BETRIEBSSYSTEME

Aufgabe eines Betriebssystems:

Die Aufgabe eines Betriebssystems ist es, eine Umgebung zu schaffen, in welcher verschiedene Anwendungsprogramme ablaufen können.

Begriff Betriebssystem:

Unter einem ***Betriebssystem*** (engl. operating system) versteht man Software, die zusammen mit den Hardwareeigenschaften des Computers die Basis zum Betrieb bildet und insbesondere die Abarbeitung von Programmen steuert und überwacht.

Was ist ein Betriebssystem?

Virtuelle Maschine
Betriebsmittelverwalter

Woraus besteht ein Prozeß?

- einem ausführbaren Programm,
- den Programmdaten,
- dem Kellerinhalt (Stack), (= ein Platz im Hauptspeicher)
- dem Programmzähler,
- dem Kellerzeiger (Stackpointer),
- Registerinhalten, (= sehr schneller Speicher auf dem Prozessor)
- und weiteren Informationen, die zur Programmausführung benötigt werden.

Monolithischer- vs. Microkernel

Ein **monolithischer Kernel** bezeichnet einen Betriebssystemkern, in den nicht nur Funktionen zu Speicher- und Prozessverwaltung und zur Kommunikation zwischen den Prozessen, sondern auch Treiber für die Hardwarekomponenten direkt eingebaut sind.

Ein **Mikrokernel** (oder auch Mikrokern) bezeichnet einen vor allem bei Echtzeitbetriebssystemen verwendeten Betriebssystemkern, der weniger Funktionen als ein normaler Monolithischer Kernel enthält. Hier finden sich in der Regel lediglich Funktionen zur Prozessverwaltung und Grundfunktionen zur Synchronisation und Kommunikation.

Was ist ein Prozeß?

Ein Prozeß ist eine Aktivität, die aus Programm, Eingaben, Ausgaben und einem Zustand (zeitlicher Ablauf) besteht.

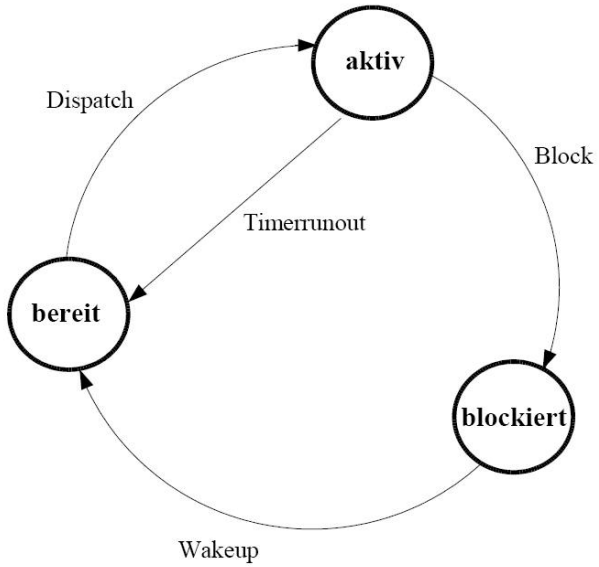
Was ist ein Programm?

Ein Programm ist ein Algorithmus in einer geeigneten Notation.

Prozeßzustände:

- **aktiv/rechnend** Dem Prozeß wird ein Prozessor zugeteilt.
- **rechenbereit** Der Prozeß ist ausführbar, aber der Prozessor ist einem anderen Prozeß zugeteilt.
- **blockiert** Der Prozeß kann solange nicht ausgeführt werden, bis ein externes Ereignis eintritt.

Grafische Darstellung:



Dispatch: bereit → aktiv
Zuteilung der CPU an einen Prozeß.

Timerrunout: aktiv → bereit
Nach Ablauf einer Zeitscheibe wird der Prozeß die CPU wieder entzogen.

Block: aktiv → blockiert
Aktiver Prozeß hat eine E/A-Operation angefordert (oder sich selbst für eine bestimmte Zeit verdrängt), bevor seine Zeitscheibe abgelaufen war. Dies ist der einzige Zustandswechsel, den ein Prozeß selbst auslösen kann.

Wakeup: blockiert → bereit
Das Ereignis, auf das der Prozeß gewartet hat ist eingetreten. An den Prozeß eine Nachricht schicken.

Welche Zustände gibt es nicht?

blockiert → aktiv
bereit → blockiert

Zeitkritische Abläufe

Zeitkritische Abläufe sind solche Situationen, in denen zwei oder mehr Prozesse gemeinsam benutzte Daten lesen und schreiben und die Endergebnisse von der zeitlichen Reihenfolge der Lese- und Schreiboperationen abhängig sind.

Was ist ein kritischer Bereich?

Der Teil eines Programms, bei dem auf gemeinsam benutzen Speicher zugegriffen wird, wird als kritischer Bereich bezeichnet.

Semaphore

Semaphor-Operationen sind atomar.

