Fragenkatalog zum Hauptdiplom Informatik - Kurs Betriebssysteme
(Ursprünglich zusammengestellt von M.Dugas; erweitert und aktualisiert von HG Wefels)

Betriebssysteme

KE1 - Einführung

Welche Aufgaben hat ein Betriebssystem?
Schließen der sogenannten ‘Semantischen Lücke’ zwischen der Rechnerhardware-/Mikroprogrammebene und der Anwendungsebene durch abstrahieren von der Komplexität der Hardware, stellt Anwendungen eine wesentlich einfachere und dadurch mächtigere Maschinenschnittstelle zur Verfügung
Steuerung angeschlossener Peripheriegeräte (I/O, Platten-/Arbeitsspeicher, Datenaustausch und Kommunikation)
Überwachung von Zugriffsrechten -> Sicherheit von Programmen und Daten
Behandlung von Ausnahmesituationen wie z.B. Division durch 0 (allgemeiner: unzulässige Operation), Störungen in Geräten wie z.B. Papierstau, Leitungsfehler, Programmfehler (Fehler sollten so dicht wie möglich bei der Schicht des Verursachers behandelt werden; so z.B sollten Hardwarefehler so dicht wie möglich an der Hardware selbst behandelt werden)
Bereitstellung bestimmter Betriebsarten wie z.B. Batch-, Echtzeit- oder Dialogbetrieb.
Prozeßverwaltung/-kommunikation/Synchronisation, Organisation des Mehrprogrammbetriebes (parallele Ausführung, Threading)
Verwaltung von Betriebsmitteln (Drucker, Speicher, CPU)
Bereitstellung von Kommandosprache(n)
Unterstützung von Administrationsaufgaben (Datensicherung, Systemkonfigurierung, Benutzerrechte)
Kostenabrechnung
Wie funktionieren Interrupts?
Ein Interrupt ist eine asynchrone Unterbrechung des Programmablaufes. Dabei wird der Zustand des Prozessors und der Zustand der unterbrochenen Anwendung z.B. auf einen Stackspeicher gerettet. Zuvor wird jedoch ausgewertet, ob es sich überhaupt um einen zulässigen (MI=maskable Interrupt) und nicht gesperrten Interrupt handelt oder ob der Interrupt maskiert war, oder ob es sich um einen NMI (=nonmaskable Interrupt) gehandelt hat. Nach dem Retten der Prozessor- und Programmzustände wird dann eine Interrupt Service Routine (ISR) ausgeführt. Nach deren Abarbeitung werden die CPU- und die Programmzustände vom Stack restauriert und die Programmausführung wird fortgesetzt.
Was versteht man unter dem Ebenen- oder Schichtenmodell eines Rechners?
Gatter Ebene
Mikroprogrammebene
konventionelle Maschinenebene
Betriebssystemebene API (applikation programming interface, Betriebssystemaufrufe)
Assemblerebene
Anwendungsebene

KE2 - Geräteverwaltung und Dateisysteme

Welche Speichertypen gibt es?
RAM
ROM
EPROM
EEPROM
FD

HD
CD-ROM
Band
WORM
MO
Warum verwendet man HDs?
HDs sind billiger als RAM (Verhältnis 1:30 derzeit) und ihr Inhalt überlebt in der Regel eine Rechnersitzung und/oder das Programmende ("nichtflüchtig"). Außerdem ist die verfügbare Speicherkapazität wesentlich größer. HDs können als Medium für Backups, zum Replizieren "Mirroring" von Datenbeständen (‘RAID 0’) oder als Baustein zur Konstruktion fehlertoleranter Disk-Arrays (‘RAID 1-5’) dienen. (Die Zahlen geben den Hamming-Abstand der verwendeten Codes - und damit die Fehlertoleranz - an).
Wie funktionieren HD-Zugriff und Controller?
Der Controller steuert die Datenübertragung von der Platte in den Rechner und umgekehrt. Er sorgt für die korrekte Positionierung der Schreib-/Leseköpfe, puffert den Datenstrom, adaptiert mech. und elektrisch an den Prozessorbus (Stichwort. Hostadapter), schreibt und liest Daten per DMA direkt aus/in den Hauptspeicher.
Die mittlere Zugriffszeit einer Festplatte ergibt sich aus Suchzeit(Positionierung) + Latenzzeit(Zeit bis gesuchter Sektor unter den Köpfen erscheint) + Übertragungszeit(Zeit zum Lesen des Sektors)
Welches sind die Eigenschaften virtueller Geräte?
Virtuelle Geräte stellen durch das Betriebssystem simulierte Geräte mit idealisierten Eigenschaften dar. Bewerkstelligt wird damit die Abstraktion von den Eigenschaften realer Geräte mit dem Ziel der Geräteunabhängigkeit. In der Regel stellen virtuelle Geräte jedem Prozeß eine synchrone Operation zur Verfügung. Dabei kann es erheblich mehr virtuelle als reale Geräte geben: z.B. Druckerspooler simuliert Drucker für jeden Anwender. Beispiele virtueller Geräte: Dateien, Drucker.
(SPOOL-System: Abk. für **S**imultaneous **P**eripheral **O**peration **O**n **L**ine)
Virtuelle Geräte können andere Funktionalitäten haben als reale Geräte.
Wie kann ein Prozeß auf Daten auf der Festplatte zugreifen?
Das virtuelle Gerät hinter dem die Festplatte verborgen wird, heißt Datei
Prozeß öffnet Datei und erhält eine Datei-ID die auf eine Datei Deskriptor tabelle verweist.
Adressierung der Sektoren
FAT, i-nodes = Sektoradresstabelle (direkte, indirekte, doppelt indirekte, dreifach indirekte Sektoradresse), Sektorfolgen
Übertragung Hauptspeicher-> Controller-> Platte
Zugriffszeiten (was dauert am längsten? =Positionierung)
Stichworte: SSN(Shortest-Seek-Next oder Elevator-Algorithmus)
Maßnahmen zur Minimierung der Suchzeit
Rotationsgeschindigkeit, Wahl der Positionierungsstrategie, Interleave
Wie teilt der Controller dem BS das Ende der Übertragung mit?
Durch einen Interrupt oder ein spezielles Register.

