Author: woe007, 18.03.2024

Fragensammlung alle bisherigen schriftlichen Prüfungen VU Betriebssysteme (Pusch) [alle VOWI verfügbaren, bis 2023-02]

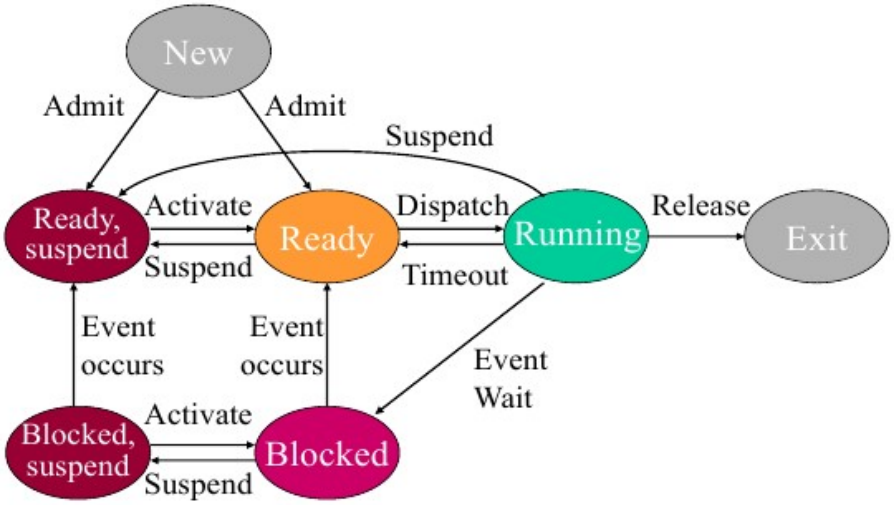
Foliensatz 1 – Betriebssysteme, eine Einführung

Nennen Sie zwei Abstraktionen, die ein Betriebssystem dem Benutzer bietet. Welche Aufgaben führt das Betriebssystem durch, um diese Abstraktionen zu realisieren?

- „ungestörte" Programmbearbeitung → Prozessmanagement
- „unendlich großer Speicher" → Speicherverwaltung durch Betriebssystem
- „private Maschine" → Betriebssystem ist für Zugriffsschutz und Datensicherheit zuständig

Foliensatz 2 – Prozesse

Zeichnen Sie das Zustandsdiagramm eines Prozesses, beginnend mit der Entstehung bis zur Beendigung des Prozesses. Geben Sie die Namen der Zustände an und beschreiben Sie kurz jeden Zustand und Übergang.



Zustände:

- **New**: OS hat einen neuen Prozess erstellt, der Prozess ist jedoch noch nicht bereit zur Ausführung
- **Ready**: bereit zur Ausführung, ist in Ready-Queue, wartet auf Zuteilung durch CPU
- **Running**: Prozess läuft gerade auf CPU
- **Blocked**: Prozess wartet auf Event (bspw. I/O-Operation), sodass er weiter arbeiten kann.
- **Ready, suspend**: wie Ready, nur ausgelagert durch Swapping in einem Sekundärspeicher
- **Blocked, suspend**: wie Blocked, nur ausgelagert durch Swapping in einem Sekundärspeicher
- **Exit**: Zustand wird durch Terminierung erreicht, Prozessinformationen/Tabellen werden gelöscht sobald nicht mehr benötigt

Übergänge:

- **Admit**: Prozess aus New ist nun bereit zur Ausführung
- **Suspend**: Auslagern des Prozesses in Sekundärspeicher (zur Effizienteren Nutzung des Speicherplatzes)
- **Activate**: Reintegrieren in Primärspeicher (entweder Blocked- oder Ready-Zustand)
- **Dispatch**: Abrufen des Prozesses vom Ready-Zustand in den Running-Zustand (CPU)

Author: woec0071, 18.03.2024

- Timeout: bezeichnet einen Interrupt, Dispatcher holt gemäß Scheduling Strategie einen neuen Prozess rein und muss daher den derzeit (Bereiten) Prozess in die Ready Queue stecken.
- Event wait: falls Prozess bspw. Auf I/O Operation wartet, wird er geblocked um in dieser Zeit die CPU anderweitig zu nutzen.
- Event occurs: ein bestimmtes Event (I/O Operation) fand statt, Prozess ist nun wieder ready.

Wodurch unterscheiden sich die Prozesszustände Ready und Blocked?

- Ready: jederzeit bereit um wieder vom Dispatcher ausgewählt zu werden und auf CPU zu arbeiten
- Blocked: wartet auf bestimmtes Event (bspw. I/O Operation, um weiter ausgeführt werden zu können)

Was ist Swapping? Wann wird es angewandt?

- Swapping ist das Auslagern von Ready/Blocked Zuständen inkl. dazugehöriger gleichartigen Zustände auf Sekundärspeicher (Festplatte) (Ready, suspend / Blocked, suspend)
- wird angewandt, wann zu viele Prozesse im Hauptspeicher (RAM) sind, dies führt zu einer Verschlechterung der Performance

Was versteht man unter einem Process Control Block? Beschreiben Sie, aus welchen Teilen der PCB besteht und welche Informationen in diesen Teilen jeweils verwaltet werden.

Process Control Block ist Teil des Process Images, PCB enthält Daten, die das OS benutzt um den Prozess zu verwalten

- Process Identification (ID): PID des Prozesses, ID des Users und ggf. ID des Parent-Prozesses
- Processor State Information: Zustand des Prozessors, speichert Register Inhalte, Kontroll- und Statusregister, und Stackpointer
- Process Control Information: Zustand des Prozesses, speichert Scheduling und Prioritätsinformationen, Queueverweise auf andere Prozesse, Interprozesskommunikation, Memorymanagement, welche Ressourcen wurden/werden verwendet, Privilegien

Geben Sie drei HW-Mechanismen an, mit denen moderne Mikroprozessoren die Aufgaben des OS unterstützen. Charakterisieren sie die Funktion jedes Mechanismus kurz.

- Process Switching: Wechsel von Prozessen gemäß Scheduling Strategy
- I/O und Hardware Access Management: zugehörige HW mittels Interfaces und I/O
- Basic Memory Management: grundlegende Unterstützung bei der Speicherverwaltung

Erklären Sie die Aktionen, die vom Betriebssystem bei einem Process Switch durchzuführen sind. Welche Arten von Ereignissen können zu einem Process Switch führen?

Process Switch ist der Wechsel des aktiven Prozesses, findet statt wenn OS im Besitz der CPU ist, entsteht durch System Call (intern wird Funktionalität des OS aufgerufen) + Trap (interner Fehler im Prozessanweisung) + Interrupt (externe Störung)

- PCB muss gespeichert werden
- CPU Register und Stackeinträge werden gespeichert, sodass Prozess an genau dieser Stelle weiterarbeiten kann.
- Durchführen der Interrupt/Call/Trap Aktion
- Umschalten des aktiven Prozesses gemäß Scheduling
- später, wenn Prozess wieder geladen werden soll, wird alter Prozess- und Prozessorzustand wieder hergestellt

Author: woe007, 18.03.2024

Erklären Sie die Begriffe Process Switch und Mode Switch, wobei Sie die Beziehung, die diese beiden Konzepte zueinander haben, verdeutlichen. Zählen Sie weiterhin die Klassen von Ereignissen auf, die einen Mode Switch nach sich ziehen.