- **DOWN-Operator**
  - Überprüft, ob der Wert der Semaphore größer als Null ist.
  - Falls ja, wird der Wert um Eins erniedrigt und der Prozeß fortgesetzt.
  - Falls nein, wird der ausführende Prozeß blockiert (schlafen gelegt).

- **UP-Operator**
  - Der Wert der Semaphore wird um Eins erhöht.
  - Haben sich Prozesse an der Semaphore blockiert, wird einer ausgewählt und aktiviert (aufgeweckt).
  - Semaphoren können sowohl für den gegenseitigen Ausschluss wie auch für die Synchronisation verwendet werden.

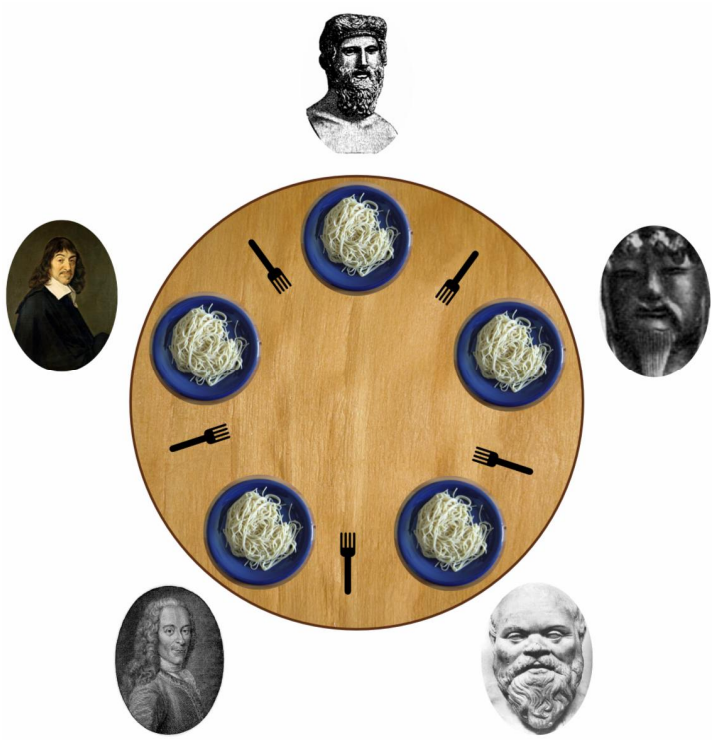
Beispiel: **WICHTIG!!!**

INT Semaphore = 1;

Prozess 1	Prozess 2
while true	while true
{	{
DOWN Semaphore	DOWN Semaphore
tue_etwas	tue_etwas
UP Semaphore	UP Semaphore
}	}

Das Philosophenproblem

Es sitzen fünf Philosophen an einem runden Tisch, und jeder hat einen Teller mit Spaghetti vor sich. Zum Essen von Spaghetti benötigt jeder Philosoph zwei Gabeln. Allerdings waren im Haushalt nur 5 Gabeln vorhanden, die nun zwischen den Tellern liegen. Die Philosophen können also nicht gleichzeitig speisen.



Die Philosophen sitzen am Tisch und diskutieren über philosophische Probleme. Wenn einer hungrig wird, greift er zuerst die Gabel links von seinem Teller, dann die auf der rechten Seite und beginnt zu essen. Wenn er satt ist, legt er die Gabeln wieder zurück und wendet sich wieder der Diskussion zu. Sollte eine Gabel nicht an ihrem Platz liegen, wenn der Philosoph sie aufnehmen möchte, so wartet er, bis die Gabel wieder da ist.

Solange nur einzelne Philosophen hungrig sind, funktioniert dieses Verfahren wunderbar. Es kann aber passieren, dass sich alle fünf Philosophen gleichzeitig entschließen zu essen. Sie ergreifen also alle gleichzeitig ihre linke Gabel und nehmen damit dem jeweils links von ihnen sitzenden Kollegen seine rechte Gabel weg. Nun warten alle fünf darauf, dass die rechte Gabel wieder auftaucht. Dies passiert aber nicht, da keiner der fünf seine linke Gabel zurücklegt. Die Philosophen verhungern.

Das Szenario der fünf speisenden Philosophen wird oft gebraucht, um das Problem der Interprozesskommunikation und Ressourcenverwaltung bei der Entwicklung von Betriebssystemen zu erklären. Das Beispiel soll darstellen, was passieren kann, wenn parallele Prozesse auf die gleichen Ressourcen angewiesen sind und gleichzeitig darauf zugreifen. Dann kann es passieren, dass alle blockiert sind und auf ein Ereignis warten, das durch ihre Blockiertheit nie eintreffen wird. So ein Zustand wird als Deadlock bezeichnet.

Zur Lösung des Problems werden typischerweise Mutexe oder Semaphoren zur Sequentialisierung verwendet.

Unterschied zwischen Pre-emptive und Non-preemptive Scheduling

Pre-emptive Scheduling: Schedulingverfahren, bei dem rechenbereite Prozesse suspendiert werden können.