KE3 - Prozeß- und Prozessorverwaltung

Was ist ein Prozeß?
Ein Prozeß ist die Abstraktion eines in Ausführung befindlichen Programms einschließlich der aktuellen Werte des Programmzählers, der Prozessorregister und der Programmvariablen. Jeder quasiparallel ausgeführte Prozeß besitzt konzeptionell gesehen seinen eigenen Prozessor.
Welche Zustände kann ein Prozeß annehmen?
initiiert
bereit

aktiv
terminiert
blockiert
Beim Übergang initiiert nach bereit ist der Prozeß bereit zur Prozessorzuteilung und wird in die Warteschlange der bereiten Prozesse aufgenommen.
Beim Übergang von bereit nach aktiv werden Anweisungen des Prozesses auf dem Prozessor ausgeführt.
Ursache für einen Übergang von aktiv nach blockiert ist das Warten auf ein bestimmtes Ereignis: (E/A, Speicherzuteilung, etc.)
Beim Übergang von blockiert nach bereit ist das erwartete Ereignis eingetreten.
Der Übergang aktiv nach bereit wird durch eine Vorrangunterbrechnung (preemption) durch einen Prozeß höherer Priorität ausgelöst.
Was steht im Widerspruch zu hohem Durchsatz und guter Auslastung?
Im Widerspruch dazu stehen gutes Antwortzeitverhalten im Dialog- und Stapelbetrieb! Denn hoher Durchsatz und gute Auslastung bedingen viel Prozesse im Rechner, so daß jeder Prozeß nur rel. selten 'dran' kommt; d.h. das Antwortzeitverhalten wird schlechter. Auch die Zuteilungsfairneß wird dadurch zunehmend schlechter; d.h. unfairer.
Erklären Sie Round Robin!
RR ist ein Zeitscheibenverfahren. Jeder neu ins System kommende Prozeß wird an das Ende einer Liste angehängt. Der aktuelle Prozeß wird an das Ende der Liste angehängt. Der aktuelle Prozeß wird bis zum Ende seiner Zeitscheibe, bis zum Ende des Prozesses oder bis zu seinem Blockieren ausgeführt (je nachdem was eher eintrifft). Der Scheduler wechselt dann zum nächsten Prozeß und hängt den alten Prozeß ans Ende der Liste (falls dieser noch nicht terminiert ist).
Nennen Sie Zuteilungsalgorithmen mit Vorrangunterbrechnung (preemptive Verfahren)!
Round Robin
LCFS (Last come first served)
DPRR(Dynamic Priority Round Robin)