Process Switch ist der Wechsel des aktiven Prozesses, findet statt wenn OS in Besitz der CPU ist
entsteht durch System Call (interne Funktion) des OS aufgerufen (Trap (interner Fehler im Prozessablauf) → Interrupt (externe Störung)

Mode Switch (wechselt eben durch System Call, Trap oder Interrupt) erzeugt Wechsel vom User-Mode in einen privilegierten Mode. Im User-Mode sind keine Process Switches möglich, insofern bedingt Mode Switches (nicht unbedingt, aber sie können) Process Switches → Execution Mode Wechsel (Execution Modes existieren zum Schutz der Datenstrukturen des Betriebssystems)

Worin liegt der grundsätzliche Unterschied zwischen Prozessen und Threads? Welchen Vorteil ergibt sich aus der Einführung von Threads für den Benutzer und worauf muss der Benutzer achten?

Prozesse können sich schwer um Ressourcenverwaltung, als auch Dispatching (kurzfristiges Scheduling) → mit der Einführung von Threads entkoppelt man diese beiden Aufgaben, Threads kümmern sich nur um das Dispatching.

- Prozesse: virtueller Adressraum mit Prozessspeicher, Dateien, I/O, Files
- Threads: derzeitiger Ausführungszustand, Stack, Kontext, thread-lokale Variablen, besitzt Zugriff auf Prozessspeicher und Ressourcen

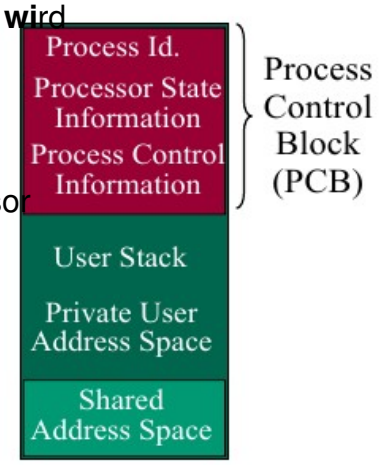
Vorteile:

- Thread-Erzeugung/Terminierung benötigt weniger Zeit,
- Umschaltung zwischen Threads geht schneller als Prozessswitch
- Kommunikation ohne Kernel
- Benutzer muss beachten: Synchronisation zwischen Threads bei User-Level-Threads

Was versteht man unter einem Prozess? Erklären Sie, aus welchen Teilen ein Prozess Image besteht.

Ein Prozess ist ein Programm in Ausführung, wobei jeder Prozess durch einen Prozessdeskriptor beschrieben wird, der alle Informationen über den derzeitigen Status/Aufbau des Prozesses, wird vom BS benötigt zur Speicherverwaltung im RAM

- Process Tables im BS zeigen auf Process Images
- im Virtual Memory
- Bestehen aus:
  - Process Control Block (Process Identification, Processor state information, Process control information)
  - User Program: enthält alle veränderlichen Information (Programmcode, etc.)
  - User Data (System Stack, wo Variablen gespeichert werden, und Heap zur variablen Allokierung)
  - System Stack: speichert Parameter und Adressen der System Calls [Shared Address Space]



Was versteht man unter einem Microkernel? Welche Services müssen ein Microkernel zur Verfügung stellen? Welche Vor- und Nachteile ergeben sich bei der Verwendung eines Microkernels

- Nur die nötigsten Basisservices befinden sich im Kernel, nicht zentrale Services des Betriebssystems als Server-Prozesse auf Prozessebene.

Author: woe007, 18.03.2024

- Vorteil: flexibel, einheitliche Portabel
- Nachteil: Micro (nicht definiert, unterschiedliche Anzahl an Services)
- Zentrale Service Prozesse, Memory Management, Hardware Access und Nachrichtenaustausch bzw. Kontrolle zwischen Prozessen und Kernel

Erklären Sie den Begriff Multithreading. Gehen Sie auf Probleme bei der Verwendung von Threads an.

- Multithreading bezeichnet, dass pro Prozess mehrere Threads existieren. Dies bedeutet, dass jeder Prozess mehrere Aufgaben (Threads) unterteilt werden kann, während der Prozess selbst nur noch das Ressourcenmanagement betrei
- Problem: Zugriff auf gemeinsam genutzte Ressourcen
- Thread-Control Block je Thread, besitzt Kontrolle über Thread

Nennen Sie die drei Kategorien von Ereignissen, mit denen das Betriebssystem die Kontrolle über das Computersystem übernimmt. Geben Sie für jede der Kategorien ein Beispiel an.

- System Call: explizite Aufforderung des Programms an das Betriebssystem (I/O, Fork, etc.)
- Trap: bedingt durch aktuelle Programminstruktion (Fehlerhafter Code, etc)
- Interrupt: Ursache liegt außerhalb des Prozesses

Welcher Vorteil ergibt sich aus der Einführung von Threads für den Benutzer? Wodurch kommt es zu diesem Vorteil? Worauf muss der Benutzer bei der Programmierung von Threads achten?
Vorteile:

- Thread-Erzeugung sowie Thread-Terminierung benötigt weniger Zeit, dies kommt daher, da Threads die Ressourcenverwaltung des Prozesses übernehmen, Threads kümmern sich nur um das Dispatching (Prozesse übernehmen beides, daher langsamer)
- Umschalten zwischen Threads geht schneller, kein Process Switch nötig
- Kommunikation zwischen Threads ist ohne Kernel möglich, jedoch ist eine Synchronisierung zwischen den Threads notwendig

Der Benutzer muss bei der Programmierung von User Level Threads darauf achten, dass die Synchronisation in seinen Händen liegt.

Was versteht man unter einem Kernel Level Thread (KLT) und unter einem User Level Thread (ULT)? Beschreiben Sie die beiden Arten der Threadimplementierung und charakterisieren Sie jeden Unterschied.

- Kernel Level Thread:
  - werden vom Kernel gesehen und verwaltet mittels Kernel Thread API
  - Threadwechsel durch Kernel (nur im Kernel Mode möglich)
  - blockieren einzelner Threads möglich, nicht ganzer Prozess wird blockiert.
  - Bei mehreren Kernel KLT kann ein Kern als ULT ausgeführt werden, gleichzeitig!
- User Level Thread:
  - Threads sind für den Kernel unsichtbar
  - Threadswitching/management durch Thread Library im User Mode möglich
  - applikationsspezifisches Scheduling

Author: woe007, 18.03.2024

- wird Prozess vom Kernel geblockt, blockiert auch Thread
- blockierender System Call (I/O), so werden alle Threads des Prozesses gestoppt
- Synchronisation ist im User Mode
- kein Ausführen auf mehreren Kernen gleichzeitig

Wie unterscheidet sich das Blockieren von Kernel Level Threads und User Level Threads?

- **ULT**: Threads werden als ganzes geblockt, sobald Prozess geblockt wird
- **KLT**: einzelne Threads können geblockt werden, nicht ganzer Prozess

Beschreiben Sie die verschiedenen Arten, wie Prozesse vom Betriebssystem getrennt werden können!

- **Nonprocess Kernel**:
  - strikte Trennung von Prozessen und Kernel
  - Prozesse sind nur Benutzerprogramme,
  - BS arbeitet getrennt von Prozessen
  - Kernel ist zentrale Schnittstelle zwischen HW/SW
- **Ausführung des Betriebssystems in User-Prozessen**:
  - BS ist Sammlung von Routinen, die in User Prozessen ausgeführt werden können
  - Verlassen des Prozesses nur zum Prozessswitching
  - jeder Prozess hat eigenen Kernelstack
- **Prozessbasiertes Betriebssystem**:
  - Betriebssystem ist Ansammlung von Prozessen (wie die Prozesse selbst)
  - nur Basisservices sind keine Prozesse, alle anderen BS-Services sind eigenständige Prozesse

Welche Kernelarchitekturen gibt es?

