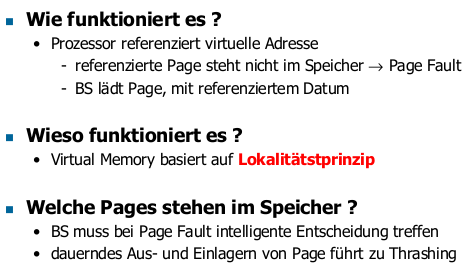


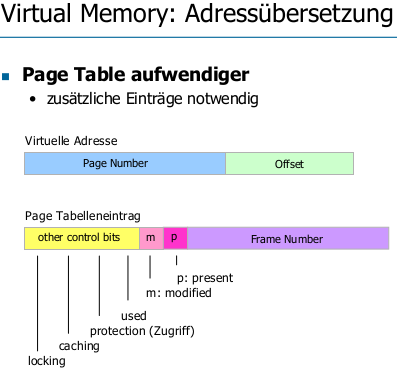


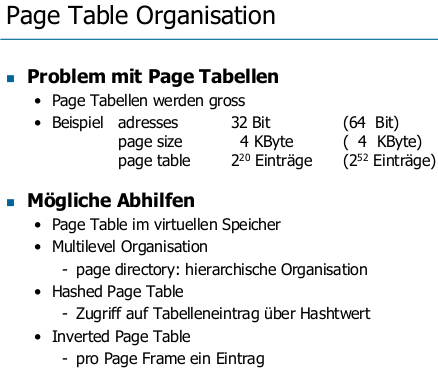


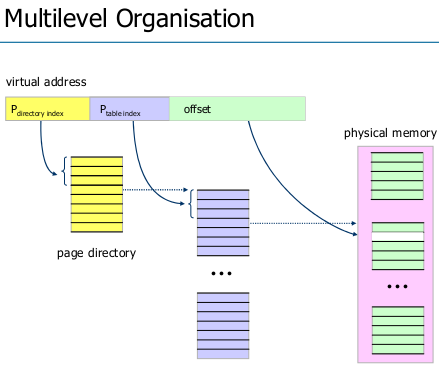
Virtual Memory

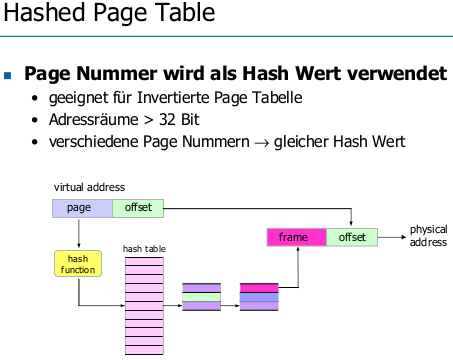


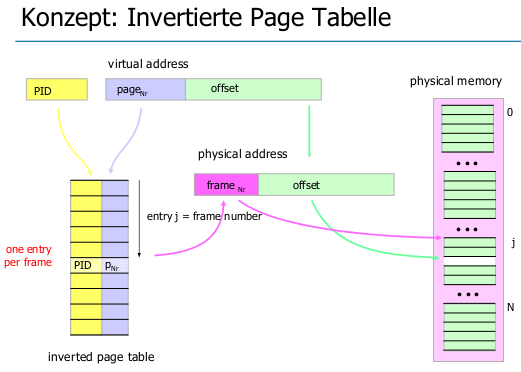


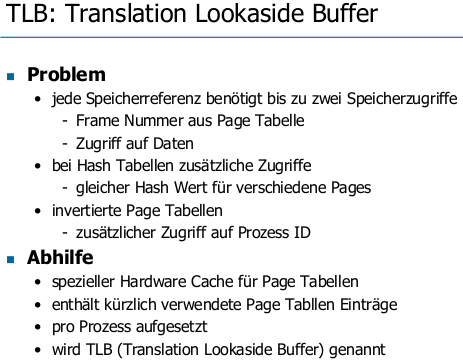


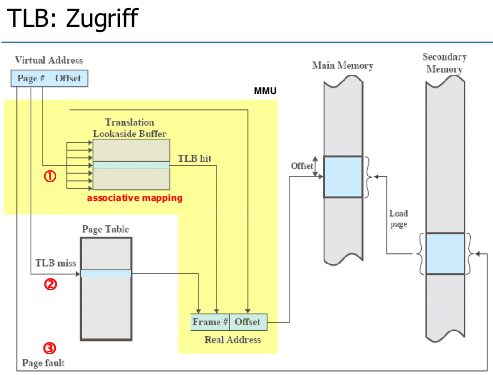