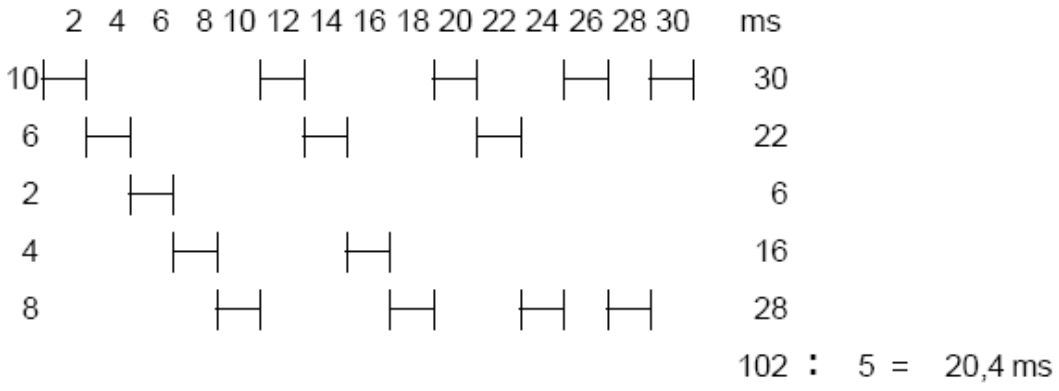
Non-preemptive Scheduling: Schedulingverfahren, bei dem rechenbereite Prozesse nicht suspendiert werden können,

Round-Robin-Scheduling

Jeder Prozeß bekommt ein gewisses Zeitintervall, das Quantum genannt wird, für seine Ausführung zugeteilt.
Annahme: Alle Prozesse gleich wichtig, also erhalten alle dasselbe Quantum. Falls das Quantum eines Prozesse abgelaufen ist, wird ihm der Prozessor entzogen und einem anderen Prozeß gegeben.
Wenn sich ein Prozeß blockiert oder seine Ausführung beendet, bevor das Quantum abgelaufen ist, wird ein Prozesswechsel vollzogen und evtl. Reste des Quantums verfallen. Die Implementierung ist einfach: Die Prozesse werden in einer Liste geführt. Der erste rechnet, suspendierte Prozesse wandern an das Ende der Liste.

Wichtige Frage: Wie lang soll das Quantum sein?
Fazit: Falls das Quantum zu klein ist, sinkt wegen der häufigen Prozesswechsel die Prozessorausnutzung, und falls das Quantum zu groß gewählt wurde, erhält man bei kurzen interaktiven Anfragen schlechte Antwortzeiten.

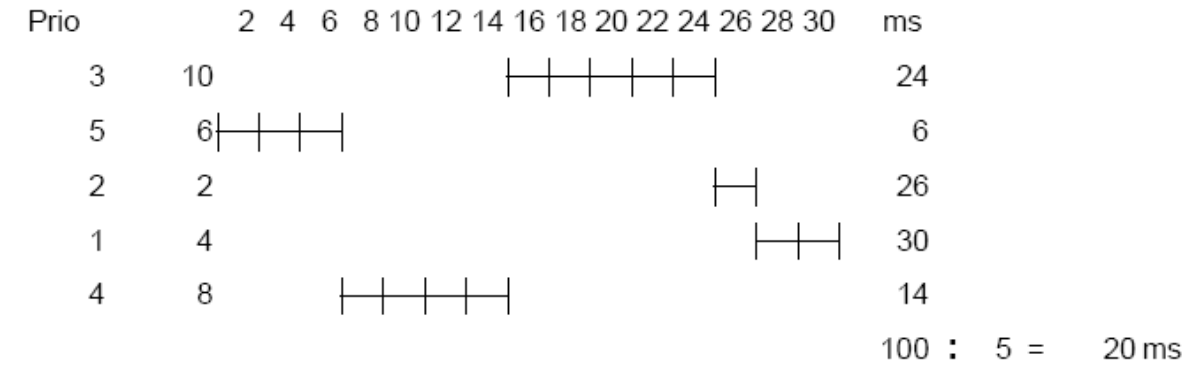
Round-Robin-Scheduling (RR) Quantum = 2 ms



Prioritäts-Scheduling

Beim Prioritäts-Scheduling wird jedem Prozess eine Priorität zugewiesen, und der ausführbereite Prozess mit der höchsten Priorität wird ausgeführt. Damit verhindert wird, dass Prozesse mit einer hohen Priorität zu lange ausgeführt werden, erniedrigt der Scheduler die Priorität des gerade ausgeführten Prozesses bei jeder Unterbrechung. Prioritäten können statisch oder dynamisch vergeben werden. Prozesse werden häufig in Prioritätsklassen eingeteilt, wobei dann innerhalb der Klassen Round-Robin-Scheduling betrieben wird und Prioritäts-Scheduling nur zwischen den Klassen.