- **Monolithic OS**: nur bei kleinen Betriebssystemen, Menge an Prozeduren die sich gegenseitig aufrufen (verwaltet)
- **Layered OS**: geschichtetes System auf hierarchischer Art, nur benachbarte Schichten können miteinander agieren, Aufbauen
- **Modular OS**: verschiedene Ansätze realisieren die wichtigsten Funktionalitäten vom Betriebssystem, Hardware Abstraction Layer unterste Schicht

Foliensatz 3 – Scheduling

Was versteht man unter Long Term Scheduling, Mid Term Scheduling und Short Term Scheduling?

Long Term Scheduling:

- bei der Kreierung von neuen Prozessen aktiv
- befasst sich mit der Frage, ob ein neuer Prozess in die Ready-Queue oder in die Ready-Suspend-Queue kommt, also ob der Prozess an dem Mid-term Scheduler oder den Short-Term Scheduler übergeben wird
- bestimmt Parallelitätsgrad

Mid Term Scheduling:

- Scheduling für das Ein- und Auslagern sekundärer Speicher (Swapping)

Short Term Scheduling:

- bestimmt welcher Prozess aus Ready Queue als nächstes CPU zugeteilt wird
- = Dispatcher

Author: woe007, 18.03.2024

- wird bei Interrupt, OS Signal, Call, Signal aufgerufen
- Interrupt kann durch I/O Operation oder durch Timer (max. Zeit zur Abarbeitung) ausgelöst werden

Beschreiben Sie die folgenden Strategien für das CPU-Scheduling und vergleichen Sie deren Eigenschaften: FCFS, Round Robin, SPN, SRT, Highest Response Ratio Next und Feedback Scheduling

First Come First Serve:

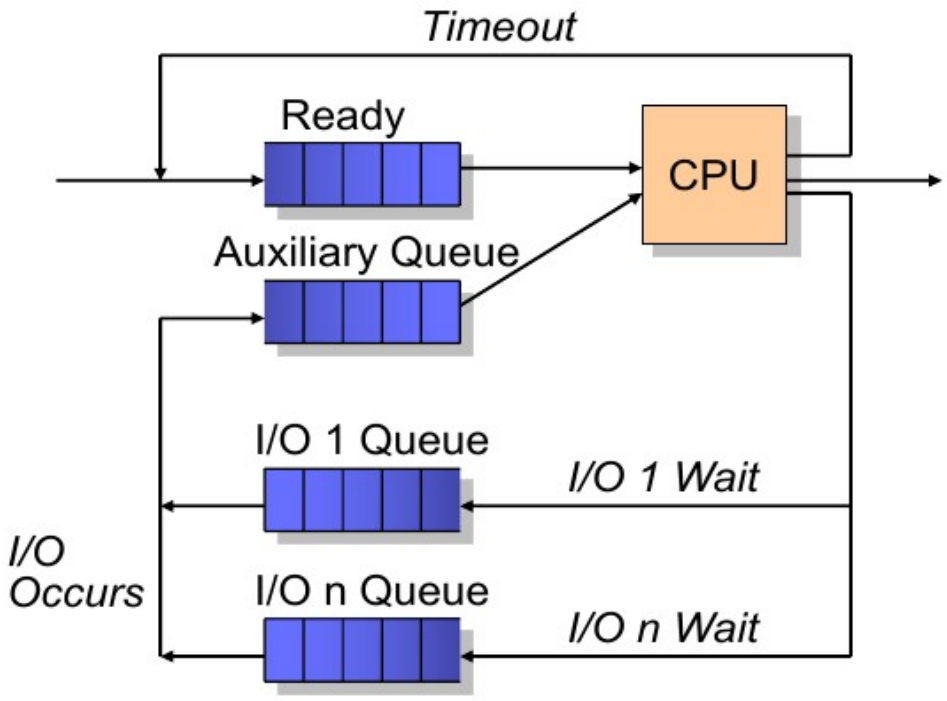
- Selection Function: wer zuerst kommt, ist zuerst dran
- Decision Mode: Non-Preemptive
- begünstigt lange und CPU-intensive Prozesse, während I/O-intensive Prozesse immer wieder ans Ende der Ready-Queue geschoben werden
- ermöglicht Starvation und Endlosschleifen

Round Robin:

- Selection Function: wie bei First Come First Serve, ältester Prozess in Ready Queue wird ausgewählt
- Decision Mode: Preemptive (unterbrechend)
- jeder Prozess bekommt gleich (kurz) Zeit, sobald Zeit abgelaufen → Prozesswechsel (zyklische Weitergabe von Zeitslots)

Virtual Round Robin:

- Round Robin benachteiligt I/O-intensive Prozesse, diese können die Zeitslots nicht voll ausschöpfen, sie werden während Blockierzeit von CPU-intensiven überholt
- daher Virtual Round Robin mit Auxiliary Queue (bessere höhere Priority als Ready Queue)



Shortest Process Next (SPN):

- Selection Function: wählt Prozess mit geringster CPU-Dauer aus

Author: woe007, 18.03.2024

- Decision Mode: Non-Preemptive
- kurze Prozesse werden begünstigt
- bessere Response Time als FCFS
- Problem: Starvation möglich, Berechnung der erwarteten Programmdauer schwierig, non preemptive ermöglicht Blockieren der CPU

Shortest Remaining Time (SRT):

- Selection Function: wie SPN, wählt Prozess mit geringster CPU Dauer aus
- Decision Mode: Preemptive (unterbrechend)
- fairer in Bezug auf kurze Prozesse als Shortest Process Next (SPN)
- Problem: Starvation möglich, Berechnung der erwarteten Programmdauer schwierig

Highest Response Ratio Next:

- Selection Function: $RR = \frac{w + s}{s}$ [w: gesamte bisher gewartete Dauer, s: geschätzte Service Time], Priorität höher je größer RR
- Decision Mode: Non-Preemptive
- fairer in Bezug auf kurze Prozesse
- keine Starvation, jedoch Berechnung der erwarteten Servicezeiten nötigen

Feedback-Scheduling:

- Selection Function: basiert auf bisher konsumierter CPU Time, je mehr ein Prozess konsumiert hat, desto niedriger wird Priorität
- Decision Mode: preemptive
- Problem: Starvation möglich, Abhilfe: mit der Zeit Priority eigene Queue für jede Priority anheben

Erklären Sie die Begriffe Deadlock, Lifelock und Starvation (4) (4)

- Deadlock: durch zyklische Abhängigkeit beim Zugriff auf Ressourcen kann keiner der beiden Prozesse die notwendigen Ressourcen anfordern, aber gibt sie auch nicht ab (no preemption) → Programm kann nicht weiterlaufen
- Lifelock: Prozess wird in einer Endlosschleife gefangen, kein Fortschritt möglich
- Starvation: ein Prozess kann nie auf bestimmte Ressourcen zugreifen, weil immer andere vorher abgearbeitet werden

Nennen Sie die Arten von Optimierungsziele, die beim Scheduling verfolgt werden und geben Sie jeweils ein Beispiel an.