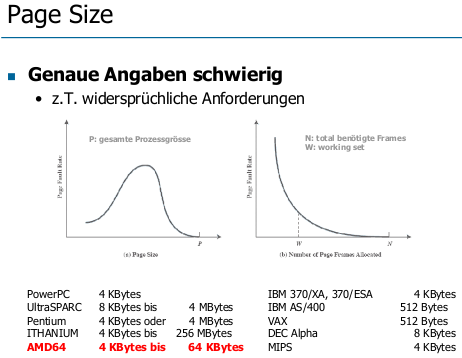




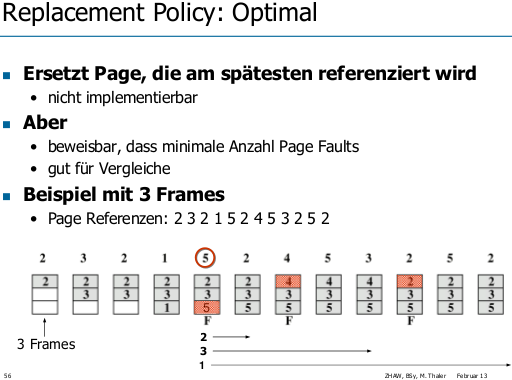


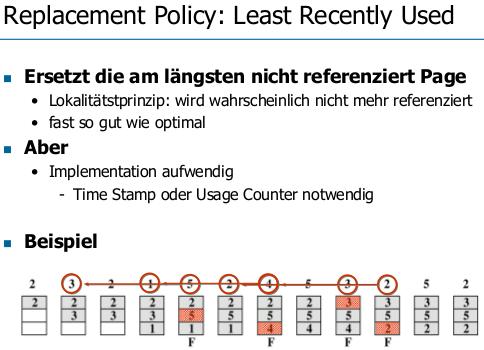


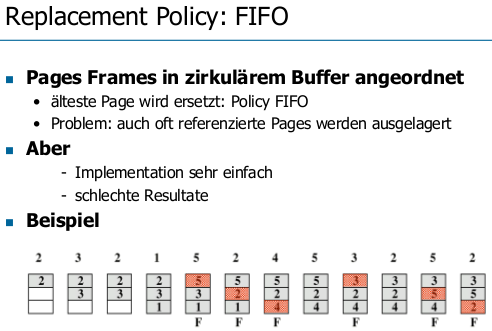


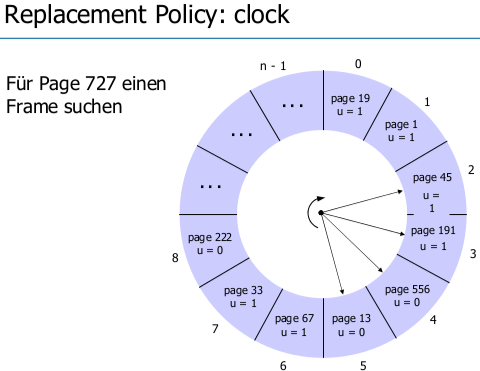


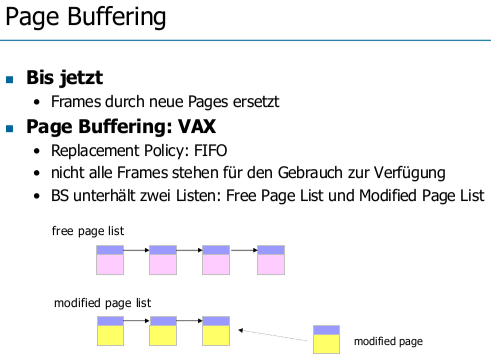
Memory Management Software

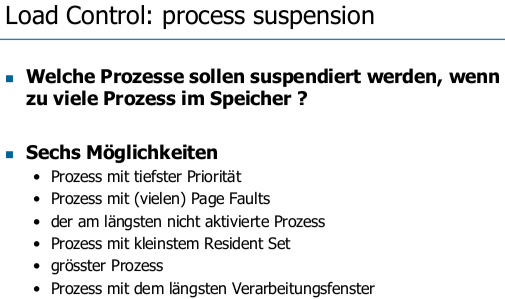


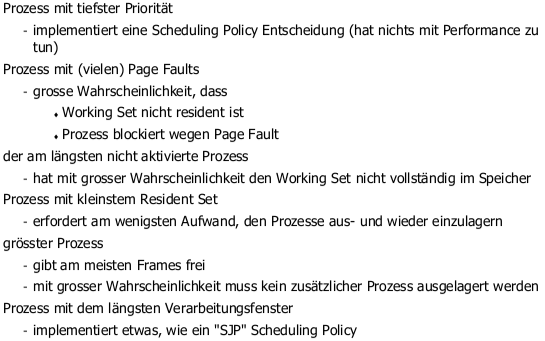