KE4 - Hauptspeicherverwaltung

Wie kann der Hauptspeicher vom Betriebssystem verwaltet werden?
Im Benutzermodus bildet das Betriebssystem die physischen Adressen des Rechners auf logische Adressen ab. Auf diese Art kann das Betriebssystem jedem Prozeß einen eigenen logischen Hauptspeicher zur Verfügung stellen. Man unterscheidet folgende Speicherzuweisungsstrategien:
einfache zusammenhängende Speicherzuweisung => nur ein Prozeß im Hauptspeicher
mehrfache zusammenhängende Speicherzuweisung => mehrere Prozesse im Hauptspeicher, jeweils kontinuierlicher Adressbereich
Bei dieser Art der Speicherverwaltung für mehrere Prozesse unterscheidet man (mit Hilfe von Abkürzungen aus der IBM OS/360-Welt :
MFT (Multiprogramming with a fixed number of Tasks):
Speicher wird in Segmente gleicher Größe geteilt => Prozessen werden z.T. zu große Segmente zugewiesen, dies führt zu interner Fragmentierung.
MVT (Multiprogramming with a variable Number of Tasks)
variable Segmentaufteilung => nach der Terminierung von Prozessen entstehen Lücken zwischen Segmenten von ebenso variabler Größe => externe Fragmentierung
Strategien zum Auffinden von 'passenden' Speichersegmenten:
First Fit, Rotating First Fit, Best Fit, Worst Fit, Buddy. Beim Buddy-Verfahren wird der Speicher rekursiv solange in zwei Hälften (Buddies) geteilt, bis eine Größe erreicht ist, die der der Prozeßgröße nächstgrößeren 2er Potenz entspricht. Beim Freigeben werden Buddies immer paarweise wiedervereinigt. Verwaltet werden Buddies in Listen oder Baumstrukturen.

nicht zusammenhängende Speicherverwaltung
Trennung von Programm und Daten in verschiedene (kleinere) Segmente. Dadurch können Speicheranforderungen eher befriedigt werden.
Seitenorientierte Speicherzuweisung
Hierbei handelt es sich ebenfalls um eine Form der nicht zusammenhängenden Speicherzuweisung. Der physische Hauptspeicher wird dabei in Abschnitte der Größe p unterteilt, die man pageframe nennt. Der logische Hauptspeicher eines Prozesses wird in Seiten der Länge p unterteilt; eine Speicherzelle berechnet sich dann zu s= x div p (Seitennummern) und d= x mod p (Speicherzellenadresse in der Seite s). Eine Seitentabelle ST verwaltet zu jeder Seite s eines Prozesses die Seitenrahmennummer in der diese Seite gerade gespeichert ist. Der Zusammenhang zwischen physischen und virtuellen Adressen ist gegeben wie folgt: r= ST[v div p] * p + (v mod p). Vorteil: Es werden jeweils nur die benötigten Seiten eingelagert, der logische Hauptspeicher kann problemlos (durch die HD) verlängert werden, Prozesse müssen nicht unbedingt durchgehenden Hauptspeicher besitzen, jede Seite ist individuell schützbar, platzsparender Code durch gemeinsame Nutzung von Seiten (shared memory/shared Libraries) möglich. Außerdem hat diese Strategie keine Probleme mit dynamisch - also zur Laufzeit - erzeugten Variablen, da einfach Speicherseiten angefordert werden können.
Welche Strategien gibt es bei zusammenhängender Speicherverwaltung?
Einfache Speicherverwaltung, MFT, MVT. Zum Auffinden des passenden Segmentes: First-Fit, Rotating-First-Fit, Best-Fit, Worst-Fit, Buddy
Was versteht man unter einem virtuellen Hauptspeicher?
Darunter versteht man einen logischen Adressraum, der in der Regel größer ist als der physische Hauptspeicher (Aber auch der umgekehrte Fall ist möglich: Man denke an die EMS-Spezifikation für 8086-Prozessoren). Man erreicht dies durch Einteilung des physischen Speichers in Seitenrahmen, wobei bei Bedarf Seiten auf den Hintergrundspeicher ein- oder ausgelagert werden können (demand Paging).
Die Beobachtung, daß jeder Prozeß zu einem gegebenen Zeitpunkt mit einer Arbeitsmenge ("Working Set") an Code und Daten auskommt, die wesentlich kleiner ist als der gesamte Prozeß (Lokalitätsannahme), führt auf die Idee des Pagings bzw. Swappings: Es werden nur jeweils diejenigen Seiten in den Hauptspeicher eingelagert, die aktuell zur Arbeitsmenge eines Prozesses gehören. Der physische Speicher besteht aus den Seitenrahmen, während der virtuelle Speicher in Seiten unterteilt ist! Grundlage dieses Speichermodells ist die Lokalitätsannahme (s.o.). Vorteile: Nur ein Teil der Daten muß im Hauptspeicher gehalten werden, fast beliebig große Adressräume möglich, virtueller Speicher muß nicht durchgehend benutzt werden, memory mapped I/O ist möglich; darunter versteht man die ‘Einblendung’ von Speicherseiten aus Dateien in den virtuellen Arbeitsspeicher, so daß diese von einem oder mehreren Prozessen bearbeitet werden können.
Wie werden die Seitenrahmen verwaltet?
Die Seitenrahmen werden in einer Seitenrahmentabelle verwaltet, diese enthält zu jedem Seitenrahmen die zugehörige Seitennummer sowie den zugehörigen Prozeß.
Zur Abgrenzung: Zu jedem Prozeß gibt es eine Seitentabelle in der für jede Seite des Prozesses die aktuell zugehörige Seitenrahmennummer verzeichnet ist.
Was passiert wenn alle Seitenrahmen belegt sind?
Dann müssen Seiten ausgelagert werden. Man vermeidet unnötige Auslagerungen mit Hilfe des sog. Dirty Bits einer Seite: Ist das Dirty-Bit (deutsch verändert oder referenziert Bit) zum Auslagerungszeitpunkt nicht gesetzt, dann existiert auf der Platte noch eine gültige Kopie und man kann sich das teure Zurückschreiben sparen. Auslagerungsstrategien sind: LRU least recently used, FIFO, Clock, Second Chance. LRU ist eine gute Strategie, die die Anzahl der Seitenfehler nahezu minimiert (Minimum erreicht der nur theoretisch denkbare ‘optimale Algorithmus’), LRU’s Hauptproblem ist die ineffiziente Implementierung, denn LRU braucht entweder ziemlich aufwendige Hardware oder