- Durchsatz: desto besser Scheduling, desto mehr Prozesse können nacheinander beendet werden – SRT
- Fairness: alle Programme müssen irgendwann beendet werden, kein Verhungern – Round Robin
- Response Time: möglichst schnelle Antwort z.B. Shortest Process Next (SPN)
- Einhalten von Deadlines: Alle (periodischen) Prozesse müssen spätestens dann beendet, wenn gefragt (EDF)
- Prozessorauslastung: möglichst wenig Leerlauf, immer Arbeit – Virtual Round Robin

Author: woe007, 18.03.2024

	User-Oriented	*System-Oriented*
Performance	Response Time, Turnaround Time, Deadlines	Throughput, Processor Utilization
Other	Predictability	Fairness, Resource Balance, Priorities

Bei welchen der folgenden Scheduling-Strategien kann es zur Starvation kommen (a) FCFS, (b) Shortest Job First, (c) Round Robin, (d) Priority Scheduling? Begründen Sie jeweils Ihre Antwort. (4)

a) Nein, keine Starvation – jedoch sehr lange Wartezeiten möglich

b) Ja, da eventuell (falls immer wieder neue Prozesse kommen) immer kürzere vorhanden sein könnten

c) Nein, da jeder Prozess ein zyklisches Zeitscheibe erhält – lästige Prozesse sind jedoch sehr benachteiligt dann RR

d) Ja

Was versteht man unter Real-Time Scheduling?

- **Echtzeit-Scheduling bezieht sich auf die WEG, also auf die Deadlines:**
  - **Soft Deadline: Verpassen ist nicht kritisch (Temperatursensor)**
  - **Hard Deadline: muss unbedingt eingehalten werden, sonst Katastrophe (FCS, Alto)**
- **üblicherweise Preemptive**
- **oft bei periodischen Prozessen**
- **Schedulability Test: überprüfen, ob Task Set scheduable ist – also alle Prozesse rechtzeitig bedient werden können**

Erklären Sie Earliest Deadline First Scheduling!

- **Ist ein Real-Time Scheduling Verfahren**
- **Selection Function: Task mit frühester Deadline (Deadline abgegeben)**
- **Decision mode: preemptive**
- **Optimierungsziel: Minimierung der verpassten Deadline**
- **Schedulability Test: Sum(C/T) <= 1 ((C: Dauer, T: Periode))**
- **Wird immer geschaut, welche Task nächste Deadline hat – wird ausgeführt, danach wieder geschaut, usw.**

Foliensatz 4 – Mutex & Semaphoren

Was versteht man unter einem Monitor zur Prozesssynchronisation? Nennen Sie die wichtigsten Komponenten und Eigenschaften eines Monitors!

- **Der Monitor ist ein Softwaremodul, bestehend aus Prozeduren, lokalen Daten und Initialisierungscode**

Author: woe007, 18.03.2024

- Prozedur regelt Zugriff durch Warten auf bestimmte Bedingung (not full, not empty, etc.)
- Pro Prozedur kann nur ein Prozess zugreifen
- Eigenschaften:
  - Monitor sorgt für Mutex, muss nicht explizit programmiert werden
  - Shared Memory wird im Monitor angelegt
  - Zugriff auf eine lokale Variable mittels Monitorprozedur
  - Eintritt eines Prozesses in den Monitor mittels Monitorprozeduren
  - max. 1 Prozess gleichzeitig im Monitorprozedur
  - Bedingungssynchronisation über Monitorvariable
  - Bedingungsvariablen lokal im Monitor, zugreifbar (wait, signal), speichert nicht

Für die Lösung des Problems des gegenseitigen Ausschlusses kritischer Abschnitte werden drei Eigenschaften gefordert. (a) Nennen Sie diese Eigenschaften und erklären Sie deren Bedeutung. (b) Wodurch werden die drei Eigenschaften gewährleistet, wenn Semaphore zum Schutz eines kritischen Abschnitts verwendet werden?

a) **Mutual Exclusion** ⇒ nur ein Prozess darf in den kritischen Abschnitt rein
Progress ⇒ jeder Prozess muss irgendwann in den kritischen Abschnitt rein und darf nicht auf ewig verzögert werden (keine Starvation!)
Bounded Waiting ⇒ nach Request für kritischen Abschnitt gibt es nur eine limitierte Anzahl von Personen, die vor einem warten dürfen

b) Semaphore bestehen aus einer Variable und einer Queue. Es wird immer überprüft ob der Wert <= 0 oder > 0 ist, somit wird Mutex bei mehreren Prozessen erzeugt. Falls <= 0, kommt der Prozess in eine (FIFO) Queue, welche die Bedingungen Progress und Bounded Waiting sicherstellt

Gegeben sei ein Computersystem, in dem Ihnen zur Synchronisation bzw. Kommunikation von Prozessen nur Nachrichten zur Verfügung stehen (d.h. es gibt keine Semaphore oder andere Synchronisationskonstrukte). Nennen Sie zwei verschiedene Möglichkeiten, wie Sie in diesem Computersystem ein konsistentes Datenaustauschprotokoll zwischen parallelen Prozessen realisieren können

- blockierendes und nicht blockierendes Message-Passing
- Blockierend: Beim Empfangen oder Senden einer Message wird solange gewartet, bis sichergestellt werden kann, dass der Message-partner alles empfangen hat
- Nicht-Blockierend: Prozess sendet Nachricht und arbeitet weiter, ohne auf Antwort zu warten

Foliensatz 5 – Deadlocks

Welche Strategien zur Vorbeugung gegen Deadlocks bzw. zur Vermeidung von Deadlocks gibt es? Beschreiben Sie diese kurz. (5)

- Deadlock Prevention beschreibt das Verhindern der 4 Deadlockbedingungen
  - No Mutex (indirekt) kann nicht verhindert werden, ist nötig aufgrund unserer Aufgabenstellung
  - Hold & Wait (indirekt) Prozesse fordert alle Ressourcen auf einmal an, Blockieren bis alle da sind → lange Verzögerung, Prozess braucht Wissen über Ressourcen, die er verwenden wird
  -

Author: woe07, 18.03.2024

- No Preemption (indirekt): zugewiesene Ressourcen werden nicht weggenommen, Prozesse geben Ressourcen frei, wenn sie eine nicht bekommen können
- Circular Wait vermeiden (direkt): Protokoll, mit strikter linearer Ordnung bzgl Ressourcenart, in der Folge werden nur Anforderungen zugelassen, die unter der Grenze liegen

- Deadlock Avoidance: Bedingungen 1, 3, 4 gelten, selektives Vergeben von Ressourcen
  - Process Initiation Denial: Prozess wird nicht gestartet, wenn seine Anforderungen zu einem Deadlock führen können (Risiko) (sehr defensiv)
  - Resource Allocation Denial: Ressourcenanforderung wird verwehrt, wenn sie zu Deadlock führen könnte. Bankers Algorithm (ergibt Safe oder Unsafe State)
  - Voraussetzung: Ressourcenbedarf muss bekannt sein
- Deadlock Detection: Ressourcenanforderungen werden gewährt, sofern vorhanden, Algorithmus um Deadlock zu erkennen (ähnlich Bankers), Strategie zur Deadlockbehebung (Recovery), sehr CPU intensiv