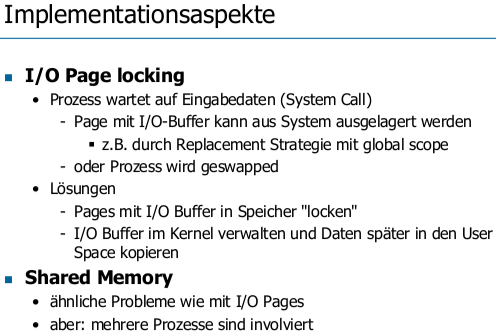


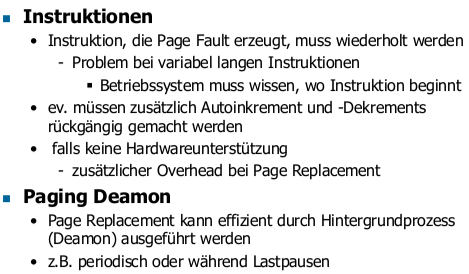


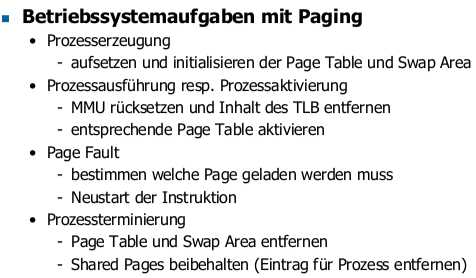




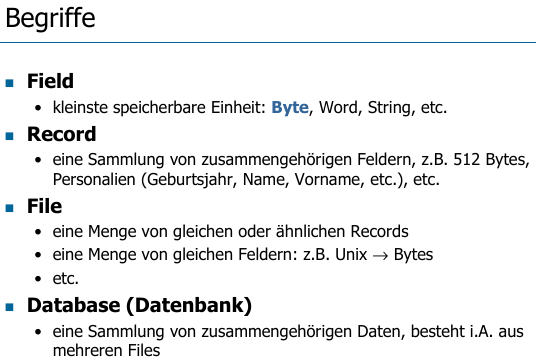


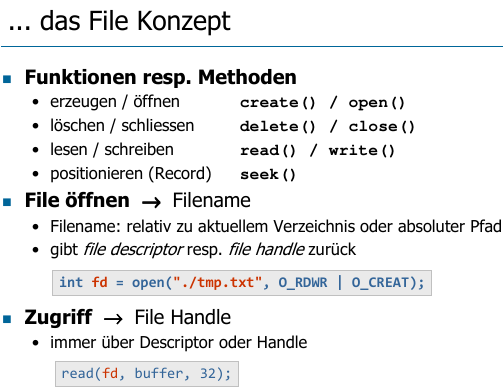


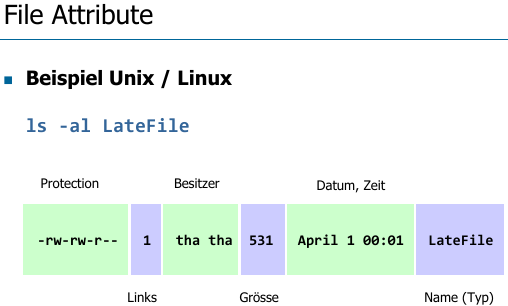


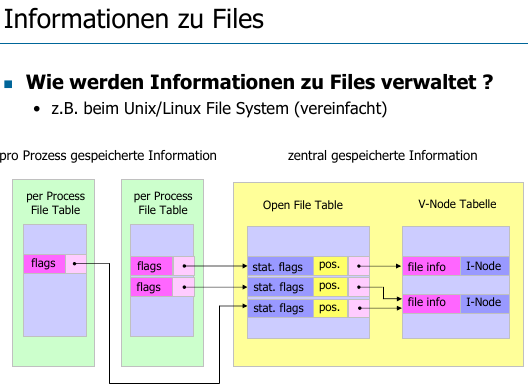


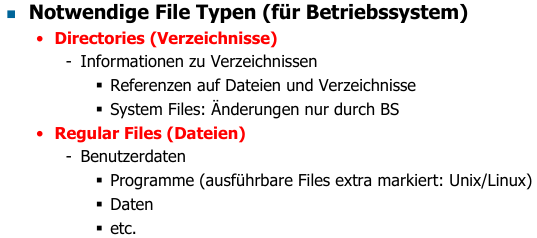
