Author: woec007, 18.03.2024

# Fragensammlung aller bisherigen schriftlichen Prüfungen – VU Betriebssysteme (Puschner) [alle VOWI verfügbaren, bis 2023 – 2010]

## Foliensatz 1 – Betriebssysteme, eine Einführung

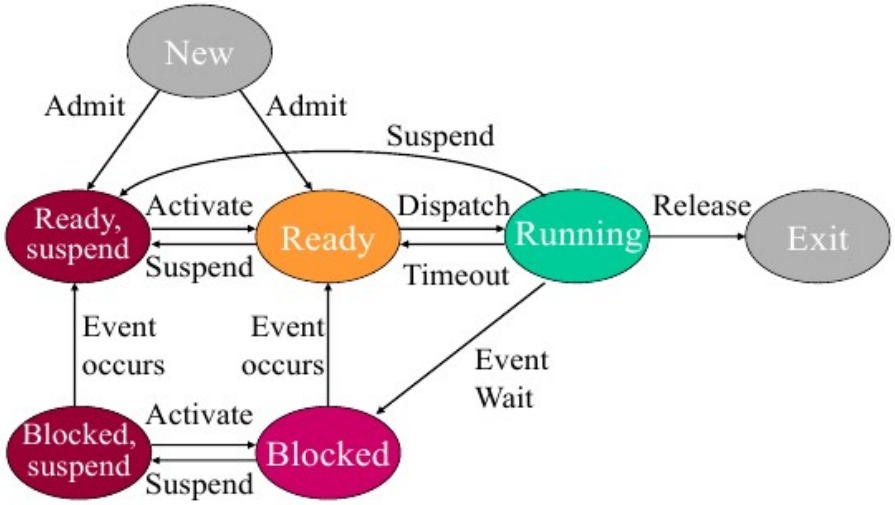
Nennen Sie zwei Abstraktionen, die ein Betriebssystem dem Benutzer bietet. Welche Aufgaben führt das Betriebssystem durch, um diese Abstraktionen zu realisieren?

- „ungestörte" Programmarbeitung → Prozessmanagement
- „unendlich" großer Speicher → Betriebssystem führt Speicherverwaltung durch
- „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 X (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 in Blocked or Ready-Zustand)
- Dispatch: Abrufen des Prozesses vom Ready-Zustand in den Running-Zustand (CPU)

Author: woec007, 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 geblockedum 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 von 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 in dazugehörigen gleichartigen Zuständen 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 Registerinhalte, Kontoll- und Statusregister, und Stackpointer
- Process Control Information: Zustand des Prozesses, speichert Scheduling und Prioriätatinformationen, Querverweise 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 eines jeden Mechanismus kurz.
- Process Switching: Wechsel von Prozessen gemäß Scheduling Strategy
- I/O und Hardware Access: Management der zugehörigen 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 Process wiedergeladen werden sollte: alter Prozess und Prozessorzustand wird wiederhergestellt

Author: woec007, 18.03.2024

Erklären Sie die Begriffe Process Switch und Mode Switch, sowie die Beziehung, in der diese beiden Konzepte stehen. Zählen Sie weiters die drei Klassen von Ereignissen auf, die einen Mode Switch nach sich ziehen.
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)

Mode Switch (werden eben durch Sys Call, Trap oder Interrupt erzeugt) ist das Wechseln vom User Mode in einen Privileged Mode. Im User Mode sind keine Process Switches möglich, insofern bedingen Mode Switches (nicht unbedingt, aber sind nötig) Process Switches → Execution Mode Wechsel (Execution Modes existieren zum Schutz der Datenstrukturen des Betriebssystems)

Worin liegt der grundsätzliche Unterschied zwischen Prozessen und Threads? Welcher Vorteil ergibt sich aus der Einführung von Threads für den Benutzer und worauf muss der Benutzer achten?
Prozesse kümmern sich sowohl um die Ressourcenverwaltung als auch um das Dispatching (kurzfristiges Scheduling) – mit der Einführung von Threads entkoppelt man diese beiden Aufgaben, Threads kümmern sich nur um das Dispatching.