Bei der Deadlock-Vermeidung spricht man von einem Safe State bzw. Unsafe State. Erklären Sie die Bedeutung dieser Begriffe.
Bei Deadlock Avoidance, in Resource Allocation Denial, kann man den sog. Bankers Algorithmus, welcher als Ergebnis einen „Safe" oder „Unsafe" State liefert. Safe: es gibt die Möglichkeit alle Prozesse abzuarbeiten ohne Deadlock; Unsafe: Deadlock möglich, aber nicht zwingend nötig

Nennen Sie die Bedingungen für das Entstehen eines Deadlocks und erklären Sie diese.
- No Mutex (indirekt): kann nicht verhindert werden, ist nötig aufgrund unserer Aufgabenstellung
- Hold & Wait (indirekt): Prozess kann Ressourcen halten, während er auf andere wartet
- No Preemption (indirekt): zugewiesene Ressourcen werden nicht weggenommen, bei leicht speicherbaren Ressourcen (Prozess Switch)
- Circular Wait vermeiden (direkt): geschlossene Kette von Prozessen, von der jeder mind. 1 Ressource hält, die der andere Prozess benötigt

Was versteht man unter Deadlock Avoidance? Geben Sie zwei Strategien für Deadlock Avoidance an und beschreiben Sie diese.
Ressourcenvergabe, die zu Deadlocks führen könnten, werden nicht gewährt (Bedingungen 1 bis 3 sind erlaubt, Circular Wait nicht). Höhere Parallelität als Deadlock Prevention, man muss jedoch Wissen bezüglich des Ressourcenbedarfs haben
- Process Initiation Denial: Prozess wird nicht gestartet, wenn seine Anforderungen zu einem Deadlock führen können (Risiko) (sehr defensiv (Claim matrix beschreibt maximalen Ressourcenbedarf eines jeden Prozesses)
- Resource Allocation Denial: Ressourcenanforderung wird verwehrt, wenn sie zu Deadlock führen könnte. Bankers Algorithm (ergibt Safe oder Unsafe State)

Foliensatz 6 – Memory Management

Was versteht man unter Virtual Memory Management? Welche Mechanismen benötigt man zur Realisierung von Virtual Memory Management? Welche Vorteile bietet es? (5)
- Unter Virtual Memory Management versteht man dynamische Adressübersetzung: logische Adressen referenzieren VM, Adressübersetzung bei jeder Ausführung
- Aufteilung des Speichers in Pages/Segmente (nicht zusammenhängend)

Author: woe007, 18.03.2024

- erlaubt es, die nur für den Prozess derzeit relevanten Pages im RAM zu laden
- Segmenttabelle für Überprüfung
- Falls angefordert Page derzeit nicht im RAM (Page Fault) → Laden aus Sekundärspeicher (kann zu Thrashing führen)
- Resident Set: Teile des Prozesses, die gerade im RAM sind
- Vorteil: Prozess kann größer als RAM sein, da nicht alles gleichzeitig geladen werden muss; einfache Adressübersetzung

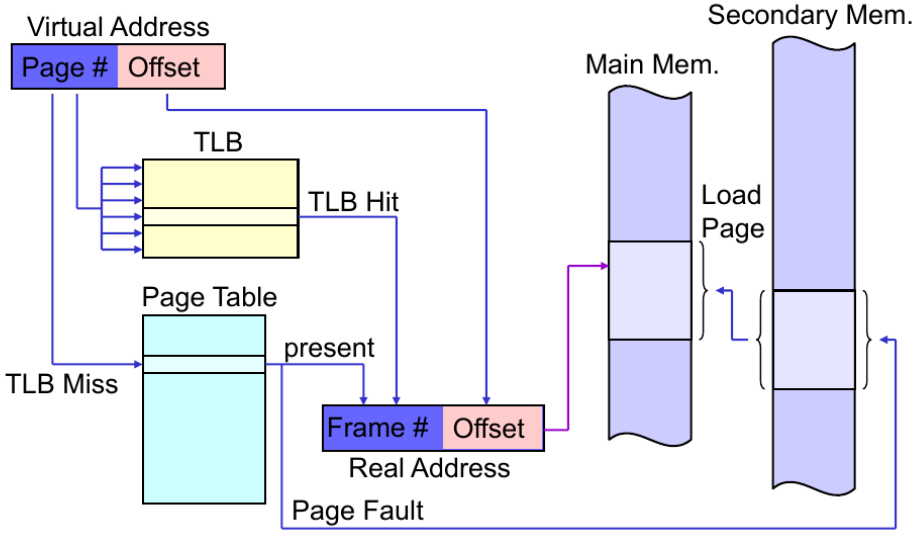
Nennen Sie Möglichkeiten, um in einem Paging-System Speicherzugriffe zu realisieren? (3)

- Bound-Check: Logische Adresse ist kontinuierlich zusammenhängend, man überprüft ob Adresse innerhalb der (maximalen) Länge des Adressbereichs liegt.
- Bei Speicherbereichsschutz mit Protection Keys
- Variante 1: Jeder (physischer) Frame hat einen Key, jeder Prozess hat einen Key, bei Adressierung auf Frame wird Key abgeglichen
- Variante 2: Jeder Prozess hat Menge von Keys, in TLB ist zu jeder logischen Adresse ein Key hinterlegt, wenn der Prozess-Key mit TLB-Key übereinstimmt → Zutritt
- sonst immer Speicherbereichsverletzung

Beschreiben Sie Aufgabe und Funktion eines Translation Lookaside Buffers? Worauf hat man bei der Betriebssystemimplementierung bei einem Prozesswechsel zu achten, wenn man einen Translation Lookaside Buffer verwendet?
Aufbau:

- Cache für Einträge der Seitentabelle (Page, Frame) der zuletzt verwendeten Seiten
- 16-512 Einträge
- assoziativer Zugriff
- Löschen bei jedem Context Switch

Funktion: möchte man die Adressübersetzung von virtuell zu physischer Adresse, schaut man zuerst mit dem ersten Teil der virtuellen Adresse (#Page) in den TLB nach, und sucht nach der passenden Pagenummer. Findet man (TLB Hit), so hat man das assoziative #Frame (Framenummer), und muss nicht mehr in die Page Table schauen. Offset bleibt gleich.



Author: woe007, 18.03.2024

Bei einem Process Switch muss die TLB gelöscht werden, da sonst beim Anfang des neuen Prozesses s
nicht viele Einträge vorhanden (unwahrscheinlich, TLB Hit zu treffen)

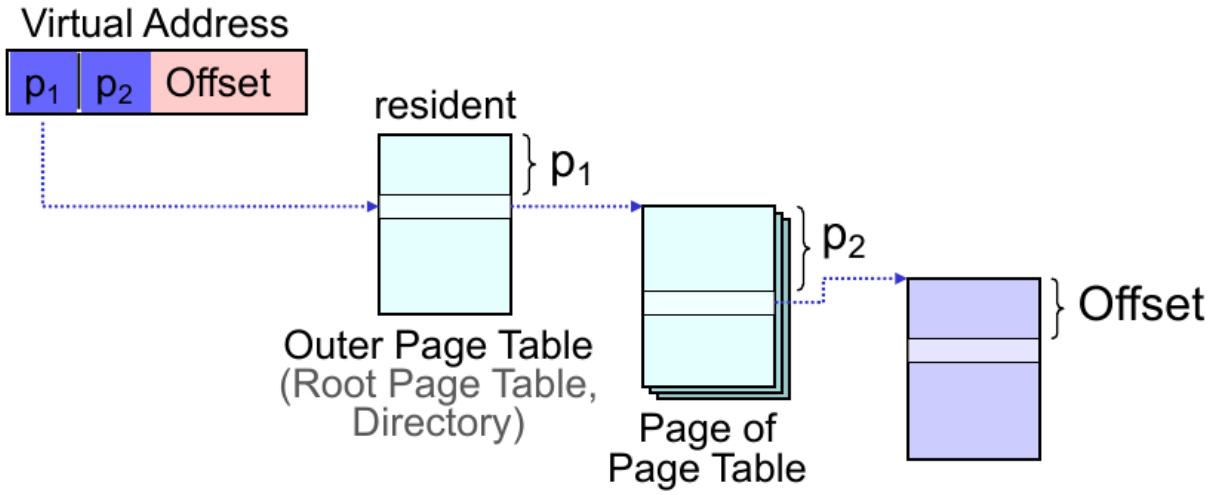
Was versteht man unter dem Begriff Relokation? Wofür ist Relokation von Bedeutung?