FileSystems

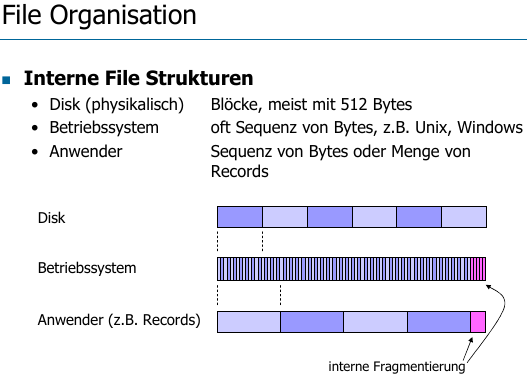


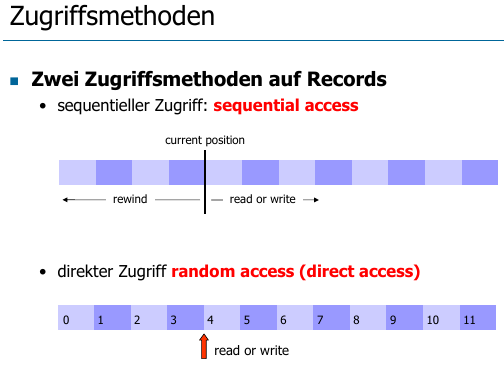




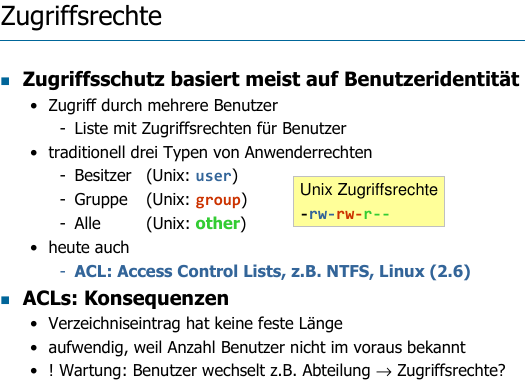


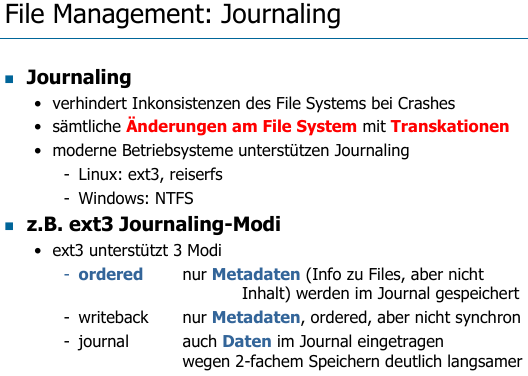


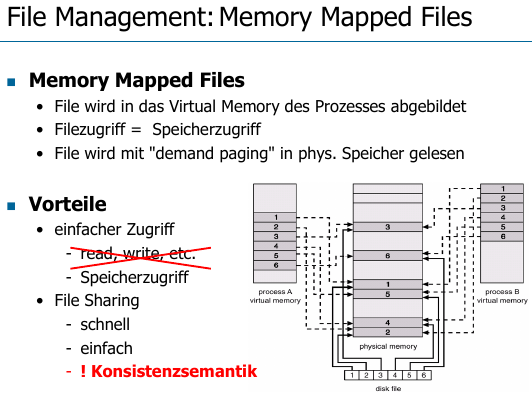


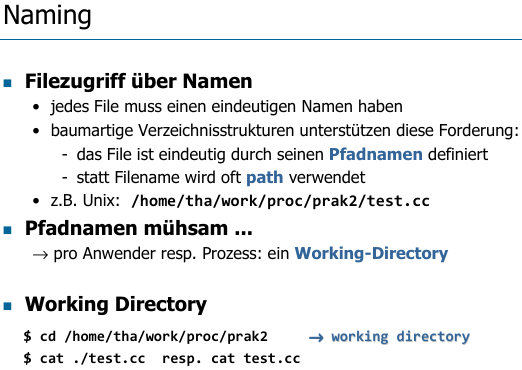


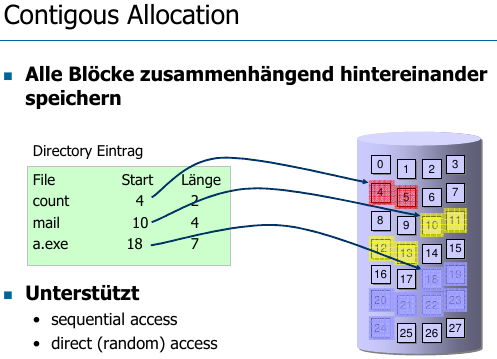
Record = Byte oder eine beliebige Struktur.