benötigt unverhältnismäßig viel Laufzeit (Suchen, verschieben und löschen in verknüpften Listen). Es gibt jedoch ein ‘Aging’ genanntes Derivat von LRU das effizient implementierbar ist. Details in Tanenbaum, Moderne Betriebssysteme.
Wie sollten die Seitenrahmen den einzelnen Prozessen zugeordnet werden? (feste oder variable Anzahl je Prozeß, wonach sollte sich die Zuteilung richten?)
Ziel dabei ist eine niedrige Seitenfehlerrate und eine effiziente Speicherausnutzung.
Den Prozessen werden variable Anzahlen Seiten - auf Verlangen - zugeteilt (‘demand paging’). Jeder Prozeß beginnt mit 0 Seiten - und einem Pagefault
Durch empirische Ermittlung des Working Sets eines Prozesses bei früheren Läufen wird es möglich die Seiten dieses Working Sets schon vor dem Prozeß-Start zu laden und so die Seitenfehlerrate zu Beginn zu reduzieren.
Weiter ist es möglich, die Anzahl der Seitenrahmen die einem Prozeß zur Verfügung stehen, bei jedem Seitenfehler zu erhöhen oder bei längerem Nichtzugriff auf bestimmte Rahmen die Anzahl der Seitenrahmen zu erniedrigen. Dieses Zuordnungsverfahren wird auch Arbeitsmengenstrategie genannt.
Was passiert beim Einsatz einer schnelleren Platte?
Die Seitenfehlerrate steigt, da die Übertragungszeit für Seiten sinkt. Die Anzahl der gleichzeitig ausführbaren Prozesse steigt, da jeder Prozeß nun mit weniger Seitenrahmen - und einer höheren Fehlerrate - ‘leben’ kann.
Welche Problematik tritt bei der Arbeitsmengenstrategie auf?
Seitenauslagerung durch Herausfallen der Seite aus dem Working Set; Seiten können aus dem Working Set ausgelagert werden, ohne das ein Seitenfehler aufgetreten ist und ohne das für diese Seite eine neue eingelagert worden wäre. Die aktuelle Lokalität wird eben nur approximiert. Die Arbeitsmenge enthält möglicherweise auch Seiten die nur einmal benutzt worden sind; der Übergang zu einer neuen Lokalität erfolgt eben nur allmählich.
Welche Problematik tritt beim Beobachten der Seitenfehlerrate (bei der Seitenfehlerhäufigkeitsstrategie) auf?
Eine Reaktion kann erst nach dem Auftreten eines Schadens (des Seitenfehlers) erfolgen.
Wie funktionieren dynamische (globale) Seitenaustauschalgorithmen?
Arbeitsmengenstrategie, Seitenfehlerhäufigkeitsstrategie (PPF page fault frequency) ansonsten s.o.
Welche Gefahr besteht bei Seitenaustauschverfahren?
Gefahr des Seitenflatterns (Trashing). Bei Überlastung mit zu vielen Prozessen die entweder zu viele Pagefaults produzieren und/oder zuwenig Seitenrahmen zur Verfügung haben, ist der Rechner weitgehend mit dem Ein-/und Auslagern von Seiten beschäftigt und kann an den eigentlichen Aufgaben der Prozesse kaum noch weiterarbeiten. Effektive CPU Auslastung sinkt durch viele Seitenfehler. Weitere Gefahr besteht dann durch zu einfach konstruierte Scheduler, die bei sinkender CPU-Auslastung mit einer höheren Prozeßintensität reagieren.
Was kann man gegen Trashing tun?
Scheduling einsetzen: Dadurch wird die Anzahl der gleichzeitig aktiven Prozesse verringert und jedem Prozeß stehen mehr Seitenrahmen zur Verfügung.
Wie läuft bei statischer seitenorientierter Speicherverwaltung (besser: Seitenauslagerungsstrategie, denn die Verteilung der Pageframes ist ja statisch) der Seitenaustausch?
Es wurden drei Strategien behandelt:
Optimale Strategie
LRU
FIFO
Kann es dabei zu Trashing kommen?
Ja. Z.B. wenn der verfügbare Speicher gleichmäßig auf eine Menge von Prozessen mit