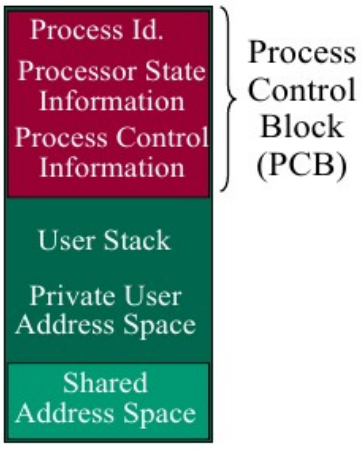
- Prozesse: virtueller Adressraum mit Process Image, Speicherschutz, I/O, Files
- Thread: der zeitiger Ausführungszustand, Stack, Kontext, thread lokale Variablen, besitzt Zugriff auf Prozessspeicher und Ressourcen

Vorteile:

- Thread-Erzeugung/Terminierung benötigt weniger Zeit,
- Umschaltung zwischen Threads geht schneller als Process switch
- Kommunikation ohne Kernel
- Benutzer muss beachten: Synchonisation zwischen Threads bei ULT seine Aufgabe!

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

- Beschreibt den derzeitigen Status/Aufbau des Prozesses, wird von BS benötigt zur Speicherverwaltung in RAM
- Process Tables in BS zeigen auf Process Images
- im Virtual Memory
- Besteht aus:
  - Process Control Block (Process Identification, Processor state information, Process control information)
  - User Programm: enthält alle unveränderlichen 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 zentralen Services muss ein Microkernel zur Verfügung stellen? Welche Vor und Nachteile ergeben sich bei der Verwendung eines Microkernels

- Nur die nötigsten Basisservices befindet sich im Kernel, nicht zentale Services des Betriebssystems als Serverprozesse → auf Prozessebene.

Author: woec007, 18.03.2024

- Vorteil: ist flexibel, einheitlich und Portabel
- Nachteil: Micro ist ungleich Micro (nicht klar definiert, unterschiedliche Anzahl an Services)
- Zentalen Services: Process Switching, Memory Management, Interrupts, Hardware Access und Nachrichtenaustausch bzw. Kontolle zwischen Prozessen und Kernel

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

- Multithreading bezeichnet man, sobald jeder Prozess n>1 Threads pro Prozess besitzt. Dies bedeutet, dass jeder Prozess in mehrere Threads (Aufgabe: Dispatching) unterteilt werden kann, während der Prozess selbst nur noch das Ressourcenmanagement/Verwaltung betreibt.
- Problem: Zugriff auf gemeinsam genutzte Resource – wer darf wann R/W Access besitzen?
- Thread-Control-Block je Thread, besitzt Kontolle über Thread

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

- System Call: explizter Aufruf 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 Ressourcen verwaltung dem Prozess überlassen, 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 Synchonisation 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 deren Unterschiede.

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

Author: woec007, 18.03.2024

    - wird Prozess vom Kernel geblockt, blockieren alle Threads. Ruft ein ULT einen blockierenden System Call (I/O) auf, so werden alle Threads des Prozesses gestoppt
    - Synchronisation ist in User Mode
    - kein Ausführen auf mehreren Kernen gleichzeitig

Wie unterscheidet sich das Blockierverhalten 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 anweisung

Beschreibe die drei 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 Betriebssystem in User-Prozessen:
    - BS ist Sammlung von Routinen, die in User Prozessen ausgeführt werden können
    - Verlassen des Prozesmodes 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-Sevices sind eigenständige Prozesse

Welche Kernelarchitekturen gibt es?

- Monolithic OS: nur bei kleinen Betriebssystemen, Menge an Prozeduren die sich gegenseitig callen (veraltet)
- Layered OS: geschichtete Systeme auf hierarchische Art, nur benachbarte Schichten können miteinander agieren, Aufbauen
- Modular OS: verschiedene Ansätze realisieren die wichtigsten Funktionalitäten vom Betriebssystem, Hardware Abstraction Layer in unterster 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 den Mid-term Scheduler oder den Short-Term Scheduler übergeben wird
- bestimmt Parallelitätsgrad

Mid Term Scheduling:

- Scheduling für das Ein- und Auslagern in Sekundärspeicher (Swapping)

Short Term Scheduling:

- bestimmt welcher Prozess aus Ready Queue als nächstes in die CPU geladen wird
- = Dispatcher

Author: woec007, 18.03.2024

- wird bei Interrupt/OS Signal Calls / Signalen 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 Serves:

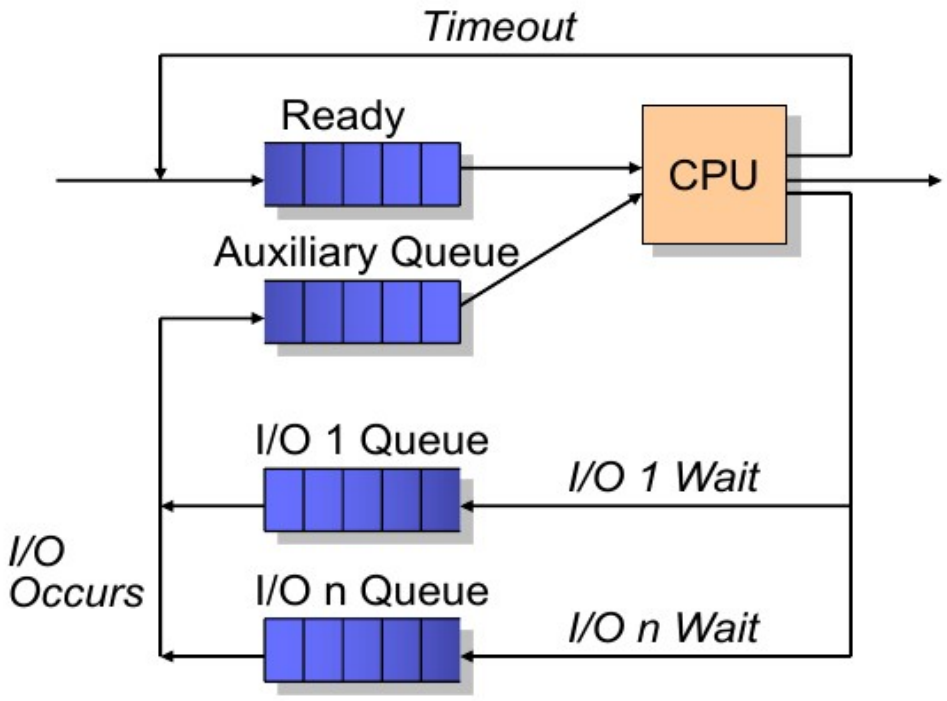
- Selection Function: wer zuerst kommt, ist zuerst dran
- Decision Mode: Non-Preemptive
- begünstigt lange und CPU-intensive Prozesse, während I/O intensive Prozesse immer wieder ans Ende der Ready Queue geschoben werden
- ermöglicht Starvation und Endlosschleifen

Round Robin:

- Selection Function: wie bei First Come First Serve; ältester Prozess in Ready-Queue wird ausgewählt
- Decision Mode: Preemptive (unterbrechend)
- jeder Running prozess bekommt den gleichen (kurzen) Zeitslot, sobald Zeit abgelaufen → Prozesswitch (zyklische Weitergabe von Zeitslots)

Virtual Round Robin:

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




Shortest Process Next (SPN):

- Selection Function: wählt Prozess mit geringster CPU Dauer aus

Author: woec007, 18.03.2024

- DecisionMode: Non-Preemptive
- kurze Prozesse werden begünstigt
- bessere ResponseTimes als FCFS
- Probleme: Starvation möglich, Berechnung der erwarteten Programmdauer schwierig, non-preemptive ermöglicht Blockieren der CPU

ShortestRemainingTime (SRT):
- SelectionFunction: wie bei ShortestProcessNext, wählt Prozess mit geringster CPU Dauer aus
- DecisionMode: Preemptive (unterbrechend)
- fairer in Bezug auf kürzere Prozesse als ShortestProcessNext (SPN)
- Probleme: Starvation möglich, Berechnung der erwarteten Programmdauer schwierig

HighestResponseRatioNext:
- SelectionFunction: RR = (w+s)/s [w... gesamte bisher gewartete Dauer, s ... geschätzte ServiceTime), Priorität höher, je größer RR
- DecisionMode: Non-Preemptive
- fairer in Bezug auf kürzere Prozesse
- keine Starvation jedoch Berechnung der erwarteten Servicetime nötig sten

Feedback-Scheduling:
- SelectionFunction: basiert auf bisheriger Ausführungszeit je mehr CPU Time ein Prozess konsumiert hat, desto niedriger wird Priorität – eigene Queue für jede Priority
- DecisionMode: preemptive
- Problem: Starvation möglich, Abhilfe mittels anhebender Priority

Erklären Sie die Begriffe Deadlock, Lifelock und Starvation. (4)
- Deadlock: durch zyklische Abhängigkeit beim Zugriff auf Resourcen, kann keiner der beiden Prozesse die notwendigen Ressourcen anfordern, aber gibt sie auch nicht mehr ab (no preemption) → Programm kann nicht weiter laufen
- Lifelock: Prozess wird der Eintritt in den kritischen Abschnitt verwehrt, kein Fortschritt möglich
- Starvation: ein Prozess kann nie auf bestimmte Ressourcen zugreifen, weil immer andere vorher abgearbeitet werden

Nennen Sie die Arten von Optimierungszielen, die ein Scheduler beim Prozess-Scheduling verfolgen kann und geben Sie jeweils Beispiel an.
- Durchsatz: desto besseres Scheduling, desto mehr Prozesse können nacheinander beendet werden – SRT
- Fairness: alle Programme müssen irgendwann beendet werden, kein Lifelock – Round Robin
- ResponseTime: möglichst schnelle Antwort – z.B. ShortestProcessNext (SPN)
- Einhalten von Deadlines: Alle (periodischen) Prozesse müssen spätestens dann beenden, wenn gefragt (EDF)
- Prozessorauslastung möglichst wenig Leerlauf, immer am Arbeiten – Virtual Round Robin

Author: woec007, 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 möglichs
b) Ja, da eventuell (falls immer wieder neue Prozesse kommen), immer kürzere vorhanden sein könnten
c) Nein, da jeder Prozess eine zyklische Zeitscheibe erhält → I/O lastige Prozese sind jedoch sehr benachteiligt daher VRR
d) Ja

Was versteht man unter Real-Time Scheduling?
- Echtzeit-Scheduling bezieht sich auf die WECT, also auf die Deadlines:
  - Soft Deadline: Verpassen ist nicht kritisch/katastrophal (Temperatursensor)
  - Hard Deadline: Deadline muss unbedingt eingehalten werden, sonst Katastrophe (FCS, Airbag)
- üblicherweise Preemptive
- oft bei periodischen Prozessen
- Schedulability Test: überprüfen, ob Task Set scheduable ist – also ob alle Prozesse rechtzeitig beendet werden können

Erklären Sie Earliest Deadline First Scheduling!
- Ist ein Real-Time Scheduling Verfahren
- Selection Function: Task mit frühester Deadline (Deadline absolut angegeben)
- Decision mode: preemptive
- Optimierungsziel: Minimierung der verpassten Deadline
- Schedulability Test: Summe ($C_i/T_i$) <= 1 ($C_i$ Dauer, $T_i$ Periode)
- Wird immer geschaut, welcher 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: woec007, 18.03.2024

- Prozedurenregeln Zugriff durch Warten auf bestimmte Bedingungen (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 mittel Monitorprozeduren
  - max. 1 Prozess gleichzeitig im Monitorprozedur
  - Bedingungssynchronisation über Monitorvariable
  - Bedingungsvariablen lokal nur im Monitor zugreifbar (cwait, csignal, csignal speichert nicht)

Für die Lösung des Problems des geregelten Eintritts in einen kritischen Abschnitt werden drei Eigenschaften gefordert. (a) Nennen Sie diese drei 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/Wartenden vor einem
b) Semaphore bestehen aus einem Value 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 einen konsistenten Datenaustausch zwischen parallelen Prozessen realisieren können

- blockierendes und nichtblockierendes Message-Passing
- Blockierend: Beim Empfangen/Senden einer Message wird solange gewartet, bis sichergestellt werden kann dass der Messagpartner alle empfangen hat
- Nicht-Bockierend: 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 einer der 4 Deadlockbedingungen
  - No Mutex (indirekt): kann nicht verhindert werden, ist nötig aufgrund unserer Aufgabenstellung
  - Hold & Wait (indirekt): Prozesse fordern alle Ressourcen auf einmal an, Blockieren bis alle da sind → lange Verzögerung, Prozess braucht Wissen über Resourcen, die er Verwenden wird
  -

Author: woec007, 18.03.2024

    - No Preemption (indirekt): zugewiesene Ressourcen werden nicht weggenommen, Prozess gibt Ressourcen frei, wenn er eine nicht bekommt – anwendbar bei leicht speicherbaren Ressourcen (Process Swt ch)
    - Circular Wating verhindern (direkt): Protokoll, mit strikter linearer Ordnung bzgl Ressourcenart, in der Folge werden nur mehr Anforderungen zugelassen die unter der Grenze liegen
- Deadlock Avoidance erlaubt Bedingungen 1 bis 3, selektives Vergeben von Ressourcen
    - Process Initiation Denial: Prozess wird nicht gestartet, wenn seine Anforderungen zu einem Deadlock führen könnten → Ri <= C(n+1)i + Summe(Cki) – 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 zu Bankers) + Strategie zur Deadlockbehebung (Recovery), sehr CPU intensiv

Bei der Deadlock-Vermeidung spricht man von einem Safe State bzw. einem Unsafe State. Erklären Sie die Bedeutung dieser Begriffe.
Bei Deadlock Avoidance, im Resource Allocation Denial führt man den sog. Bankers Algorithmus, welcher als Ergebnis einen „Safe state" oder einen „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 Eintreten eines Deadlocks und erklären Sie diese.
    - No Mutex (indirekt Deadlock prevention) kann nicht verhindert werden, ist nötig aufgrund unserer Aufgabenstellung
    - Hdd & Wait (indirekt Deadlock prevention): Prozess kann Ressourcen halten, während er auf andere wartet
    - No Preemption (indirekt Deadlock prevention): zugewiesene Ressourcen werden nicht weggenommen bei leicht speicherbaren Ressourcen (Process Swt ch)
    - Circular Wating verhindern (direkt Deadlock prevention): geschlossene Kette von Prozessen, von denen jeder min. 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.
Ressourcenvergaben, die zu Deadlocks führen könnten, werden nicht gewährt (Bedingungen 1 bis 3 sind erlaubt, Circular Wating nicht). Höhere Parallelität als Deadlock Prevention, man muss jedoch Wissen bezüglich des Ressourcen Bedarfs haben
    - Process Initiation Denial: Prozess wird nicht gestartet, wenn seine Anforderungen zu einem Deadlock führen könnten → Ri <= C(n+1)i + Summe(Cki) – sehr defensiv (Claimmatrix 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: woec007, 18.03.2024

- erlaubtes, die nur für den Prozess derzeit Relevanten Pages in den RAM zu laden
- Segmenttabelle für Überprüfung
- Falls angeforderte 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 ja nicht alles gleichzeitig geladen werden sein muss., einfache Adressübersetzung

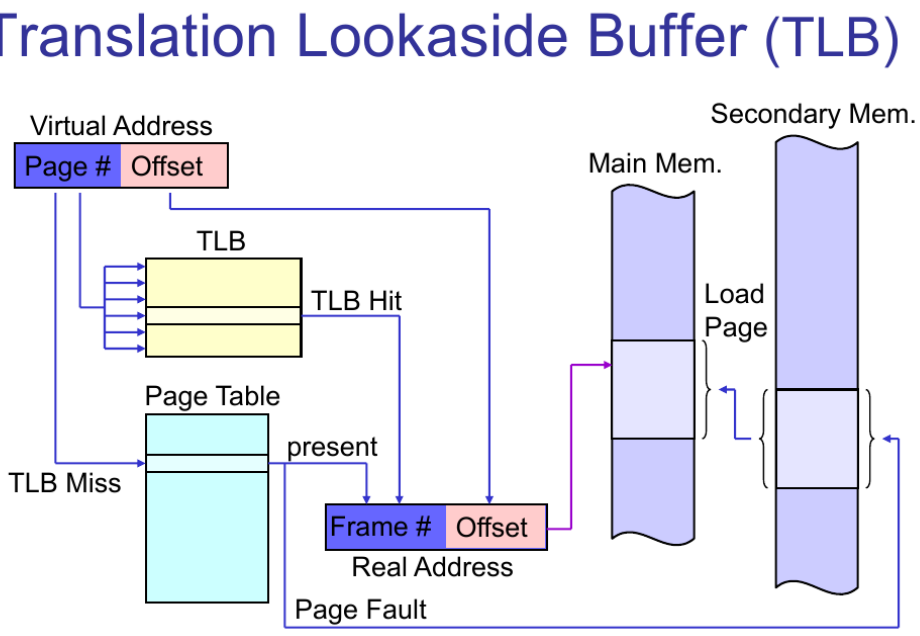
Nennen Sie Möglichkeiten, um in einem Paging-System Speicherschutz zu realisieren? (3)

- Bound Check: Logischer Adressbereich ist kontinuierlich zusammenhängend, man überprüft ob Adresse innerhalb der (maximalen) Länge des Adressbereichs liegt.
- Bei Speicherbereichs-Sharing: Protection Keys
- Variante 1: Jeder (physische) 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 einer der Prozesskeys mit TLB Key übereinstimmt → Zutritt
- sonstimmer: Speicherbereichsverletzung

Beschreiben Sie Aufgabe und Funktion eines Translation Lookaside Buffers? Worauf hat man bei der Betriebssystemimplementierung bei einem Process Switch 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 zur physischen Adresse, schaut man zuerst mit dem ersten Teil der virtuellen Adresse (#Page) in der TLB nach, und sucht nach der passenden Page number. Findet man eine (TLB Hit), so hat man das assoziative #Frame (Frame number), und muss nicht mehr in die Page Table schauen. Offset bleibt gleich.



Author: woec007, 18.03.2024

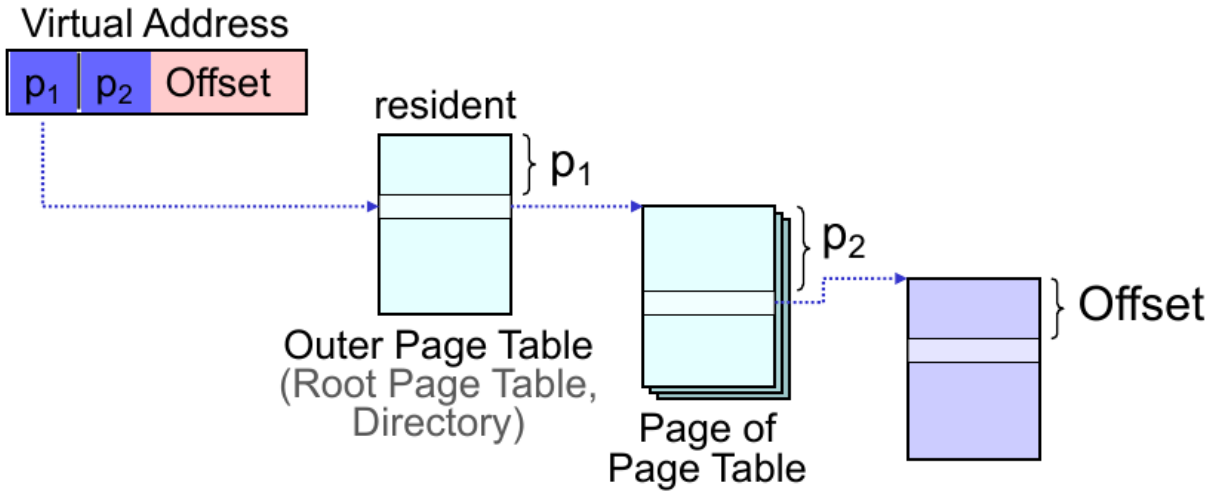
Bei einem Process Switch muss die TLB gelöscht werden, d.h. am Anfang des neuen Prozesses sind nicht viele Einträge vorhanden (unwahrscheinlich, einen TLB Hit zu treffen)

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

- Keine statische Fixierung des Speicherbereichs vom Prozess → Instruktionen müssen an verschiedener 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: Adresse von bekanntem Punkt aus]

Wir betrachten ein System mit Virtual Memory Management. Diskutieren Sie, welche der folgenden Situationen beim Referenzieren einer virtuellen Adresse auftreten bzw. nicht auftreten kann: (a) TLB Miss ohne Page Fault, (b) TLB Miss mit Page Fault, © 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 wurde (und somit muss sie derzeit im Resident Set sein)

Beschreiben Sie den Aufbau und die Verwendung einer Multilevel Page Table (ev. Mit Skizze). Warum werden Multilevel Page Tables verwendet?




Anwendung große Prozesse, mit großer Page Table, falls Page Table nicht mehr in RAM passt, wird unterteilt in zwei Page tables (min 3. Zugriffe, dauert länger)

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

Author: woec007, 18.03.2024

wird gemacht, da ganzer Prozess oft nicht in Hauptspeicher passt. Page Table ist zur Übersetzung zwischen Prozess (Pages) und RAM (Frames) da.

Page Table wird zur Adressübersetzung einer virtuellen Adresse in eine physische Adresse benutzt. Dabei baut sich eine Virtuelle Adresse (Page# | Offset) auf. Der Offset bleibt bei der physikalischen Adresse gleich, die Page# wird in der Page Table als Index benutzt. In diesem Index findet man daraufhin die physische Frame number 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 deren Funktionsweise.

- Ist eine Page Replacement Strategie
- wird also angewandt, um mit möglichst wenig Page Faults immer die 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 eine Flag auf 1 gesetzt. Es wird immer das ersetzt, auf das der Pointer zeigt. Falls Pointer auf eines 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 deren 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 ist W(D,t): blickt zu einem Zeitpunkt t D Zeiteinheiten zurück, und speichert (variabel in der Anzahl) die in dieser Zeitheit benutzten Pages ab
- basiert auf dem Lokalitätsprinzip
- wächst schnell an, stabilisiert sich & wächst wieder schnell bei einem Process Switches
- Resident Set soll dann immer zu bestmmten Zeitpunkten sich an Working Set orientieren
- Problem: optimales D unbekannt, variert – mit bggen sehr schwierig, daher gängige Praxis:
  - Beobachtung der PF/Zeitintervall 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 beseitigt werden?
Häufige Page faults erzeugen beim Prozessor viele Ladevorgänge vom Sekundärspeicher in den RAM – führt zu drastischem Einbruch der Effektivität
Behebung mehr Speicher dem Prozess zuordnen – größeres Resident Set (sonst suspend und später versuchen)
Erkennung mehr Zeit zum Laden der Seiten benötigt als zum Prozess ausführen (Resident set zu klein)

Was versteht man unter interner Fragmentierung und externer Fragmentierung? Beschreiben Sie die Begriffe und geben Sie je ein Beispiel an.
Interne Fragmetierung: Verschwendung von Speicher innerhalb der Partition, z.B. bei fixer Partitionierung wird einem kleinen Speicherbereich oft eine große Partition zugewiesen – dadurch entsteht innerhalb der Partition viel ungenutzter Speicherplatzes

externe Fragmetierung: Zerstückelung des Speicherbereichs ausserhalb von Partitionen, z.B. bei dynamischer Vergabe des Speicherplatzes kommt es beim Löschen/Vergeben von neuem Speicher

Author: woec007, 18.03.2024

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

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

- erzeugt minimale Anzahl an Pagefaults
- ersetzt die Seiten, 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 der Betriebssystemimplementierung zu achten?

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

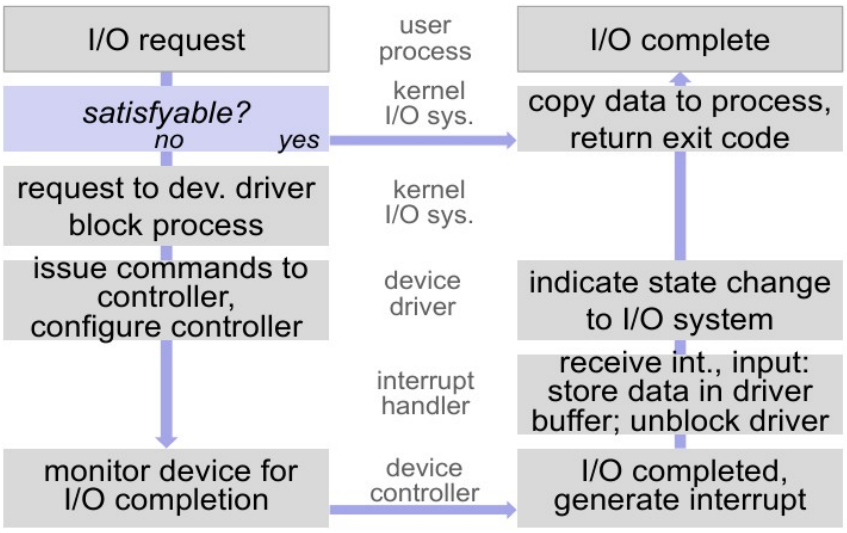
Eine Festplatte hat 5000 Zylinder, die von 0 bis 4999 nummeriert sind. Es wird gerade auf Zylinder 195 zugegriffen, der vorhergehende Zugriff erfolgte auf Zylinder 164. Requests auf folgende Zylinder stehen zur Behandlung an (in FIFO Reihenfolge angegeben): 92, 1470, 917, 1841, 967, 1518, 1154, 1620, 173. Über wieviele Zylinder muss sich der Lese/Schreibkopf von der aktuellen Position in Summe bewegen, um die Requests nach folgenden Strategien abzuarbeiten: (a) FCFS, (b) Elevator Algorithm (SCAN) und (c) C-SCAN? (6)
a) 103 + (1470-92) + (1470-917) + (1841-917) + (1841-967) + (1518-967) + (1518-1154) + (1620-1154) + (1620-173) = 6660
b) (917-195) + (967-917) + (1154-967) + (1470-1154) + (1518-1470) + (1620-1518) + (1841-1620) + (1841-173) + (173-92) = 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?