- Keine statische Fixierung des Speicherbereichs vom Prozess → Instruktionen müssen an verschiedenen Speicherstellen (dynamisch) positioniert werden können (bsp Swapping)
- Referenzen auf physischen Speicher müssen veränderbar sein
- Konzept: Unterscheidung zwischen physischer (absoluter) und logischer Adresse (Referenz unabhängig von Organisation des Speichers) [Relative Adresse: Adressen von bekanntem Punkt aus]

Wir betrachten ein System mit Virtual Memory Management. Diskutieren Sie, welche der folgenden Situationen bei Referenzierung einer virtuellen Adresse auftreten bzw. nicht auftreten kann: (a) TLB Miss ohne Page Fault, (b) TLB Miss mit Page Fault, (c) TLB Hit ohne Page Fault, (d) TLB Hit mit Page Fault.

a) möglich
b) möglich
c) muss so auftreten
d) kann nicht auftreten, da TLB Hit nur stattfindet, falls physische Adresse gerade aufgerufen wurd
(und somit muss sie derzeit im Speicher sein)

Beschreiben Sie den Aufbau und die Verwendung einer Multi-Level Page Table (evtl. Mit Skizze).
Warum werden Multi-level Page Tables verwendet?



Anwendung große Prozesse, mit großer Page Table, falls Page Table nicht im RAM passt,
wird unterteilt in zwei Page Tables (mit 3 Zugriffe, dauert länger)

Beschreiben Sie, wozu und wie eine Page Table verwendet wird. Gehen Sie weiters an,
welche Information in den Tabelleneinträgen einer Page Table gespeichert werden.
Prozesse → wird in Pages unterteilt
RAM ⇒ in Frames

Author: woe007, 18.03.2024

wird gemacht, da ganze Prozesse oft nicht in den Hauptspeicher passen. Page Table zur Übersetzung zwischen Prozess (Pages) und RAM (Frames) da.

Page Table wird zur Adressübersetzung einer virtuellen Adresse in eine physische Adresse benutzt. Dabei besteht eine virtuelle Adresse (Page#, Offset) aus der Offset bleibt bei der physikalischen Adresse gleich, die Page# wird in der Page Table als Index benutzt. In diesem Index findet man darauf hin die physische Frame-Nummer der Adresse. Falls nicht in der Page Table vorhanden (die Page# kommt es zu einem Page Fault und die Seite muss aus einem Sekundärspeicher in den RAM geholt werden (Page Replacement).

Wozu wird die Clock Policy verwendet? Beschreiben Sie die Funktionsweise.

- Ist eine Page Replacement Strategie
- wird also angewandt, um möglichst wenig Page Faults im Gegensatz zu der Optimalen Pages in den Frames zu haben
- Funktion: Hat einen Zeiger, der auf die nächste zu ersetzende Seite zeigt. Falls eine Seite aufgerufen wird, wird ein Flag auf 1 gesetzt. Es wird immer das ersatzweise auf das der Pointer zeigt. Falls Pointer auf ein zeigt, wo Flag = 1, wird das nächstliegende Flag = 0 ersetzt.
- Ist nicht viel schlechter als die LRU Strategie

Was versteht man unter der Working Set Strategie? Beschreiben Sie die Funktionsweise und erklären Sie, wie diese Strategie zur Optimierung eines Paging-Systems eingesetzt werden kann

- Ist eine Strategie, um möglichst wenig Page Faults zu erzeugen
- Working Set W(t, Δ) blickt zu einem Zeitpunkt t Δ Zeiteinheiten zurück, und speichert (variabel in der Anzahl) die in dieser Zeit benutzten Pages ab
- basiert auf der Lokalitätsprinzip
- wächst schnell an, stabilisiert sich & wächst wieder schnell bei einem Prozess Switches
- Resident Set soll dann immer zu bestimmten Zeitpunkten sich am Working Set orientieren
- Problem: optimales Δ unbekannt, variiert mitunter sehr schwierig, daher gängige Praxis:
  - Beobachtung der Fehlerhäufigkeit und dementsprechende Anpassung der Anzahl der Frames für den Prozess

Was ist Thrashing, wodurch kommt es dazu? Wie erkennt das Betriebssystem Thrashing? Wie kann dieses Problem behoben werden?

Häufige Page Faults erzeugen bei Prozessor viele Ladevorgänge von Sekundärspeicher in den RAM – führt zu drastischem Einbruch der Effektivität

Behebung: mehr Speicher dem Prozess zuordnen (größeres Resident Set (somit suspend und sp versuchen)

Erkennung: mehr Zeit zum Laden der Seiten benötigt als zum Prozess (Resident set zu klein)

Was versteht man unter interner Fragmentierung und externer Fragmentierung? Beschreiben Sie die Begriffe und gehen Sie jeweils auf ein Beispiel an.

Interne Fragmentierung: Verschwendung von Speicher innerhalb der Partition, z.B. bei fixer Partitionierung wird ein kleiner Speicherbereich oder eine große Partition zugewiesen – dadurch entsteht innerhalb der Partition ungenutzter Speicherplatz

externe Fragmentierung: Zerstückelung des Speicherbereichs außerhalb von Partitionen, z.B. bei dynamischer Vergabe des Speicherplatzes kommt es beim Löschen von neuem Speiche

Author: woe007, 18.03.2024

dazu, dass zwischen den Partitionen kleine Bereiche übrig bleiben, die nicht weiter gefüllt werden können

Beim Paging ist ein optimaler Seitenersatzalgorithmus bekannt. Beschreiben Sie diesen und gehen Sie darauf ein, warum er in der Praxis nicht verwendet wird. (3)
OPT Algorithmus:

- erzeugt minimale Anzahl Page Faults
- ersetzt die Seite, die am weitesten in der Zukunft erst wieder verwendet werden
- Praxis unmöglich, da normalerweise nicht im Vorhinein bekannt ist, welche Pages wann verwendet werden
- Sinn nur in der Bewertung anderer Strategien.

Foliensatz 7 – I/O

Was versteht man unter Buffering? Welche Vorteile bietet es, wo liegen seine Grenzen und worauf hat man bei der Verwendung von Puffern bei Betriebssystemimplementierung zu achten?