großem Speicherbedarf verteilt wird und die zugeteilte Menge für jeden Prozeß zu klein ist.
Was tut man dagegen?
Man legt einen der Prozesse schlafen und verteilt seine Seitenrahmen auf die übrigen Prozesse.
Kann es bei der Arbeitsmengenstrategie zu Trashing kommen?
Nein, denn es sind ja jedem Prozeß ausreichend viele Seitenrahmen zur Verfügung gestellt worden. Sind keine freien Seitenrahmen mehr vorhanden, dann wird ein Prozeß ausgelagert und seine Seitenrahmen werden zur Verfügung gestellt.
Ausnahme: Ein einziger Prozeß ist noch aktiv und die Arbeitsmenge dieses Prozesses wird größer als der verfügbare Hauptspeicher.
Welche dynamischen Seitenaustauschverfahren gibt es?
Arbeitsmengenstrategie
Kontrolle der Seitenfehlerrate
Wodurch wird die Größe der Arbeitsmenge bestimmt?
Alle Seiten die in 'letzter Zeit' (<100ms) benutzt wurden, gehören zur Arbeitsmenge. 'In letzter Zeit' wird auch als Fenster bezeichnet. Die Bestimmung des Zugriffes läuft über Zugriffsbits.
Ist beim Arbeitsmengenverfahren Trashing möglich?
Szenario: Ein Prozeß bläht sich auf, verdrängt alle anderen Prozesse aus dem Hauptspeicher. Kann dann seine Arbeitsmenge weiter wachsen?
Erst dann wenn die Arbeitsmenge des verbliebenen Prozesses so groß wird, wie der physische Speicher kann es zum Trashing kommen.

KE5 - Prozeßkommunikation

Warum schaltet ein Prozeß der auf gemeinsame Daten/Resourcen zugreifen möchte, nicht einfach alle Interrupts ab?
Führt zu keinem optimalen Scheduling, weil dann jeder Prozeß solange die Kontrolle behalten darf wie er will. D.h. die wartenden Prozesse sind auf die Kooperation des gerade laufenden Prozesses angewiesen: Daher kooperatives Multitasking (z.B. MS-Windows 3.XX).
Was sind disjunkte und was sind überlappende Prozesse?
Zwei Prozesse sind disjunkt, wenn sie nicht auf gemeinsame Daten zugreifen. Dies gilt auch dann, wenn sie zwar gemeinsam auf denselben Daten arbeiten, diese aber nur lesend nutzen. Man spricht erst dann von überlappenden Prozessen, wenn einer der Prozesse die gemeinsamen Daten auch schreibend verändert. Bei DBS heißen solche Transaktionen 'in konfliktstehend'.
Wie kann der Zugriff auf gemeinsame Daten realisiert werden?
Unter konkurrenten Prozessen versteht man simultane oder parallele Aktivitäten. Überlappende Prozesse sind solche mit gemeinsam genutzten Daten. In einem kritischen Abschnitt eines Prozesses werden gemeinsam genutzte Ressourcen benutzt und verändert. Nicht gemeinsam benutzbare Betriebsmittel (z.B. periphere Geräte, veränderliche Daten) können von konkurrenten Prozessen nur in kritischen Abschnitten belegt werden um exklusive Ausführung und damit gegenseitigen Ausschluß zu gewährleisten. Man erreicht dies durch Prozeßsynchronisation. Wichtig ist hierbei die Vermeidung von zyklischem Warten (Deadlock).
Wie können Prozesse miteinander kommunizieren?
Durch Zugriff auf gemeinsame Daten, z.B. gemeinsame Dateien, Pufferspeicher oder shared Memory, Austausch von Nachrichten über Mailboxen, Rendevouz-Konzept (in ADA).
Wie kann man Prozesse synchronisieren?
Semaphor (globale Synchronisationsvariable), Monitor(=Lock-Manager in DBMS), Nachrichtenaustausch