- Schichtmodell mit 3 Schichten
  - Logical I/O: logische Bedienung des Geräts, einheitlich (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: woec007, 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 Synchronous bzw. Asynchonous I/O? Beschreiben Sie die beiden Arten, I/O-Operationen durchzuführen.

- Synchronous: Bei Synchronos I/O wird der auszuführende Prozess blockiert, bis die I/O Operation wirklich durchgeführt wurde (Prozess wartet, bis Ausgabe am Bildschirm)
- Asynchronos: Hier kann der Prozess die Daten bei einer I/O Operation an den Buffer weitergeben und parallel zur stattfindenen 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 mittleren Zugriffszeiten auf die Festplatte zu reduzieren?

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

Strategien:

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

Beschreiben Sie das Ziel von Disk Scheduling. Nennen Sie drei „inteligente" Disk-Scheduling

Author: woec007, 18.03.2024

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

- Elevator Algorithm
- C-Scan
- FSCAN

## Foliensatz 8 – File Management

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

- unstructured sequence of bytes
- pile: Records variable Länge werden in Reihenfolge des Ankommens gespeichert
- sequential file: lauter Records mit fixem Format, ein Key-Feld bestimmt die Postion innerhalb der Datei format
- indexed sequential file: index für direkten Zugriff
- direct (hashed) file: Hash-Funktion über Keyfeld, 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 Beschreobung des Pfads von 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 Informationen enthält er? (4)

- Jedes File besitzt einen i-node → ein i-node/File.
- Speichert Flags zum Bestmmen von Permissions, Zähler wie viele Einträge im System auf den iNode verweisen, owner/group id, Größe des Files, Speicheradresse, letzter Zugriff, letzte Änderung
- Aufbau: Attribute (File) und Referenzen 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-Alokierung)
- (Filename, Anfangspunkt, Länge) wird gespeichert
- wird bei Indexed Allocation genutzt (Pointer auf Dateien in Tabelle)
- Vorteil: sowohl direkt/sequentielle Zugriffe
- Nachteil: großer Platzbedarf für FAT

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

- Contguous Allocation: eine Datei → Menge an einander grenzende Blöcke
  - Vorteil: gute Performance beim Lesen
  - Nachteil: Probleme beim Vergrößeren einer Datei
- Chained Allocation: Belegung einzelner Blöcke, die über Zeiger verkettet werden (wird in Block gespeichert)
  - Vorteil: keine externe Fragmetierung, leicht erweiterbar

Author: woec007, 18.03.2024

    - Nachteil: kein Lokalitätsprinzip, langsamer Zugriff daher
- 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, sobald benutzt
    - Nachteil: begrenzte Anzahl der Blockreferenzen pro i-node → Verkettung und daher Verwaltung doppelte/dreifache 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 in 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 eine exe ist. (ist ein spezielles Bitmuster in den ersten Bytes des Headers).

## Foliensatz 9 – Security

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

- Access Control Lists: 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 in Kernel Space, Tickets regeln Zugriff (User muss Tickets für ein Zugriffsevent auf Objekt besitzen)

Was beschreibt das Modell von Bell und LaPadula? Geben Sie die vom Modell geforderten Eigenschaften an. (4)
Dieses Modell beschreibt Regeln für den Informationsfluss → Hierarchie von Security Classifications für Subjects und Objects (top secret, public, secret).
Nun können die Subjects auf den Objects Operationen ausführen read-only, read-write, append, execute

gefordert – Security Axiome:

- simple security property: Read: 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)

Author: woec007, 18.03.2024

- Open Design: keine Sicherheit durch besonders schwere Codes → 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 Mechanics: Fehlervermeidung durch Einfachhalten 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 Ressource, 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)
grundsätzliche Unterscheidung in passive Threats (Abhören/Monitoring ohne Wissen des Betroffenen) und in Active Threats (System/Daten werden manipuliert)

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

Beschreiben Sie das Prinzip einer Sicherheitsattacke durch Stack/Buffer Overflow. Wodurch kann man sich bei der Implementierung eines Betriebssystems vor einen solchen Angriff schützen? (4)
Funktionsweise: Wird eine Funktion ausgeführt, so schreibt der Angreifer über Pufferende und will somit die Return Adresse überschreiben, die sagt, was er als nächstes (nach Funktionsaufruf) machen soll. Nun ist das Ziel, den Stack mit schädlichem Code zu beschreiben und die Returnadresse überschreiben, sodass statt der eigentlichen Funktionalität danach der schädliche Code (Remote Code Execution) durchgeführt wird.

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

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