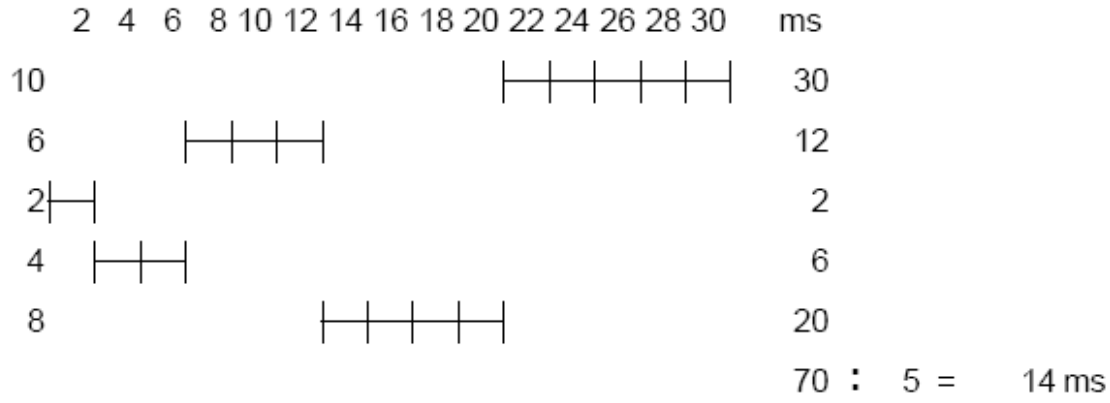
Prioritäts-Scheduling Quantum = 2 ms



Shortest-Job-First !!!GANZ WICHTIG!!!

Sind alle Ausführungszeiten zu Beginn bekannt, ist Shortest-Job-First eine optimale Strategie und liefert minimale durchschnittliche Verweilzeiten.

Shortest-Job-First-Scheduling (SJF) Quantum = 2 ms



CPU-Auslastung

$$Auslastung = 1 - p^n$$

Übung:
Berechnen Sie die Verbesserung der CPU Auslastung bei Erweiterung mit einem 2. und 3. MByte Speicher, wenn beim Grundausbau von 1 MByte das BS 200 KByte und jedes Benutzerprogramm ebenfalls 200 KByte beanspruchen.(80% durchschnittliche I/O-Wartezeit)

Lösung:
Mit 80 % durchschnittlicher I/O-Wartezeit erreicht man eine CPU-Ausnutzung (ohne BS-Overhead) von ca. 60 Prozent. ($(1-0{,}8^4)*100=59{,}04$).
Das Hinzufügen eines weiteren Megabyte Speicher erlaubt nun neun Prozesse ablaufen zu lassen. Steigerung der CPU-Ausnutzung auf 87%. ($(1-0{,}8^9)*100=86{,}58$)
Das Hinzufügen des dritten Megabyte Speicher erlaubt nun 14 Prozesse ablaufen zu lassen. Steigerung der CPU-Ausnutzung auf 96% ($(1-0{,}8^{14})*100=95{,}60$)

Relokation

Das Relokationsproblem besteht darin, dass zum Zeitpunkt des Bindens des Programms nicht bekannt ist, an welchen Speicherplatz das Programm zur Ausführungszeit geladen wird. Der Adressbereich muss deshalb zu Beginn der Ausführung verschoben (relokiert) werden.
Zum Schutz anderer Prozesse wird die Hardware der Maschine mit einem Basis- und einem Grenzregister ausgestattet.

Das **Basisregister** enthält die Startadresse des rechnenden Prozesses.
Das **Grenzregister** die Länge der Partition.

Damit ist es möglich, Zugriffe auf Speicherzellen außerhalb der zugewiesenen Partion zu erkennen.

Durch Änderung des Basisregisters ist auch eine Relokation zur Laufzeit möglich.

Swapping

Reicht der Speicher nicht aus, um alle bereiten Prozesse im Speicher zu halten, ist es notwendig, Speicherbereiche vom Arbeitsspeicher auf den Hintergrundspeicher (Festplatte) aus- und wieder einzulagern. Dieser Vorgang wird Swapping genannt.

Speicherverwaltung

First Fit: Durchsuchen der Segmentliste aufsteigend, bis ein Segment gefunden ist, das groß genug ist. Das Segment wird dann in den belegten Teil und den freien Teil aufgeteilt.
Vorteil: Kurze Suchzeiten.

Next Fit: Wie First Fit, nur dass mit der Suche dort begonnen wird, wo das letzte Mal ein Segment gefunden wurde.
Nachteil: Geringfügig schlechtere Performance als First Fit.

Best Fit Die gesamte Liste wird durchsucht und das Segment mit dem kleinsten Verschnitt

ausgewählt. Die Suche lässt sich beschleunigen, wenn die Liste der freien Segmente der Größe nach sortiert wird.
Nachteil: Die Suchzeit ist länger als bei First Fit. Best Fit tendiert dazu, sehr kleine und damit unbrauchbare freie Segmente zu produzieren.

Buddy-System

Idee: Der Rechner benutzt Binärzahlen zur Adressierung. Dies kann ausgenutzt werden, um die Verschmelzung angrenzender freier Segmente zu beschleunigen.