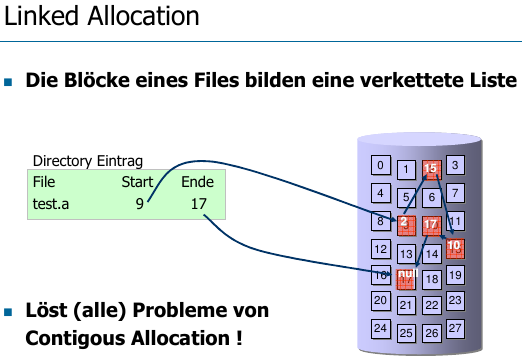




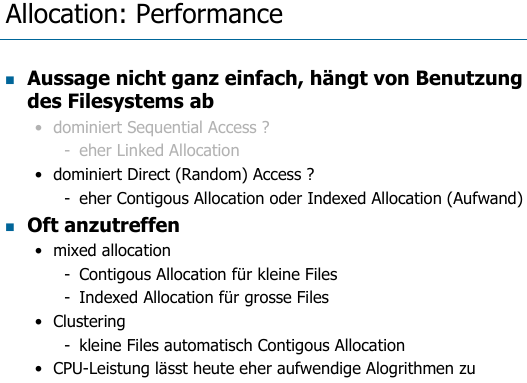


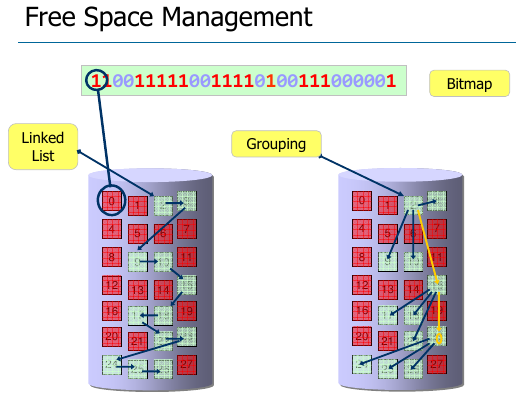


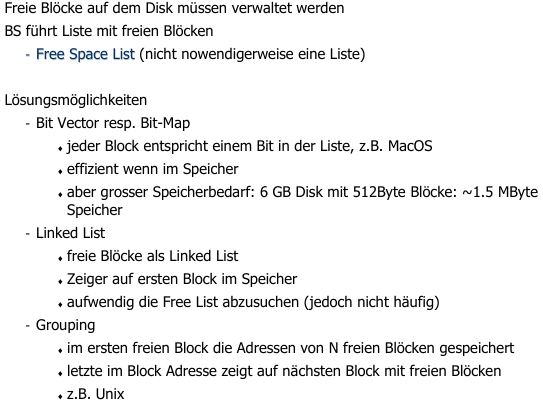


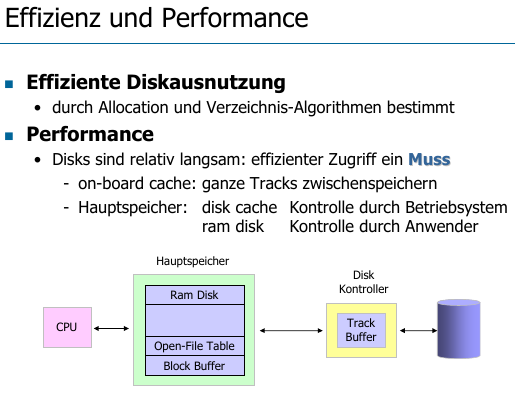


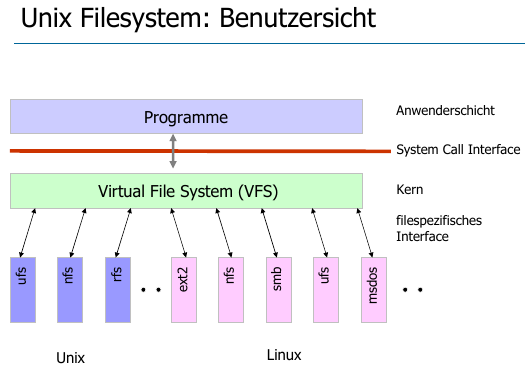


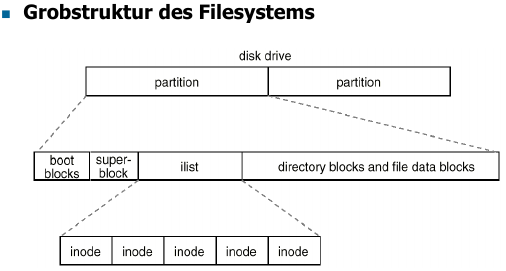




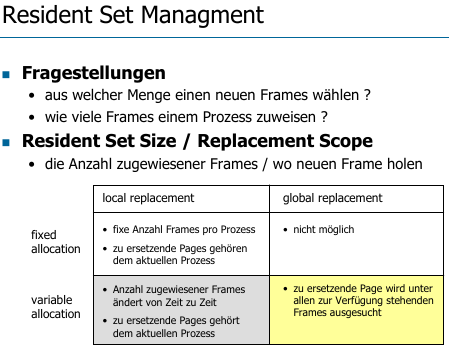


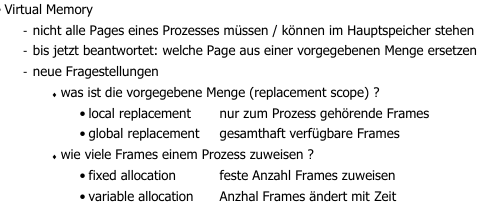






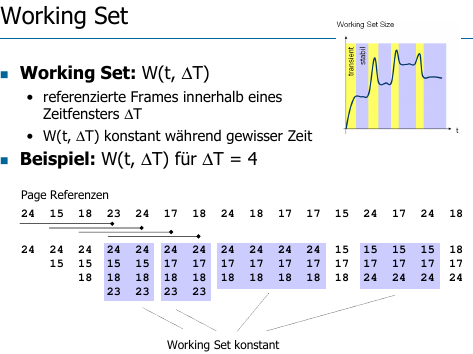
FileSysteme Slides bis Folie 50/65.