- Zwischenspeicher bei I/O Transfer
- Vorteile: Zusammenfassung von I/O Operationen (statt 1000 I/O Operationen, nur 1 gesamte), Entkoppelung von Prozess und BS (Swapping), Maskieren von Geschwindigkeitsunterschieden bei Lastspitzen
- Grenzen: Speicherkapazität, Maskieren von Geschwindigkeitsunterschieden bei durchgehend hoher Last nicht möglich, Puffermanagement

Eine Festplatte hat 5000 Zylinder, die von 0 bis 4999 numeriert sind. Es wird gerade auf Zylinder 1995 zugegriffen, der vorangegangene Zugriff erfolgte auf Zylinder 1841. Requests auf folgende Zylinder stehen zur Behandlung an (in FIFO Reihenfolge angegeben): 967, 1470, 917, 1841, 967, 1518, 1554, 1620, 1773. Über wie viele Zylinder muss der Lese/Schreibkopf von der aktuellen Position insgesamt bewegen, um die Requests nacheinander abzuarbeiten: (a) FCFS, (b) Elevator Algorithm (SCAN) und (c) C-SCAN? (6)

a) 103+(1470-967)+(1470-917)+(1841-917)+(1841-967)+(1518-967)+(1554-1518)+(1620-1554)+(1620-1773) = 6660

b) (1995-1995)+(1995-967)+(1554-967)+(1470-1554)+(1518-1470)+(1620-1518)+(1841-1620)+(1841-1773)+(1773-917) = 3395

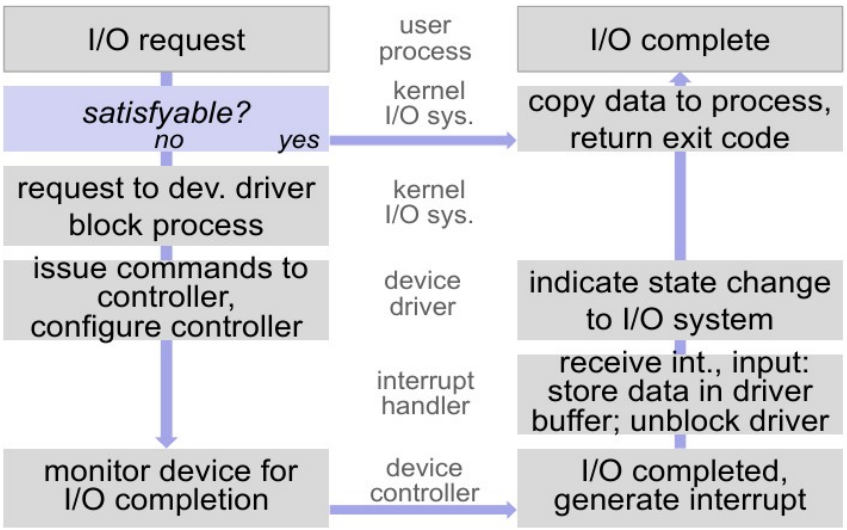
c) 3710

Mit welcher logischen Struktur des I/O Systems versucht man bei der Realisierung von I/O Funktionen sowohl ein einheitliches Programmierinterface, als auch eine möglichst gerätespezifische Ansteuerung zu erreichen?

- Schichtenmodell mit 3 Schichten
  - Logical I/O: logische Bedienung des Geräts, einheitliches Interface), User Input
  - Device I/O: leitet daraus I/O Funktionen ab
  - Scheduling & control: Verwaltung von I/O Operationen, maximierung der Performance

Skizzieren Sie die Folge der Schritte, die bei der Abarbeitung eines Synchronen I/O Requests ablaufen.

Author: woe007, 18.03.2024



Was versteht man unter Blocking bzw. Non-Blocking I/O? Beschreiben Sie die beiden Arten, I/O-Operationen durchzuführen.

- **Blocking I/O:** Prozess wird vom Zustand Running sofort in die Blocked Queue gestellt und blockiert
- **Non-Blocking I/O:** I/O Operation wird sofort durchgeführt, Return liefert sofort Feedback der Operation, kein Blockieren

Was versteht man unter Synchron bzw. Asynchron I/O? Beschreiben Sie die beiden Arten, I/O-Operationen durchzuführen.

- **Synchron:** Bei Synchron I/O wird der auszuführende Prozess, blockiert, bis die I/O Operation wirklich durchgeführt wurde (Prozess wartet, bis Ausgabe am Bildschirm)
- **Asynchron:** Hier kann der Prozess die Daten bei einer I/O Operation auf den Buffer weitergeben und parallel zur stattfindenden I/O Operation weiterarbeiten (obwohl noch nicht am Bildschirm, arbeitet Prozess weiter)

Wie berechnet sich die mittlere Zugriffszeit beim Lesen von Daten von einer mechanischen Festplatte? Geben Sie die charakteristischen Zeitparameter einer Festplatte an. Durch welche Strategien kann das Betriebssystem dazu beitragen, die mittlere Zugriffszeit auf die Festplatte zu reduzieren?

- $T_a = T_s + T_{rd} + T_{Tf}$
- T_s … Seek Time benötigte Zeit, um Disk Arm zur richtigen Spur zu bringen (Mittelwert)
- T_{rd} … Rotational Delay Zeitverzögerung, bis Anfang des gesuchten Sektors gefunden $T_{rd} = 1/2r$
- T_{tf} … Transfer Time benötigte Zeit, um Daten zu übertragen $T_{tf} = b/r*N$ (b Anzahl der übertragenden Bytes, N Anzahl Bytes pro Spur, r Umdrehungsgeschwindigkeit)
- T_a … Average Access Time (benötigte Zeit für Datenzugriff, im Mittelwert)

Strategien:

- **Disk Scheduling** (LIFO, FIFO, Priority, Shortest Service Time First, Elevator Algorithm, C-SCAN, FSCAN)
- **Disk Caching:** Teile der Disk sind im Hauptspeicher zwischengespeichert, das Lokalitätsprinzip mittels Cache (Kombination aus LRU und LFU, 3 Sections)

Beschreiben Sie das Ziel von Disk Scheduling. Nennen Sie „drei gute“ Disk-Scheduling

Author: woe007, 18.03.2024

Algorithmen und beschreiben Sie diese kurz.
Disk Scheduling soll dabei helfen, die Anfragen auf Disk so abzuarbeiten, dass die Seek Time möglichst kurz wird.
Intelligente AG:

- Elevator Algorithm
- C-Scan
- FSCAN

Foliensatz 8 – File Management

Nennen Sie Möglichkeiten, wie die Belegung von Dateien auf Festplatte repräsentiert werden kann.