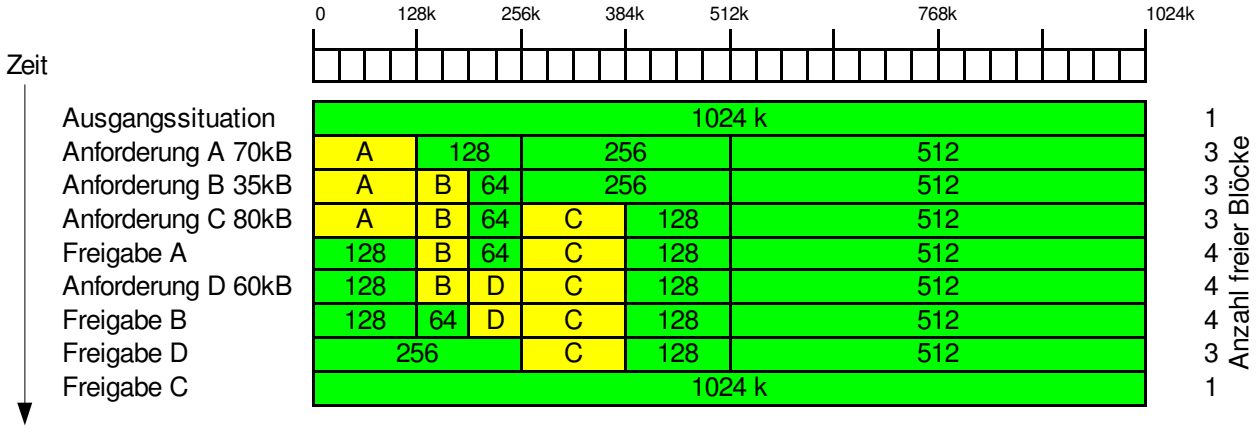
Vorgehen:
Die freien Segmente werden in Listen verwaltet, getrennt nach Segmentgröße von 1,2,4,8,16...Speichergröße Bytes.

Zu Beginn ist der gesamte Speicher frei, alle Listen sind leer, bis auf die für den gesamten Speicher.

Die Speichervergabe sucht passende Blöcke. Sind keine Blöcke der geeigneten Größe vorhanden, werden größere Blöcke durch Halbieren auf die benötigte Größe gebracht.
Bei der Freigabe werden benachbarte Blöcke verschmolzen, wenn sich dadurch ein Segment ergibt, dass auch bei der Verteilung entstanden wäre. Buddy-Systeme sind extrem ineffizient bei der Ausnutzung des Speichers, es entsteht so genannte Fragmentierung. Das Problem rührt von der Tatsache her, dass alle Anfragen auf eine Potenz von 2 aufgerundet werden müssen. Ein 35K Prozeß allokiert somit 64K. Die ergänzenden 29K sind verschwendet. Diese Form des Überhangs wird Interne Fragmentierung genannt. Die Löcher zwischen den Segmenten wird Externe Fragmentierung genannt.

Fragmentierung

Interne Fragmentierung: ist der Unterschied zwischen der Größe des zugeteilten Segments und der benötigten Größe.
Externe Fragmentierung: sind die Lücken zwischen den zugeteilten Segmenten.



Not Recently Used (NRU)

NRU benötigt die hardwaremäßige Unterstützung von benutzt und modifiziert Bits zu jeder Seite. Es entstehen die Klassen:

1. nicht referenziert, nicht modifiziert
2. nicht referenziert, modifiziert
3. referenziert, nicht modifiziert
4. referenziert, modifiziert

Der NRU entfernt zufällig eine der Seiten aus dem Speicher der aus der kleinstnummerierten, nicht leeren Klasse.
NRU ist einfach zu implementieren und hat eine akzeptable Performance.

First In, First Out (FIFO)

Eine Liste verwaltet die Zeitpunkte der Seiteneinlagerung. Die Seite, die am längsten im Speicher war, wird ausgelagert. FIFO wird in dieser reinen Form selten eingesetzt, da auch intensiv genutzte Seiten ausgelagert werden.

Least Recently Used (LRU)

Idee: Wenn Seiten in der (jüngeren) Vergangenheit häufig benutzt wurden, wird angenommen, dass die Seiten auch in der (näheren) Zukunft benutzt werden. Es wird die Seite entfernt, die am längsten unbenutzt ist. Die Implementierung von LRU ist schwierig, da eine Liste der nach Referenzzeitpunkten sortierten Seiten benötigt wird. Bei jeder Referenz muss diese Liste aktualisiert werden, was sehr zeitaufwendig ist oder teure Spezialhardware benötigt.

Beispiel:

Seite	geladen	letzte Referenz	R	M
0	126	279	0	0
1	230	260	1	0
2	120	272	1	1
3	160	280	1	1

a) Welche Seite wird NRU ersetzt? Seite 0!
b) Welche Seite wird FIFO ersetzt? Seite 2!
c) Welche Seite wird LRU ersetzt? Seite 1!

Datei

Die in Dateien gespeicherten Informationen müssen *persistent (=anhaltend z. B. Informationen auf der Festplatte gehen nicht verloren, wenn der Strom abgeschaltet wird)* sein, d.h. nicht von der Erzeugung oder Terminierung von Prozessen abhängen.