Linux: Variable Allocation / Global Scope

Winows 2k: Variable Allocation / Local Scope

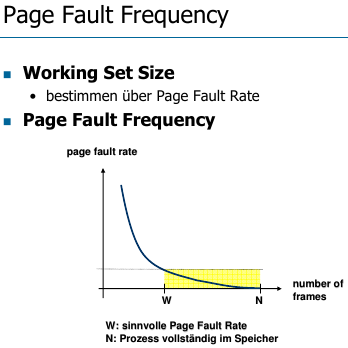


**Resident Set**: Anzahl aktuell zugewiesener Frames.

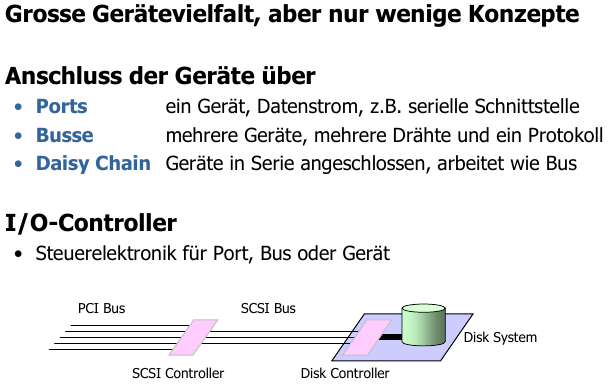
**Working Set**: In während DeltaT zugewiesene Frames.

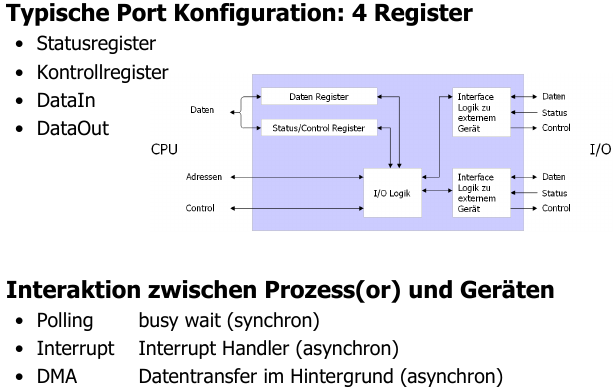
Wenn voll: Das am längste nicht mehr gebrauchte raus.

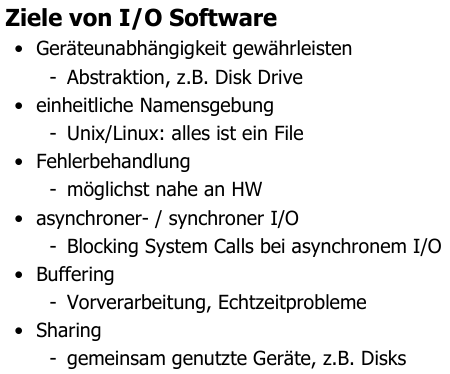
Wichtig: Wenn DeltaT = 2, dann muss eines, dass zweimal nicht mehr gebraucht wurde, entfernt werden, auch wenn der Speicherplatz für nichts sonst gebraucht wird! **Das Hinzufügen ist Nummer 1**!