Diese Betriebsmittel zur Synchronisation sind äquivalent.
Mechanismen des wechselseitigen Ausschlusses?
Semaphore, atomare Operationen p(s) und v(s); Nachrichtenaustausch (Kommunikation zweier Prozesse über gemeinsamen Puffer oder Briefkasten); Monitore (Prozeduren oder Datenstrukturen, die nur von einem Prozeß zu einer Zeit benutzt werden können, analog zu einem Raum, für den es nur einen Schlüssel gibt); auf höherer Ebene Transaktionen.
Funktionsweise der Semaphore?
Semaphore sind ganzzahlige, nichtnegative globale Variable verbunden mit einer Warteschlange w(s). (E.W. Dijkstra, 1968)
Es gibt zwei atomare Operationen:
P(s): if s=0 then wait; s:=s-1; (hol. passeer = hineingehen)
V(s): s:=s+1; if Wartelliste nicht leer then wecke nächsten Prozeß; (hol. verlaat = verlassen)
Die Operation P wird von einem kritischen Prozeß ausgeführt, der einen kritischen Abschnitt betreten möchte. Ist gerade ein anderer Prozeß in dem gemeinsamen Abschnitt dann ist s=0 und es wird gewartet. Der Initialwert des Semaphors s bestimmt, wieviele Prozesse sich gleichzeitig in dem kritischen Abschnitt aufhalten dürfen. Beim Verlassen des kritischen Abschnittes wird s durch die Operation V wieder auf 1 gesetzt bzw. inkrementiert. Die Synchronisation mehrerer Ereignisse kann durch Listen von Semaphoren bewerkstelligt werden: $P(s_1,.....,s_n)$, $V(s_1,.....,s_n)$.
Beschreiben Sie ein praktisches Anwendungsbeispiel von Semaphoren wo deren Anzahl bei 5 oder 6 liegt?!
Bei einer Lösung des Philosophenproblems mit Semaphoren beträgt deren Anzahl beispielsweise 5.
Was ist das Erzeuger-Verbraucher-Problem?
Var inhalt:semaphor; inhalt:=0;
Erzeuger Prozeß: repeat ErzeugeWare; BringeWareNachPuffer; V(inhalt); until(false);
Verbraucher Prozeß: repeat P(inhalt); HoleWareAusPuffer; VerbraucheWare; until false;
Dies ist das einfachste Beispiel, der Schutz der Operation Bringe/HoleNach/AusPuffer fehlt!
Was versteht man unter Scheduling?
Unter Scheduling versteht man das Koordinieren von konkurrenten Prozessen, das eine Maximierung der Auslastung und eine Minimierung der Antwortzeiten zum Ziele hat.
Scheduling ohne Preemption
FCFS (First Come First Served)
SJN (Shortest Job Next)
HRN (Highest Response Ratio Next; Antwortzeit/Bedienzeit Verhältnis)
Scheduling mit Preemption
Round Robin
LCFS (Last Come First Served)
DPRR (Dynamic Priority Round Robin)
Adaptive Prozessorzuteilung
Erster adaptiver Zuteilungsalgorithmus
(dynamische Änderung der Prozessorzuteilungspriorität)
Zweiter adaptiver Zuteilungsalgorithmus
(dynamische Änderung der der Prozessorzuteilungspriorität und dynamische Änderung der Zeitscheiben))
Welche Zuteilungsstrategie sollte man wählen, wenn Bedienzeiten vorher bekannt sind und man die mittlere Antwortzeit minimieren will?
Shortest Job Next (SJN). Nachteil: Aufträge mit langen Bedienzeiten ‘verhungern’, dies wiederspricht dem Implementationsziel ‘Fairness’.
Warum läßt man nicht einfach alle Prozesse nacheinander vollständig ablaufen?