- unstructured sequence of bytes
- pile: Records variabler Länge werden in Reihenfolge des Ankommens gespeichert
- sequential file: Records mit fixem Format, ein Key-Feld bestimmt die Position innerhalb der Datei
- indexed-sequential file: Index für direkten Zugriff
- direct (hashed) file: Hash-Funktion über Key-Feld, keine sequentielle Reihenfolge der Dateien

Erklären Sie die Begriffe absoluter Pfadname und relativer Pfadname und geben Sie jeweils ein Beispiel an. (4)
absolut: identifiziert Datei durch Beschreibung des Pfades vom Root weg: Unix /usr/hans/mailbox/file.txt
relativ: identifiziert Datei vom CD (Current Directory) aus: /hans/mailbox/file.txt (aus CD Sicht, hier /usr)

Wie ist ein i-node aufgebaut? Welche Information enthält er? (4)

- Jedes File besitzt einen i-node = eindeutige i-Node/File.
- Speicher Flags zu Bestimmen von Permissions, Zähler wieviele Einträge im System auf den i-node verweisen, owner/group ID, Größe des Files, Speicheradresse, letzter Zugriff, letzte Änderung
- Aufbau: Attribute (File) und Referenz auf die Datenblöcke der Datei

Was versteht man unter einer File Allocation Table? Wie ist diese organisiert? (2)

- speichert die Aufteilung der Files innerhalb der Disk (eine Strategie der Block-Allokierung)
- (Filename, Anfangspunkt, Länge) wird gespeichert
- wird bei Indexed Allocation genutzt (Pointer auf Datei in Tabelle)
- Vorteil: sowohl direkt- sequentieller Zugriffe
- Nachteil: großer Platzbedarf für FAT

Bei der Realisierung von Dateisystemen gibt es verschiedene Möglichkeiten, um die zu einer Datei gehörenden Datenblöcke zuzuordnen bzw. auffindbar zu machen (Block-Allokierung). Nennen Sie vier verschiedene Strategien zur Block-Allokierung von Dateien und beschreiben Sie diese mit ihren Vor- und Nachteilen.

- Contiguous Allocation: eine Datei = Menge aneinander gereihter Blöcke
  - Vorteil: gute Performance beim Lesen
  - Nachteil: Problem bei Vergrößerung einer Datei
- Chained Allocation: Belegung einzelner Blöcke, die über Zeiger verkettet werden (wird in Block gespeichert)
  - Vorteil: keine externe Fragmentierung, leicht erweiterbar

Author: woe007, 18.03.2024

    - Nachteil: kein Lokalitätsprinzip, langsamer Zugriff
- Indexed Allocation wie Chained Allocation, jedoch werden die Pointer in einer Tabelle (FAT), und nicht in den Blöcken.
    - Vorteil: sowohl direkter als auch sequentieller Zugriff gut unterstützt
    - Nachteil: großer Platzbedarf im RAM
- I-Nodes: Datenstruktur für das ganze File, speichert sowohl Attribute als auch Pointer auf andere Blöcke des Files
    - Vorteil: I-Node wird nur gebraucht, solange benutzt
    - Nachteil: begrenzte Anzahl der Blockreferenzen pro i-node, Verkettung und daher Verwaltung doppelter/dreifacher indirekter Blöcke

Beschreiben Sie das typische Layout einer Disk bzw. eines Filesystems. Welche Rolle spielen die einzelnen Teile beim Hochfahren des Betriebssystems? (5)

- Disk ist in Partitionen unterteilt, mit unabhängigen Filesystems
- Master Boot Record (Sektor 0 der Disk) enthält Boot Code und Partition Table)
- System startet BIOS, exekutiert MBR, aktive Partition wird lokalisiert und der erste Boot Block wird ausgeführt (Laden des OS der aktiven Partition)

Welches Dateiformat einer regulären Datei hat für ein OS besondere Bedeutung? Warum? Und wie wird es erkannt?

Binary Files erlauben beliebige Bitmuster und ausführbare Dateien (.exe)

Werden erkannt mit einer bestimmten, Magic Number im Header, dass das bin ein (ist exe ist. (ist ein spezielles Bitmuster in den ersten Bytes des Headers).

Foliensatz 99 – Security

Die Implementierung einer Zugriffsmatrix kann in der Form von Access Control Lists oder Capability Lists erfolgen. Erklären Sie diese Begriffe. (4) (4)

- Access Control List: Zugriffsrechte sind bei Objekten gespeichert (Spaltenzerlegung), es wird nach Benutzergruppen differenziert, werden im Kernel Space gehalten
- Capability List: pro Prozess gibt es eine Liste mit Objekten und den dazugehörigen Rechten, ebenfalls im Kernel Space, Tickets regeln Zugriff (User muss Ticket für einen Zugriffsevent auf Objekt besitzen)

Was beschreibt das Modell von Bell und LaPadula? Geben Sie die Modelleigenschaften an. (4)

Welche Regeln für den Informationsfluss hierarchischer Security Classifications für Subjects und Objects (top secret, secret, public, sec ret) können die Subjects auf den Objekten Operationen ausführen, read-only, read-write, append, execute

gefordert – Security Axiome:

- simple security property: Read (S) ⊆ SC(S) ⊇ SC(O) (no read up)
- property append: SC(S) ⊆ SC(O) (no write down)
- property read & write: SC(S) = SC(O)

Nennen Sie Design Prinzipien für die Konstruktion von sicheren Systemen. Geben Sie für jede Regel ein Beispiel an. (4) (4)

Author: woe007, 18.03.2024

- **Open Design:** keine Sicherheit durch besondere, schwer zu durchschauende Systeme → Sicherheitssysteme müssen verständlich bleiben
- **Default:** Einstellung keine Berechtigung → Bsp: User soll von Haus aus keine Admin Rechte haben
- **Least Privilege:** so wenig Rechte wie möglich, nur die nötigsten
- **Economy of Mechanism:** Fehlervermeidung durch Einfachheit der Sicherheitsmechanismen, niedrige Ebene implementieren
- **Acceptability:** System nicht so umständlich, dass Nutzer es nicht nutzen möchten
- **Überprüfung der gegenwärtigen Berechtigungen:** nicht einfach annehmen, dass anfängliche Berechtigungen immer konstant sind
- **Complete Mediation:** Kontrolle aller Zugriffe auf Ressourcen, auch in Ausnahmesituationen

Nennen Sie die drei Kategorien von Security Threats und beschreiben Sie diese. Geben Sie für jede Kategorie an, welches grundlegende Security Ziel dadurch bedroht wird. (4)
Es gibt drei grundsätzliche Kategorien von Security Threats (Abhören/Monitoring des Wissen des Betroffenen) und Active Threats (System/Daten werden manipuliert)

- **Denial of Service (Interruption):** vorübergehende oder permanente Unterbrechung eines Services (durch Zerstörung, Überlast) → bedroht AVAILABILITY
- **Exposure:** unautorisierter Lesezugriff → CONFIDENTIALITY
- **Modification:** Veränderung der Datenintegrität (Man in the Middle), Daten werden verändert → INTEGRITY

Beschreiben Sie das Prinzip einer Sicherheitslücke durch Stack Buffer Overflow. Wodurch kann man sich bei der Implementierung eines Betriebssystems vor einem solchen Angriff schützen? (4)
Funktionsweise: Wenn eine Funktion aufgerufen wird, schreibt der Angreifer über Puffer, und wird somit die Return-Adresse überschrieben, die sagt, was als nächstes (nach Funktionsaufruf) gemacht werden soll. Nun ist das Ziel, den Stack mit schädlichem Code zu beschreiben, und die Return-Adresse so zu überschreiben, sodass stattdessen der eigentliche Funktionalität dann der schädliche Code (Remote Code Execution) durchgeführt wird.

Schützen: Immer kontrollieren bei Funktionen wie gets() oder strcpy() ob der Platz für User-Input vorhanden ist (Länge des Inputs!)

In welcher Art von Systemen ist der Einsatz von kryptographischen Verfahren zur Sicherung der Vertraulichkeit und Integrität von Daten notwendig? Was sind kryptographische Verfahren in diesen Systemen sicher? (3)
In offenen Systemen notwendig (wie Internet), basiert auf dem Besitz von geheimen Schlüsseln Sicherstellen durch Verschlüsseln der Nachricht (Confidentiality) und Signieren der Nachricht (Integrity/Authenticity)