I-Nodes

I-Node ist ein Begriff aus dem Computerbereich. In der Dateiverwaltung eines Unix Betriebssystems erreicht man jede Datei auf einer Festplatte oder einem anderen Speichermedium über solche I-Nodes.

Man kann I-Node also im deutschen am besten als Informationsknoten oder Indexeintrag bezeichnen. Der I-Node ist selbst eine kleine Minidatei mit wichtigen Informationen über eine daran anhängende größere Datei mit den eigentlichen gespeicherten Daten.

Der I-Node enthält u.a. eine eindeutige Informationsnummer für genau die Datei , die er verwaltet. Im I-Node sind nicht nur die Dateinummer sondern auch andere Informationen gespeichert:

- Besitzer der Datei
- die Zugriffsrechte der Datei
- Typ der Datei
- Größe der Datei
- Referenzzähler (Anzahl der Verweise aus den Verzeichnissen auf das File existieren - Symbolic Links)
- Datum der Dateierstellung, der letzten Dateiöffnung und der letzten Modifikation der Datei

Der Dateiname ist nicht im I-Node gespeichert.

Der I-Node bildet den Eingang zum Inhalt einer Datei eines Unixbetriebssystems.

Ist eine Datei sehr klein, werden ihre Daten direkt im I-Node gespeichert.
Ist sie größer dann verweist ein Eintrag im I-Node auf einen Datenblock, in dem der Inhalt der Datei gespeichert wird. Reichen die in einem I-Node referenzierten Blöcke für eine Datei nicht aus, zeigt ein Eintrag im I-Node auf weitere I-Nodes, welche nun die eigentlichen Verweise zu den Datenblöcken beinhalten. Wird auf einen einzigen weiteren I-Node verwiesen, dann spricht von einem einfach indirekten Block. Bis zu dreifach indirekte Blöcke sind möglich, so dass die maximale Dateigröße 16 GByte werden kann.

Es gibt in einem Unix Betriebssystem eine Tabelle mit allen aktuellen I-Nodes.
Ferner kann man Listen aller nichtbelegter I-Nodes erstellen.
Man kann auch eine Liste aller inkonsistenten (d.h. noch nicht gespeicherten) Dateien und deren I-Nodes erstellen.

Beispiel: I-Node: Adressierung von Plattenblöcken

- ➢ 10 direkt Blöcke
- ➢ einfach indirekter Block
- ➢ 1 doppelt indirekter Block
- ➢ 1 dreifach indirekter Block

1 KByte = 1 Block
1 Adresse eines Blocks 4 Byte = 32 Bit
1 Block enthält damit 256 Adressen (1024/4)

im I-node:	10 x 1 KB =	10 KB
im einfach indirekten Block	256 x 1 KB =	256 KB
im doppelt indirekten Block	256 x 256 x 1 KB =	65.536 KB
im dreifach indirekten Block	256 x 256 x 256 x 1 KB =	16.777.216 KB
		16.843.018 KB

Absoluter Dateiname vs. Relative Pfadnamen

Jeder Datei wird ein absoluter Dateiname zugeordnet, bestehend aus dem Pfad vom Wurzelverzeichnis zur Datei. Absolute Dateinamen beginnen immer im Wurzelverzeichnis und sind immer eindeutig.

Relative Pfadnamen werden in der gleichen Art und Weise gesucht wie absolute, es wird nur vom Arbeitsverzeichnis aus gestartet anstatt vom Wurzelverzeichnis.

Gemeinsam benutzte Dateien

Ein **symbolischer Link** ist eine Verknüpfung in einem Dateisystem (Datei oder Verzeichnis), die auf eine andere Datei oder ein anderes Verzeichnis verweist. Es ist also lediglich eine Referenz und kein richtiges Element.
Allgemein muss man beachten, dass ein symbolischer Link anders als ein „**harter Link**“ auf einen Pfad zeigt. Ein „harter Link“ zeigt auf einen I-Node, was bedeutet dass ein „harter Link“ immer noch funktioniert wenn die Zieldatei verschoben wurde. Wenn man aber die Zieldatei verschiebt und ein Softlink zeigt drauf, kann die Zieldatei nicht wissen, dass ein Link auf sie zeigt, womit der symbolische Link ins Leere zeigt.

Backup

Inkrementelle Datensicherung

- Die einfachste Form ist die periodische Durchführung einer kompletten Sicherung und die tägliche Sicherung der Dateien, die sich seit der letzten kompletten Sicherung geändert haben.

Dateisystemkonsistenz

- Es können zwei Arten von Konsistenzchecks durchgeführt werden:
    1. Blöcke und
    2. Dateien.

- Um die Blockkonsistenz zu überprüfen, bildet das Programm eine Tabelle mit zwei Zählern pro Block, beide werden mit 0 initialisiert. Der erste Zähler verwaltet, wie oft ein Block in einer Datei präsent ist; der zweite Zähler zählt, wie oft der Block in der Freiliste (bez. in der Bitmap der freien Blöcke) vorhanden ist.

- Das Programm liest dann alle I-Nodes. Beginnend bei einem I-Node ist es möglich, eine Liste von allen Blocknummern, die in der entsprechenden Datei verwendet werden, aufzustellen. Sobald jede Blocknummer gelesen wurde, wird der Zähler in der ersten Tabelle inkrementiert. Das Programm überprüft dann die Freiliste oder die Bitmap, um alle Blöcke zu finden, die nicht verwendet werden. Jedes Auftreten eines Blockes in der Freiliste bewirkt eine Inkrementierung des Zählers in der zweiten Tabelle.

- Wenn das Dateisystem konsistent ist, besitzt jeder Block entweder eine 1 in der ersten oder in der zweiten Tabelle.

- Fehlerbehandlung:

Fehler:	Lösung:
Verlorene Blöcke	Verlorene Blöcke werden der Freiliste zugeordnet
Duplikate in der Freiliste	Ein Eintrag wird aus der Freiliste entfernt
Duplikate belegter Blöcke	Dies ist der schlimmste Fall, da Blöcke in zwei oder mehreren Dateien präsent sind. Einträge von Blocks in der Freiliste werden gelöscht. Für jedes Duplikat werden freie Blöcke allokiert, der Inhalt dorthin kopiert und den betroffenen Dateien zugeordnet. Der Inhalt der Blöcke wird voraussichtlich unbrauchbar sein.

Cache(s)

Cache(s) sind Bereiche, in die Daten unabhängig von der Verarbeitungsreihenfolge eingeschrieben und bei freien Kapazitäten des Peripheriesystems mit diesem ausgetauscht werden. Bekannt sind Caches zwischen CPU und Hauptspeicher, aber auch Festplatten und andere Massenspeicher besitzen Hardwarecaches. Zusätzlich legen moderne Betriebssysteme softwaremäßig File Caches an, die bei heutigen Speichergrößen beträchtliche Umfänge annehmen können. Kennzeichen von Caches ist die nicht geordnete Reihenfolge bei der Zwischenspeicherung. Spezielle Algorithmen suchen die vermutlich als Nächstes benötigten Daten und laden diese auf Vorrat, während andererseits sorgfältig überwacht wird, welche Daten verändert wurden, also zum Datenträger zurückgeschrieben werden müssen.

Nonwirte-Through-Cache: sofortiges Schreiben aller modifizierten Blöcke auf die Platte. (UNIX)
Write-Through-Cache: schreibt modifizierte Blöcke so bald wie möglich auf die Platte. (DOS)

Was ist ein Wurm?

Ein Programm, das sich selbst kopiert. Dies kann z. B. von einem Laufwerk auf ein anderes erfolgen oder indem sich das Programm über eine E-Mail oder einen anderen Transportmechanismus kopiert. Hierdurch können Schäden verursacht oder die Sicherheit des Computers beeinträchtigt werden. Ein Wurm kann in Form eines Scherzprogramms oder einer Scherz-Software auftreten.

Was ist ein Virus?

Viren sind Programme oder Codes, die sich vervielfältigen (d. h., sie infizieren andere Programme, Boot-Sektoren oder Dokumente, die Makros unterstützen, indem sie sich selbst einschleusen oder an das Medium anhängen). Die meisten Viren vervielfältigen sich lediglich. Einige verursachen außerdem Schäden.

Entwurfsprinzipien für die Sicherheit

- Der Entwurf für ein System sollte öffentlich sein. Geheimhaltung dient nur der Täuschung der Entwickler und Betreiber.
- Zugriffe sollten standardmäßig abgelehnt werden.
- Zugriffsrechte sollten bei jedem Zugriff überprüft werden und nicht nur einmal pro Sitzung oder Operationstyp.
- Man sollte jedem Prozeß sowenig Zugriffsrechte einräumen wie möglich (least priviledgeed principle).
- Der Schutzmechanismus sollte einfach, einheitlich und in den untersten Schichten des BS verankert sein und nicht nachträglich integriert werden.
- Das gewählte Schema muss psychologisch akzeptabel sein.

Zugriffskontrolllisten

Jedem Objekt wird eine geordnete Liste zugeordnet, die Zugriffskontrollliste oder ACL.
Jeder Eintrag in der ACL gibt die Benutzeridentifikation, die Gruppenidentifikation und die erlaubten Zugriffe an.

Direkter Speicherzugriff (DMA)

Der Begriff direkter Speicherzugriff oder englisch Direct Memory Access (DMA) eine Zugriffsart, die direkt auf den Speicher zugreift.

Unter DMA (übersetzt direkter Speicherzugriff) ist eine Schaltungs- und Steuermaßnahme zu verstehen, die über spezielle Datenleitungen auf dem Motherboard eine Verbindung zwischen Steckkarten und dem Arbeitsspeicher herstellen. Dadurch können die Daten ohne Umweg über den Prozessor direkt in den Speicher geschrieben werden. So lässt sich die Ausführungsgeschwindigkeit erhöhen.