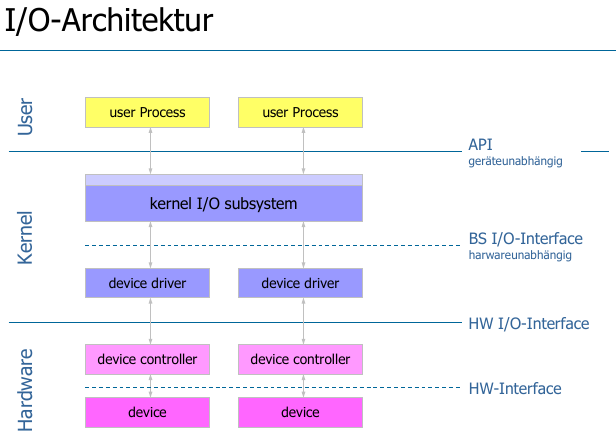


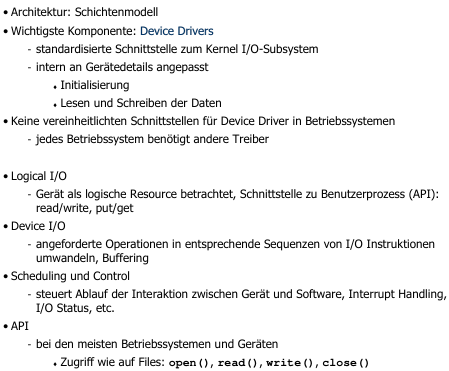
IO

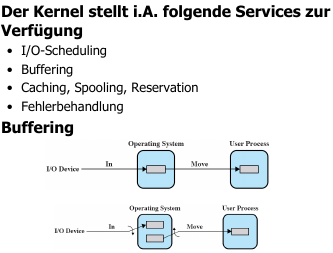


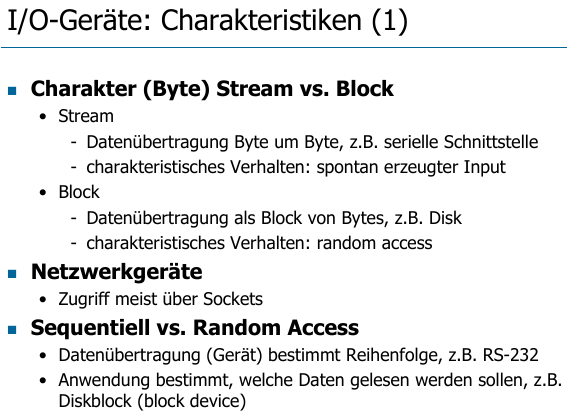


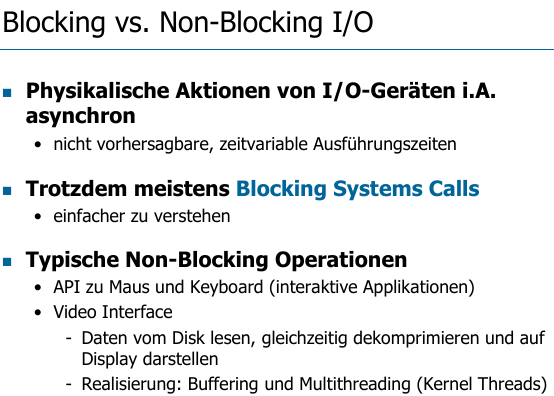




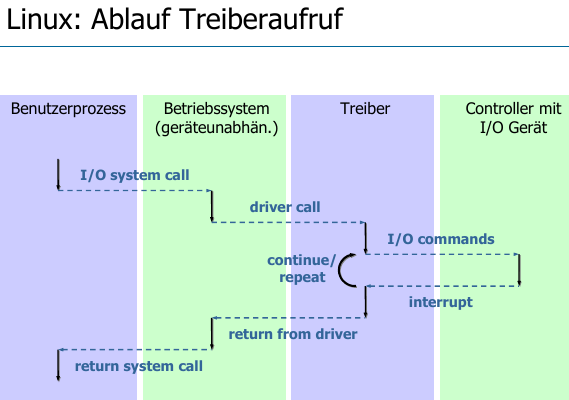


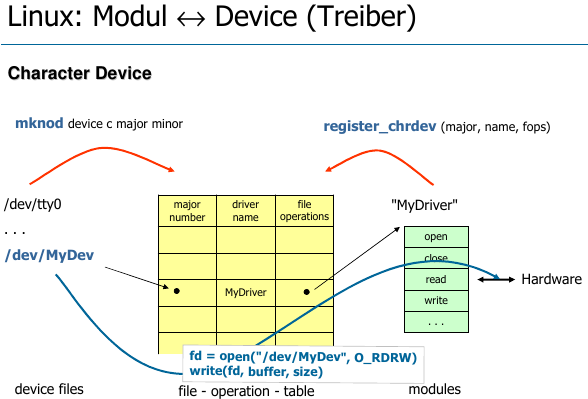




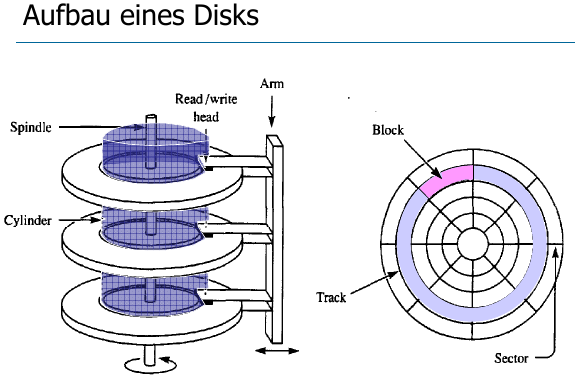