Das sollte man im Batchbetrieb so machen, wo die Verweilzeit zu minimieren ist. Im Timesharingbetrieb muß nicht nur der Prozessor ausgelastet werden, sondern es kommt vor allem auf gleichmäßig kurze Antwortzeiten für alle Benutzer an. Dies wird mittels Mehrprogrammbetrieb erreicht.
Welche Ziele verfolgt man mit einer Prozessor-Zuteilungsstrategie?
Fairneß ‘jeder kommt mal dran’
Effizienz Prozessor ist immer ausgelastet
Antwortzeit Antwortzeiten für interaktive Benutzer werden minimiert
Verweilzeit Wartezeit für die Ausgabe von Stapelaufträgen wird minimiert.
Durchsatz wird maximiert
Welche Strategie ist bei Batch-Verabeitung sinnvoll?
Die non-preemtive. Begründung:
Man läßt einfach jeden Batch-Prozeß solange laufen bis er fertig ist (run-to-completion). Vorteil: Kein Overhead durch Prozeßverwaltung/umschaltung, einfach zu implementieren. CPU-Auslastung immer gleichmäßig hoch. Kann man machen, weil beim Batch-Betrieb nicht auf Antwortzeiten optimiert werden muß.
Was bedeutet ‘Deadlock’ und wann liegt ein Deadlock vor (bezogen auf Betriebssysteme)?
Unter Deadlock versteht man eine Systemverklemmung (zyklisches Warten). Beispiel Buchausleihe: Zwei Entleiher E1 und E2 benötigen jeweils identische Bücher A und B. E1 leiht zuerst A und E2 leiht zuerst B. Jetzt wartet E1 auf die Rückgabe von B und E2 auf die von A => Deadlock.
Folgende allgemeine Bedingungen müssen für das Zustandekommen eines Deadlocks erfüllt werden:
d(1) Bedingung des gegenseitigen Ausschlusses, Prozesse erhalten exklusive Kontrolle über Betriebsmittel
d(2) Bedingung der Nichtunterbrechbarkeit (non preemtive), Betriebsmittel können nicht temporär zurückgegeben werden.
d(3) Warte-Bedingung: Prozesse belegen exklusiv Betriebsmittel während sie noch auf weitere benötigte Betriebsmittel warten.
d(4) Bedingung der geschlossenen Kette (circular wait), jeder Prozeß belegt mindestens ein Betriebsmittel das der nächste Prozeß in der Kette benötigt. (Schlingen im Wartegraph)
Welche Prozesse sind an einem Deadlock beteiligt?
Prozesse die mehrere Betriebsmittel mit jeweils exklusiven Zugriff benötigen.
Wie kann man Deadlocks (im Voraus) erkennen?
Exakt: Es muß die Unmöglichkeit eines kompletten Ablaufes aller im System befindlichen Prozesse gezeigt werden. Dies ist einerseits ein sehr komplexes Problem und andererseits ist das zukünftige Verhalten der Prozesse meist unbekannt.
Zur Vereinfachung des Problems, kann jeder Prozeß vorab seine maximale Anforderung von Betriebsmitteln bekannt geben, der Prozeß führt seinen nächsten Schritt nur dann aus, wenn alle zukünftig benötigten Betriebsmittel zur Verfügung stehen (Preclaiming bei DBMS).
Algorithmisch wird dies durch zyklische Prüfung (für jeden Prozeß) auf Vorhandensein der erforderlichen Betriebsmittel erreicht,
Wie werden Deadlocks beseitigt?
Abbruch der beteiligten Prozesse und Freigabe der belegten Betriebsmittel (entspricht den optimistischen Sperrverfahren bei DBMS).
Wie werden Deadlocks vermieden?
Exakte Lösung: Bei jeder Betriebsmittelanforderung wird geprüft, ob diese zu einem Deadlock führt. Nachteil: sehr aufwendig.
Methode 2: Bei Anforderung zusätzlicher Betriebsmittel werden zunächst alle bisher reservierten Betriebsmittel freigegeben und dann zusammen mit den zusätzlichen Be-

triebsmitteln erneut angefordert (Form von Preclaiming).
Methode 3 Betriebsmitteltypen werden linear angefordert (nach Priorität); wenn schon Betriebsmittel reserviert sind, können keine Betriebsmittel, die wichtiger sind als die schon reservierten, angefordert werden. Damit wird zyklisches Warten unmöglich (ähnelt dem 2-Phasen-Sperrprotokoll).
Problem: Vergabe geeigneter Numerierungen, da reale und abstrakte Betriebsmittel (z.B. Spoolbereiche auf Festplatten)
Methode 4: Banker Algorithmus: Ein Verfahren zu Erkennung sicherer Systemzustände bei der Verteilung von Ressourcen. Idee: Der Banker hat einen bestimmte Menge einer Ressource. Jeder Kunde hat ein Limit, bis zu dem er Ressourcen vom Banker bekommt. Der Banker hat aber so viele Ressourcen, daß er das größte vorhandene Limit bedienen kann. Der Kunde bekommt die Ressource, falls der Banker danach noch genügend Ressourcen hat, um mindestens einem der Kunden sein komplettes Limit zuteilen zu können.
Beispiel dazu: Der Banker habe 10 Einheiten einer Ressource zur Verfügung. Das Limit von Kunde A betrage 8 Einheiten; das von Kunde B betrage 7 Einheiten. Aktuelle habe A 5 Einheiten belegt, B hat 2. Falls B nun eine weitere Einheit anfordert dann wird dies verweigert, dann hat der Banker nur noch 10-5-2-1=2 Einheiten übrig. Damit kann er keinem seiner beiden Kunden mehr sein Limit zuteilen und falls beide im nächsten Anforderungsschritt ihr Limit haben möchten müssen beide Prozesse warten => Deadlock. Solch ein Zustand heißt unsicher.
Würde A noch eine Einheit fordern wäre das oK, da der Banker mit zwei Einheiten die Maximalanforderung von A noch befriedigen könnte. A kann dann sein Vorhaben irgendwann erledigen und seine Schulden zurückzahlen, so daß der Banker dann wieder genug Einheiten hat, um jemand anderem weitere Ressourcen zuzuteilen. Dieser Zustand heißt sicher.
Schlußfolgerung bezogen auf Betriebssysteme: Jeder Prozeß muß im Voraus sagen, wieviele Einheiten eines Betriebsmittels er maximal braucht.
Warum funktioniert die Deadlock-Vermeidung bei linearer Ordnung (beispielsweise durch eine Numerierung) der Betriebsmittel?
(ähnelt dem Zeitstempelverfahren bei DBS)
Dabei wird nach der Regel verfahren: Die Prozesse können alle Betriebsmittel anfordern, aber alle Anforderungen müssen gemäß der Numerierungsreihenfolge geschehen. Somit ist es ausgeschlossen, daß ein Prozeß der ein Betriebsmittel höherer Ordnung besitzt, ein Betriebsmittel niedrigerer Ordnung, das von einem anderen Prozeß belegt ist, anfordern kann. Also werden Schlingen im Wartegraph und damit Deadlocks vermieden (d4 fordert notwendig solche Schlingen!).
Was versteht man unter einem Monitor und wie funktioniert er?
Ein Monitor besteht aus einer Menge von Prozeduren, Variablen und Datenstrukturen die zu einer besonderen Art von Modul oder Paket zusammengefaßt sind. Dieses Monitormodul verkapselt Semaphore und die Operationen auf diesen Semaphoren. Nützlichste Eigenschaft eines Monitors ist, daß jeweils nur ein Prozeß zu einer Zeit einen Monitor benutzen kann.
Ein Monitor ist - als Betriebsmittel betrachtet - vielen Prozessen zugänglich, aber nur jeweils ein Prozeß kann den Monitor zu einem gegebenen Zeitpunkt benutzen. Monitore werden häufig zur Verwaltung von Puffern und Geräten eingesetzt.
Nennen Sie die Mechanismen des wechselseitigen Ausschlusses (mutual exclusion)?
- Semaphore
- Monitore
- Signale (Nachrichten)
- Ereigniszähler (im Kurs nicht behandelt)

Dies sind Semaphore die nur inkrementiert und nie dekrementiert werden.

KE6 - Sicherheit

Welche Sicherheitsfunktionsbereiche gibt es?
Identifikation (User-ID) und Authentisierung (Paßwort): login-Prozedur
Zugriffskontrollen: Rechteverwaltung und Rechteprüfung; least privileg principle: jeder Benutzer sollte nur Rechte haben, die er unbedingt braucht; Dateien: r,w,x-Permission für user/group/others
Beweissicherung/Audit: Protokollieren sicherheitsrelevanter Ereignisse
Initialisierung von Speicherbereichen mit ‘sinnlosem’ Inhalt bevor dieser Bereich dem nächsten Prozeß zugewiesen wird.
Gewährleistung der Funktionsfähigkeit: Verhinderung eines provozierten Absturzes oder einer provozierten Fehlfunktion (Robustheit, Stabilität)
Sicherung von Datenübertragungen: Identifikation,/Authentisierung, Zugriffskontrolle, Sicherung von Vertraulichkeit und Integrität der Daten.
Welche Arten von Zugriffskontrolle gibt es?
diskretionäre Zugriffskontrolle: Besitzer eines Objektes entscheiden ob und wie andere Subjekte auf dieses Objekt zugreifen dürfen => Vertraulichkeit und Integrität der Information kann nicht garantiert werden, weil jedes Subjekt seine Objekte öffentlich zugänglich machen kann. Es gibt folgende Zugriffsmodi: read, write, append, execute, navigate, delete, owner. Der Rechtezustand wird durch eine Zugriffskontrollmatrix (je Zeile ein Subjekt, je Spalte ein Objekt) repräsentiert.
granulatorientiert (in diesem Zusammenhang eigentlich objektorientiert): Rechte werden beim Objekt gespeichert (z.B. als Zugriffskontrolllisten ACL, Schutzbits bei UNIX entsprechend einer ACL mit drei Subjekten,
für persistenten Rechtezustand)
subjektorientiert: Rechte werden beim Subjekt gespeichert (z.B. Profile, Capabilities (Ticket für das Objekt), für transienten Rechtezustand)
informationsflußorientierte Zugriffskontrolle: Zugriffsbeschränkungen werden von den von einem Prozeß gelesenen Daten auf die erzeugten Daten vererbt. Label: jedes Objekt/Subjekt wird genau einer Vertraulichkeits- und genau einer Integritätsklasse zugeordnet; einfache Geheimhaltungsbedingung: Prozeß darf Objekt nur lesen, wenn die Vertraulichkeitsklasse des Prozesses dominiert; Ruhe-Prinzip: Ein Prozeß kann die Vertraulichkeitsklasse eines Objektes nicht verändern; Kommunikationsregel: Prozeß P1 darf P2 nur dann Daten senden wenn die Vertraulichkeitsklasse von P2 die von P1 dominiert.
KE7 - Kommadosprachen