Will der Prozessor Daten senden oder empfangen, trennt der DMA-Controller den Prozessor vom Bussystem. Der DMA-Controller führt dann die Anforderung mit hoher Geschwindigkeit aus. Danach wird die Verbindung zwischen Prozessor und Bussystem wieder hergestellt. Für den Speichertransfer benötigt der Prozessor ca. 40 Takte. Der DMA-Controller führt den Zugriff innerhalb von 4 Takten aus. Der DMA-Controller ist dafür gedacht, Daten zwischen Arbeitsspeicher und Peripherie zu transportieren und den Prozessor mit diesen Aufgaben zu entlasten.

Gerätetreiber

Die Aufgabe eines Gerätetreibers ist es, eine abstrakte Anfrage von der geräte-unabhängigen Software entgegenzunehmen und dafür zu sorgen, daß die Anfrage ausgeführt wird.
Der gesamte geräte-abhängige Code gehört in den Gerätetreiber.
Der Gerätetreiber behandelt einen Gerätetyp oder höchstens eine Klasse von eng verwandten Geräten.
Ein Gerätetreiber setzt die Kommandos in die Register des Steuerwerkes ab und überprüft die korrekte Ausführung.

Festplattenalgorithmen:

FCFS/FIFO First Come First Served

- Der Plattentreiber akzeptiert immer nur eine Anfrage.
- Die Anfragen werden in der Reihenfolge des Eintreffens bearbeitet.
- Vorteil: fair
- Nachteil: langsam

SSF Shortest Seek Time First

- Der Plattentreiber akzeptiert beliebig viele Anfragen, die in einer Tabelle gespeichert werden.
- Als nächste zu bearbeitenden Anfrage wird diejenige ausgewählt, bei der die erforderliche Armbewegung am kleinsten ist.
- Vorteil: schnelle Antwortzeiten
- Nachteil: Bei stark belasteten Platten tendiert der Arm dazu, sich in der Mitte aufzuhalten, Anfragen nach peripheren Zylindern werden sehr selten bearbeitet.

Aufzugsalgorithmus (elevator algorithm)

- Ein Bit gibt an, in welche Richtung (up, down) sich der Arm bewegt.
- Der Arm bewegt sich solange in eine Richtung, bis alle Anforderungen auf seinem Weg erfüllt sind und kehrt dann um.
- Die Obergrenze für die Armbewegungen ist 2x die Anzahl der Zylinder, um alle Anfragen zu bearbeiten.
- Vorteil: akzeptable Geschwindigkeit und fair

Deadlock-Situation

Für eine Deadlock-Situation müssen die folgenden vier Bedingungen erfüllt sein:

1. **Die Bedingungen des wechselseitigen Ausschlusses:** Jedes Betriebsmittel wird entweder von genau einem Prozeß belegt oder ist frei.

2. **Die Belegungs- und Wartebedingung:** Ein Prozeß, der bereits Betriebsmittel belegt, kann weitere Betriebsmittel anfordern.

3. **Die Ununterbrechbarkeitsbedingung:** Die Betriebsmittel, die von einem Prozeß belegt werden, können nicht entzogen werden, sondern müssen explizit vom belegenden Prozeß freigegeben werden.

4. **Die zyklische Wartebedingung:** Es muss eine zyklische Kette aus zwei oder mehr Prozessen existieren, so daß jeder Prozeß ein Betriebsmittel anfordert, das von dem nächsten Prozeß in der Kette belegt wird.

Alle vier Bedingungen müssen erfüllt sein, damit eine Deadlock-Situation eintreten kann. Falls eine Bedingung nicht erfüllt ist, ist keine Deadlock-Situation möglich.

Deadlock-Behebung

- **mittels Unterbrechung:** Temporärer Entzug eines Betriebsmittels und Zuteilung an einen anderen Prozeß.

- **mittels teilweiser Wiederholung:** Prozesse schreiben ihren Berechnungszustand an sog. Checkpunkten in eine Datei. Aufgrund dieser Beschreibung kann eine Berechnung dann fortgesetzt werden. Der Deadlock wird behoben indem ein Prozeß, der die benötigten Betriebsmittel belegt, auf seinen letzten Checkpunkt zurückgesetzt wird und ihm die Betriebsmittel entzogen werden.

- **mittels Prozessabbruch:** Ein Prozeß wird abgebrochen und seine Betriebsmittel werden entzogen. Können die anderen Prozesse ihre Arbeit dann immer noch nicht fortsetzen, müssen weitere Prozesse abgebrochen werden.

Vermeidung von Deadlocks

Deadlocks können in realen Systemen nicht verhindert werden, da die notwendigen Informationen über zukünftige Betriebsmittelanforderungen nicht verfügbar sind. In realen Systemen versucht man, Deadlocks prinzipiell zu vermeiden.

Bedingung	**Ansatz**
Wechselseitiger Ausschluss	Spooling
Belegungs- und Wartebedingung	Betriebsmittelanforderung zu Beginn der Berechnung
Ununterbrechbarkeitsbedingung	Betriebsmittel entziehen
zyklische Wartebedingung	Betriebsmittel numerisch ordnen