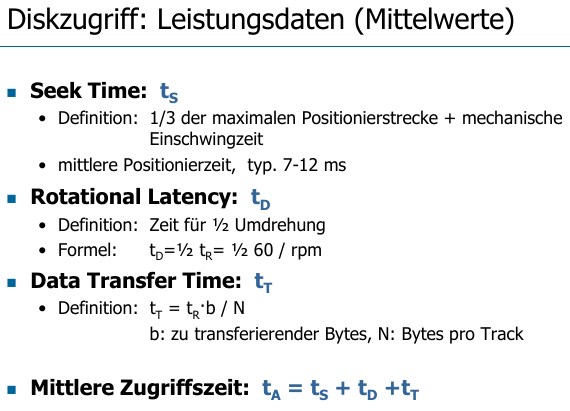
IO: Treiber

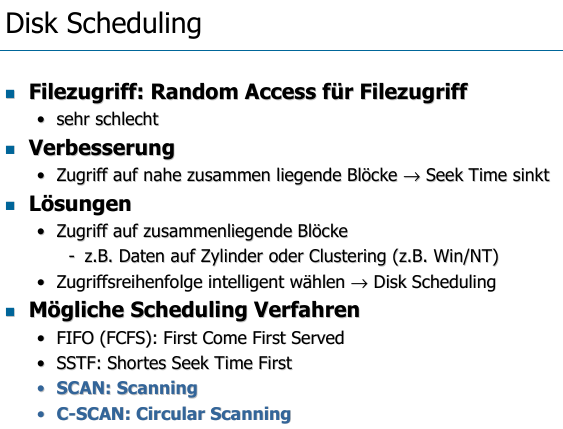


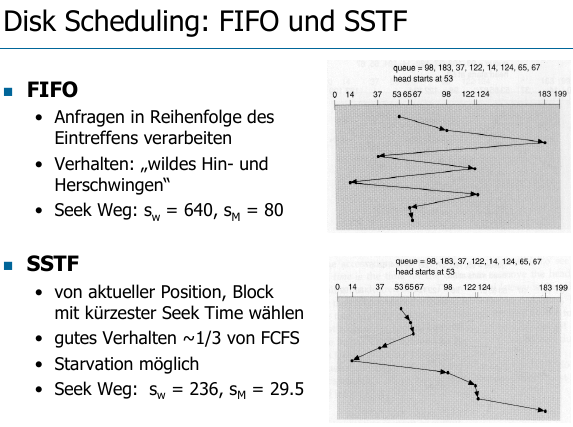


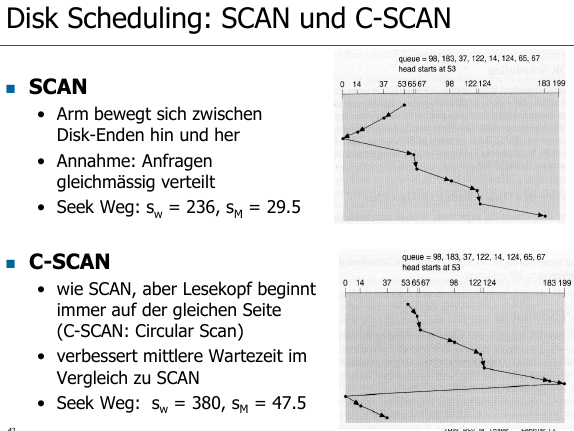
IO: Disk

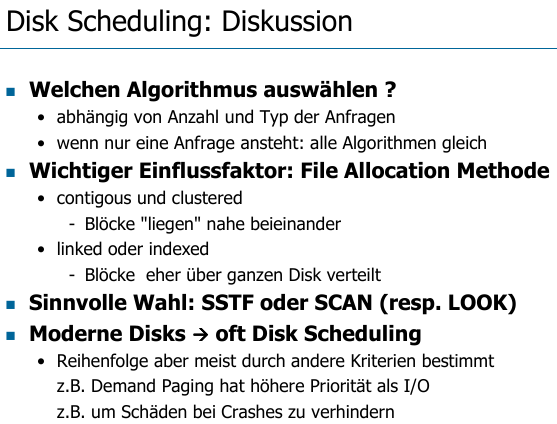


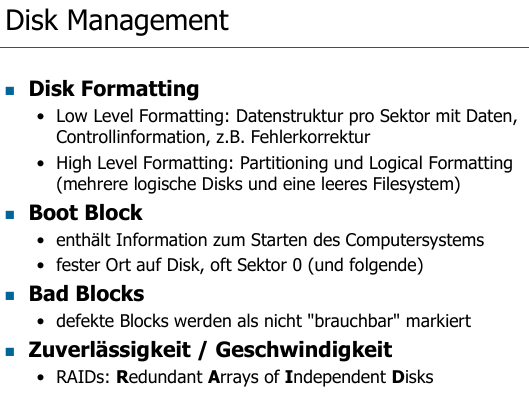


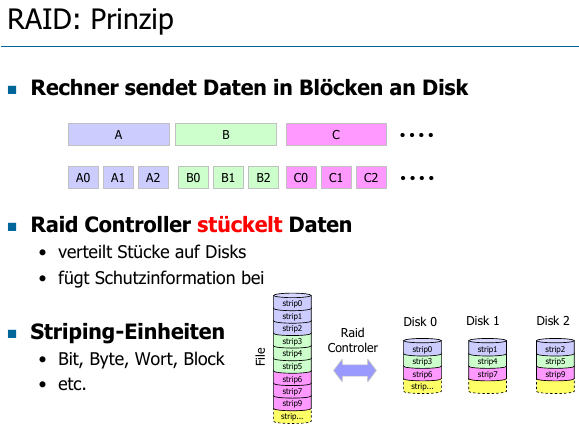












Raid-Level:



