Diplomrelevanz „Betriebssysteme“

Allgemeiner Hinweis:
- wenn explizit nach 3 Strategien gefragt wird, nur 3 Strategien zur Antwort geben, denn der Rest zählt nicht
- auf Punkteangaben achten wegen Zeiteinteilung!!!

Betriebssystem

DIN 44300: Die Programme eines digitalen Rechensystems, die zusammen mit den Eigenschaften dieser Rechenanlage die Basis der möglichen Betriebsarten des digitalen Rechensystems bilden und insbesondere die Abwicklung von Programmen steuern und überwachen.

Aufgabe eines Betriebssystems: ist es, eine Umgebung zu schaffen, in welcher verschiedene Anwendungsprogramme ablaufen könne. Es bietet verschiedene Möglichkeiten an, den Computer einzusetzen.
- virtuelle Maschine
- Betriebsmittelverwalter (Zuordnung der Ressourcen, Verwaltung, Schutz)

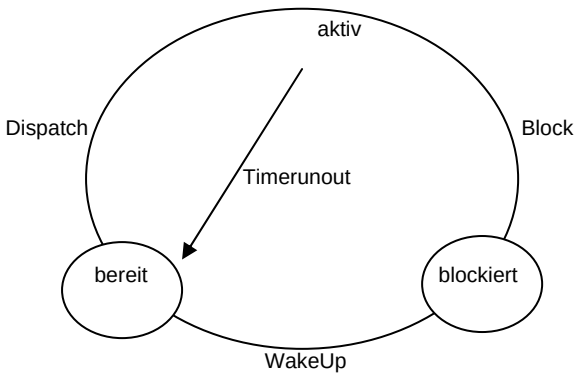
Virtuelle Maschine:
- verbergen der realen Architektur (mit Befehlen wie C:\, DIR, etc. und nicht direkt in der HW-Architektur via Parameter, Spur, Sektor, etc.)
- Bereitstellung einer einfach zu handhabenden virtuellen Maschine

Betriebsmittelverwalter:
- Verwaltung aller Bestandteile des Rechners möglichst effizient
- geordnete und kontrollierte Zuteilung der Ressourcen
- Verwaltung und Schutz

Prozess:
- ist eine Aktivität, die aus Programm, Eingaben, Ausgaben und einem Zustand besteht.

Prozess besteht aus:
- einem ausführbaren Programm,
- den Programmdaten,
- dem Kellerinhalt (Stack),
- dem Programmzähler,
- dem Kellerzeiger (Stackpointer),
- Registerinhalten (Register = sehr schneller Speicher auf der CPU)
- und weiteren Infos, die zur Programmausführung benötigt werden.

Prozesszustände:
- aktiv/rechnend (dem Prozess wird ein Prozessor zugeteilt)
- rechenbereit (Prozess ist ausführbar, aber der Prozessor ist einem anderen Prozess zugeordnet)
- blockiert (Prozess kann so lange nicht ausgeführt werden, bis ein externes Ergebnis eintritt)



Kernel-Systeme:

Monolithische System:
- BS, das keine ausgezeichnete Struktur besitzt
- BS besteht aus einer Menge von Prozeduren.
- keine Mechanismen zum Verbergen von Infos
- Bestandteile:
  - Hauptprogramm, welches angeforderte Dienstprozedur aufruft
  - Dienstprozeduren, die die Systemaufrufe ausführen
  - Hilfsprozeduren, die die Dienstprozeduren unterstützen
  - groß, unbeweglich, alles in einem

Client-SRV-Modell (Mikro-Kernel):
- klein
- Linux, Unix
- Systeme funktionieren über Grenzen hinweg
- nur die Funktionen enthalten, die im privilegierten Modus ablaufen müssen

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

kritischer Bereich:
- der Teil eines Programms, bei dem auf gemeinsam benutzen Speicher zugegriffen wird
- zu jedem Zeitpunkt darf sich höchstens ein Prozess in einem kritischen Abschnitt befinden
- es dürfen keine Annahmen über die Ausführungsgeschwindigkeit oder die Anzahl der Prozessorengemacht werden
- kein Prozess, der sich nicht in einem seiner kritischen Abschnitte befindet, darf andere Prozesse blockieren
- kein Prozess soll unendlich lange warten müssen, bis er in seinen kritischen Bereich eintreten kann

Semaphore:
- Variable (meist Integer), die definiert wird
- es gibt 2 Operationen:
  - DOWN (wenn Semaphore > 0, dann Wert um 1 mindern und Prozess fortsetzen, sonst wird Prozess blockiert)
  - UP (Wert der Semaphore wird um 1 erhöht, sind Prozesse blockiert, wird ein Prozess ausgewählt und aktiviert)

Beispiel:

```
Int Semaphore = 1

Prozess 1                    Prozess 2

While .+. {                  While .+. {
Down Semaphore               Down Semaphore
tue_etwas                    tue_etwas
Up Semaphore                 Up Semaphore
}                            }
```

siehe Extrablatt zum Philosophenproblem (Codierung)!!!
- Meinung zu den 3 Varianten äußern
- man kann besser machen:
  - nur 1 Gabel für jeden
  - Semaphoren einsetzen
  - Gar nicht essen
- wenn langer Code, dann können 2 Personen essen

Scheduling

- der Teil des BS, der mit der Entscheidung betraut ist, welcher Prozess als nächstes ausgeführt werden soll heisst Scheduler

- (mögliche) Aufgaben:
  - **Fairness** (jeder Prozess erhält einen gerechten Anteil der Prozessorzeit)
  - **Effizienz** (Prozessor wird immer vollständig ausgelastet)
  - **Antwortzeit** (Antwortzeit für interaktiv arbeitende Benutzer wird immer minimiert)
  - **Verweilzeit** (Wartezeit auf die Ausgaben von Stapelaufträgen wird minimiert)
  - **Durchsatz** (Anzahl der Aufträge, die in einem bestimmten Zeitintervall ausgeführt werden, wird maximiert)

Pre-emptives-Scheduling ist jenes Verfahren, bei dem rechenbereite Prozesse suspendiert werden. Nur dieses Verfahren ist für den praktischen Einsatz in Rechnern relevant.

Non-pre-emptives Scheduling ist jenes Verfahren, bei dem rechenbereite Prozesse nicht suspendiert werden (Run-to-Completion-Verfahren). Zustand „TimeOut“ kommt nicht vor!

Round-Robin-Scheduling
RRS ist das älteste, einfachste, weit verbreiteste und fairste Verfahren. Jeder Prozess bekommt ein gewisses Zeitintervall (Quantum) für seine Ausführung zugeteilt. Alle Prozesse sind gleich wichtig – bekommen also auch alle das gleiche Quantum. Ist das Quantum eines Prozesses abgelaufen, so wird ihm der Prozessor entzogen und einem anderen Prozess gegeben. Wenn sich ein Prozess blockiert oder die Ausführung beendet ist bevor das Quantum abgelaufen ist, wird ein Prozesswechsel vollzogen und etwaige Reste des Quantums verfallen.
Die Implementierung ist einfach. Die Prozesse werden in einer Liste geführt. Der erste Prozess rechnet – suspendierte Prozesse gehen an das Ende der Liste.
Länge des Quantums stellt eine wichtige Frage dar. Ist das Quantum zu klein gewählt, so sinkt wegen der häufigen Prozesswechsel die Prozessorausnutzung. Ist das Quantum zu gross erhält man bei kurzen interaktiven Anfragen schlechte Antwortzeiten.
Beispiel: Sektempfang

Prioritäts-Scheduling
Jedem Prozess wird eine Priorität zugewiesen und der ausführbereite Prozess mit der höchsten Prio wird ausgeführt. Um zu verhindern, dass Prozesse mit hoher Prio zu lange ausgeführt werden, reduziert der Scheduler die Prio des gerade ausgeführten Prozesses bei jeder Unterbrechung.
Prios können statisch oder dynamisch vergeben werden.
Prozesse werden häufig in Prio-Klassen eingeteilt, wobei dann innerhalb der Klassen RRS betrieben wird und Prio-Schedluing nur zwischen den Klassen.

Shortes-Job-First
SJF ist eine Strategie, die besonders gut bei Stapelaufträgen geeignet ist, bei denen die Ausführungszeiten im Voraus bekannt sind. Sind alle Ausführungszeiten im Voraus bekannt, so ist SJF eine optimale Strategie und liefert minimale durchschnittliche Verweilzeiten.

First Come First Served
Beispiel: Mc Donalds

Verbesserung der CPU-Auslastung

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. **Auslastung = 1 - p^n** (n = Anzahl der Prozesse)

1 MB	+	1 MB	+	1 MB
- 200 KB - 200 KB - 200 KB - 200 KB - 200 KB (BS)		- 200 KB - 200 KB - 200 KB - 200 KB - 200 KB		- 200 KB - 200 KB - 200 KB - 200 KB - 200 KB
Auslastung (A) = $1 – 0{,}8^4$ = 60%		A = $1 – 0{,}8^9$ = 87%		A = $1 – 0{,}8^{14}$ = 96%
		→ Verbesserung um 45%		**→ Verbesserung um 10%**
		→ Investition ist sinnvoll		**→ Sinn der Investition ???**

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 daher zu Beginn der Ausführung verschoben (relokiert) werden.
Basisregister: enthält Startadresse des rechnenden Prozesses
Grenzregister: enthält die Länge der Partition

Swapping: Genügt der Speicher nicht um alle bereiten Prozesse im Speicher zu halten, ist es notwendig Speicherbereiche vom Arbeitsspeicher auf den Hintergrundspeicher (HD) aus- und wieder einzulagern.

Algorithmen:

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

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

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 FF, zudem werden sehr kleine und somit unbrauchbare freie Segmente produziert.

Beispiel: benötigter Speicherplatz 12K, 10K, 9K

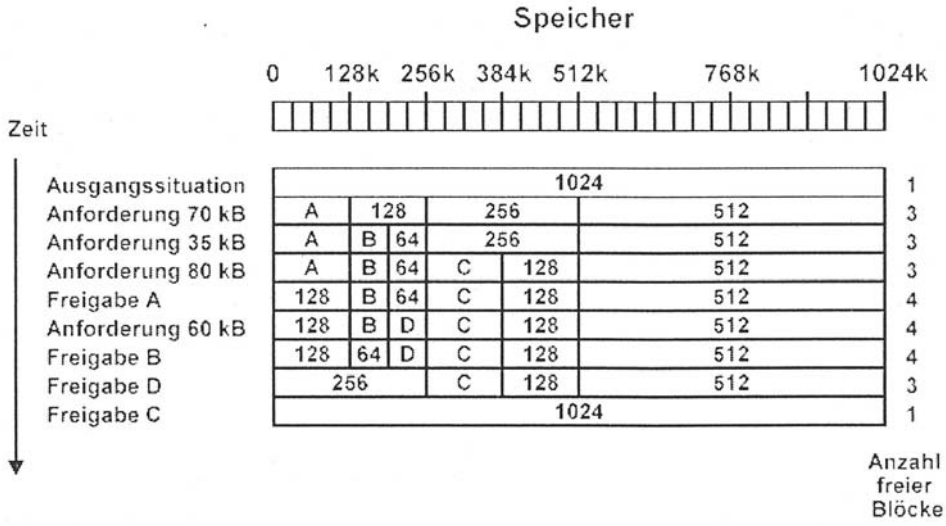
	10 K	**4 K**	**20 K**	**18 K**	**7 K**	**9 K**	**12 K**	**15 K**
FF	10 K 0 K		12 K 8 K	9 K 9 K				
NF			12 K 8K	10 K 8 K		9 K 0 K		
BF	10 K 0 K					9 K 0 K	12 K 0 K	

Buddy-System

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

Getrennt nach Segmentgrösse (1, 2, 4, 8, 16, ...) werden die freien Segmente in Listen verwaltet. Zu Beginn ist der gesamte Speicher frei und alle Listen sind leer, bis auf jene für den gesamten Speicher. Die Speichervergabe sucht passende Blöcke. Wenn keine Blöcke der geeigneten Grösse vorhanden sind, werden grössere Blöcke durch Halbieren auf die benötigte Grösse gebracht. Bei der Freigabe werden benachbarte Blöcke miteinander 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 sogenannte Fragmentierung. Das Problem entsteht dadurch, 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 als **externe Fragmentierung** bezeichnet.



Die in Dateien gespeicherten Informationen müssen **persistent** sein, das bedeutet **nicht von der Erzeugung oder Terminierung von Prozessen abhängen**.

Relative Pfadnamen werden in der gleichen Art und Weise gesucht wie **absolute Pfadnamen**, es wird nur vom Arbeitsverzeichnis (Bsp.: Eigene Dateien) aus gestartet anstatt vom Wurzelverzeichnis (C:\VWA\Informatik\Betriebssysteme.....).

I-Nodes

Zu jeder Datei wird ein I-Node eingerichtet. I-Nodes sind kleine Tabellen, welche die Attribute und die Plattenadressen der Dateiblöcke enthalten. Die ersten Plattenadressen werden im I-Node selbst gespeichert, so dass für kleine Dateien die notwendigen Infos im I-Node selber steckt, welche durch das Öffnen der Datei von der Platte in den Hauptspeicher geholt wird. Bei grösseren dateien ist eine der Adressen in dem I-Node die Adresse eines Plattenblockes, der auch einfacher indirketer Block genannt wird. Dieser Block enthält weitere Plattenadressen.

Beispielrechnung:

Blockgrösse	1.024 Byte
Anzahl der Adressen pro Block	256

10 I-Node Adressen	10 * 1.024	10.240
einfach indirekt	256 * 1.024	262.144
zweifach indirekt	256 * 256 * 1.024	67.108.864
dreifach indirekt	256 * 256 * 256 * 1.024	17.179.869.184
		∑ 17.247.250.432

Kbyte: 1.024	16.843.018
Mbyte: 1.024	16.448,259
Gbyte: 1.024	16,0627

Seitenersetzungtsalgorithmen:

Not Recently Used
NRU benötigt hardwaremässigen Support von **benutzt** und **modifiziert** Bits zu jeder Seite. NRU entfernt zufällig eine der Seiten aus dem Speicher der aus der kleinstnummerierten, nichtleeren Klasse. Folgende Klassen entstehen:

- nicht referenziert, nicht modifiziert
- nicht referenziert, modifiziert
- referenziert, nicht modifiziert
- referenziert, modifiziert

NRU ist einfach zu implementieren und hat eine akzeptable Performance.

Firt In, First Out
Eine Liste verwaltet die Zeitpunkte der Seiteneinlagerung. Die Seite, die am längsten im Speicher war, wird ausgelagert.
Einsatz von FIFO erfolgt nur selten, da auch intensiv genutzte Seiten ausgelagert werden.

Second Chance
SC vermeidet es, soweit möglich, intensiv genutzte Seite auszulagern.
Soll eine Seite ersetzt werden, deren Benutzt-Bit gesetzt ist, wird das Benutzt-Bit gelöscht und der Einlagerungszeitpunkt auf die aktuelle Zeit gesetzt und die Seitren ans Ende der Liste verschoben. Ist das Benutzt-Bit nicht gesetzt, so ist die Seite alt und unbenutzt und kann sofort ersetzt werden.

Least Recently Used
Die Idee hierbei ist, dass wenn Seiten in der jüngeren Vergangenheit häufig benutzt worden sind, diese auch in der näheren Zukunft benutzt werden. Somit 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.

Übung: Ein Rechner besitzt 4 Seitenrahmen:

Seite	Geladen	Letzte Referenz	Referenz	Modifizieren
0	126	279	0	0
1	230	260	1	0
2	120	272	1	1
3	160	280	1	1

1.) Welche Seite wird zuerst ersetzt beim Algorithmus NRU? → Seite 0

2.) Welche Seite wird zuerst ersetzt beim Algorithmus FIFO? → Seite 2

3.) Welche Seite wird zuerst ersetzt beim Algorithmus LRU? → Seite 1

Symbolischer Link:

Die gemeinsame Benutzung der Dateien führt zu einigen Problemen, die man löst, indem man entweder die Plattenblöcke nicht in den Verzeichnissen einträgt, sondern in einer separaten Datenstruktur (z.Bsp.: I-Nodes) oder die Verbindung wird mittels einer vom System erzeugten Datei vom Typ LINK erzeugt, die nur den Pfadnamen der Datei enthält.

Inkrementelle Datensicherung

- reduziert den Aufwand
- einfachste Form ist die periodische Durchführung einer kompletten Sicherung und die tägliche Sicherung der Dateien, die sich seit der letzten kompletten Sicherung geändert haben.
- Dazu muss eine Liste der Sicherungsdaten für jede Datei auf der Platte verwaltet werden.

Dateisystemkonsistenz

1. Blöcke und 2. Dateien

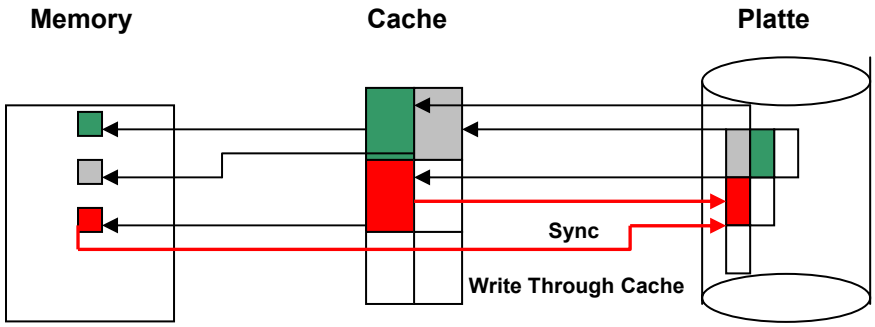
Um die Blockkonsistenz zu überprüfen, bildet das Programm eine Tabelle mit zwei Zählern pro Block, beide werden mit „0“ initialisiert. Der erste Zähler verwaltet, wie oft ein Block in einer Datei präsent ist; der zweite Zähler zählt, wie oft der Block in der Freiliste vorhanden ist.
Das Programm liest 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.
Ist das Dateisystem konstant, so besitzt jeder Block entweder eine „1“ in der ersten oder in der zweiten Tabelle.

Fehlerbehebung:
- verlorene Blöcke (werden der Freiliste zugeordnet)
- Duplikate in der Freiliste (Eintrag wird aus der Freiliste entfernt)
- Duplikate belegter Blöcke (Einträge in der Freiliste entfernen, für jedes Duplikat freie Blöcke allokieren und Inhalt dorthin kopieren)

Cache

Cache ist eine Sammlung von Blöcken, die auf die Platte gehören, aber aus Gründen der Performance im Speicher gehalten werden. Der gebräuchlichste Algorithmus zur Verwaltung des Cache ist, alle Leseanfragen zu überprüfen, ob der benötigte Block im Cache ist.

UNIX erlaubt mit dem Befehl *sync* das sofortige Schreiben aller modifizierten Blöcke auf die Platte (sog. *Nonwrite-Through-Cache*). MS-DOS schreibt modifizierte Blöcke zu bald wie möglich auf die Platte (sog. *Write-Through-Cache*).



Trojanische Pferde

Veränderung eines Programms, z. Bsp. dem Editor, so dass neben der ursprünglichen Funktionalität zusätzliche Funktionen zur Umgehung der Systemsicherheit vorhanden sind. Z. Bsp.: Dateien stehlen oder kopieren.

Wurm

Ein Wurm ist ein sich selbst replizierendes Programm, dass sich somit innerhalb von Sekunden auf alle erreichbaren Rechner ausbreiten kann. Ein Wurm besteht aus einem Bootstrap-Programm und dem eigentlichen Wurm.

Viren

Ein Virus ist ein Programmstück, dass sich an ein gültiges Programmstück anfügt, mit der Absicht, andere Programme zu infizieren.

Entwurfprinzipien 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 so wenig 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. (Bsp.: ..., denn 20-stellige Passwörter werden notiert)

Zugriffskontrollisten

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

Direkter Speicherzugriff (Direct Memory Access)

Viele Steuergeräte, insbesondere für blockorientierte Geräte, unterstützen den direkten Speicherzugriff. Der Prozessor übergibt dem Steuerwerk die Blockadresse auf der Platte und die Speicheradresse, an die der gelesene Block kopiert werden soll, sowohl die Anzahl der zu lesenden Bytes.
Nachdem das Steuerwerk vom Gerät den ganzen Block gelesen hat und die Prüfsummer überprüft hat, kopiert es die geforderte Anzahl Bytes an die angegebene Adresse und löst eine Unterbrechung aus, wenn der Vorgang beendet ist.

Gerätetreiber

Aufgabe liegt darin, eine abstrakte Anfrage von der geräte-unabhängigen SW entgegen zu nehmen und dafür zu sorgen, dass die Anfrage ausgeführt wird.
Der gesamte geräte-abhängige Code gehört in den Gerätetreiber.

Algorithmen des Plattenarm-Scheduling

FSFS / FIFO (First Come First Served):
- Plattentreiber akzeptiert immer nur eine Anfrage
- Anfragen werden in der Reihenfolge des Eintreffens bearbeitet
- Vorteil: fair
- Nachteil: langsam

SSF (Shortest Seek Time First):
- Plattentreiber akzeptiert beliebig viele Anfragen, die in einer Tabelle gespeichert werden
- als nächste zu bearbeitende 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 und Anfragen nach peripheren Zylindern werden selten bearbeitet

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

Verklemmungen:

Eine Menge von Prozessen befindet sich in einem Deadlock-Zustand, falls jeder Prozess der Menge auf ein Ereignis wartet, das nur ein anderer Prozess der Menge auslösen kann.

Bedingungen:
- wechselseitiger Ausschluss (jedes BM wird entweder genau von einem Prozess belegt oder ist frei)
- Belegungs- und Wartebedingung (Prozess, der bereits BM belegt, kann keine weiteren BM anfordern
- Ununterbrechbarkeitsbedingung (BM, die von einem Prozess belegt werden, können nicht entzogen werden, sondern müssen explizit vom Prozess freigegeben werden)
- zyklische Wartebedingung (es muss eine zyklische Kette aus 2 oder mehr Prozessen existieren, so dass jeder Prozess ein BM anfordert, dass vom nächsten Prozess in der Kette belegt wird)

Behebung:
- mittels Unterbrechung (temporärer Entzug eines BM und Zuteilung an einen anderen Prozess)
- mittels teilweiser Wiederholung (Prozess, der die benötigten BM belegt, wird auf seinen letzten Checkpunkt zurückgesetzt und das BM entzogen -> Word, Excel,...)
- mittels Prozessabbruch (Prozess wird abgebrochen und dessen BM entzogen; können andere Prozesse immer noch nicht fortsetzen, so müssen weitere Prozesse abgebrochen werden)

Vermeidung: Deadlocks können in realen Systemen nicht verhindert werden, da die notwendigen Informationen über zukünftige BM-Anforderungen nicht verfügbar sind -> Vermeidung

Bedingung	**Ansatz**
Wechselseitiger Ausschluss	Spooling
Belegungs- und Wartebedingung	BM-Anforderung zu Beginn der Berechnung
Ununterbrechbarkeitsbedingung	BM entziehen
Zyklische Wartebedingung	BM numerisch ordnen