# Kapitel 1 - Überblick

1. Was sind die wesentlichen Aufgaben eines Betriebssystems?

* Schnittstelle zwischen Benutzer, den Anwendungen und der Hardware
* steuert die Ausführung von Programmen

2. Welche Arten von Schnittstellen hat ein Betriebssystem?

* Benutzerschnittstelle
  + GUI
  + Kommandointerpreter mit Skript-Sprache
* Programmierschnittstelle
  + Hardware-unabhängige Prozeduraufrufe

3. Was versteht man unter Uni- bzw. Multiprogramming?

* Uniprogramming
  + Nur ein Prozess gleichzeitig
  + Prozessor wartet auf das Ende jeder (langsamen!) I/O-Operation, bevor das Programm fortfahren kann
* Multiprogramming
  + Mehrere Programme (Jobs) sind gleichzeitig im Hauptspeicher
  + Wenn ein Programm (Job) die CPU „freiwillig“ abgibt und auf I/O wartet, kann der Monitor die CPU auf ein anderes Programm umschalten

4. Was ist der Unterschied zwischen Multiprogramming und Timesharing?

|  |  |  |
| --- | --- | --- |
|  | Batch Multiprogramming | Time Sharing |
| Prinzipielles Ziel | Maximierung der CPU-Auslastung | Minimierung der  Antwortzeit |
| Anweisungen kommen von: | JCL – Elementen, die jedem Programm beigefügt sind | Kommandos, die am Bildschirm eingegeben werden |
| Unterbrechung wegen: | I/O-Anforderung | Ablauf der Zeitscheibe |

5. Wie funktioniert das Grundprinzip eines Universalrechners?

* ???

6. Nennen Sie für Betriebssysteme wichtige CPU-Register!

* Befehlszähler (PC – Program Counter)
  + enthält die Hauptspeicher-Adresse des Befehls, der ausgeführt werden soll
* Speicheradressregister (MAR – Memory Address Register)
  + Hauptspeicheradresse des nächsten zu holenden Datenworts (für Adressbus)
* Speicherpufferregister (MBR – Memory Buffer Register)
  + Nimmt das nächste zu holende Datenwort vom Datenbus
* Befehlsregister (IR – Instruction Register)
  + enthält den Befehl, der ausgeführt werden soll
* Ein oder mehrere Statusregister (PSW – Program Status Word) mit einzelnen Bits für:
  + Conditions Code (Flags – Ergrebnisanzeigen)
  + Kernel/User – Modusbit
    - Usermodus = eingeschränkter Zugriff
    - nicht alle Instruktionen stehen zur Verfügung!
    - Je nach CPU: Mehrere Ebenen (Ringe)
  + Interrupt-Bit (Interrupts freigegeben /gesperrt?)
* User-Register
  + sind Arbeitsregister für Benutzerprogramme (wie temporäre Variablen)
  + reduzieren Zugriffe auf Hauptspeicher

7. Was ist Interrupt?

* Eine Unterbrechung des Programmablaufs

7a. Welche Interrupt-Arten unterscheidet man?

* Asynchroner Interrupt
  + Hardware Interrupt
* Synchroner Interrupt („Trap“)
  + Exception
  + Software Interrupt

7b. Wie werden Interrupts durch die Hardware realisiert?

* Ein Hardwaregerät löst einen Interrupt aus. z. B. Timer, Reset, Serielle Schnittstelle, Festplatte

7c. Was geschieht, falls während des Behandlung eines Interrupts ein neuer Interrupt auftritt?

* Wenn neuer Interrupt gleiche oder niedrigere Priorität hat → Aktuelle Behandlung wird fortgeführt
* Wenn neuer Interrupt höhere Priorität hat → Neuer Interrupt wird zu erst behandelt

8. Was versteht man unter „Speicherhierachie“?

* Anordnung von Speichermedien absteigend sortiert nach Schnelligkeit des Zugriffs

9. Erläutern Sie das Cache-Prinzip!

* Nach Zugriff auf ein langsames Speichermedium werden die zuletzt gelesenen Daten im schnelleren Speicher gehalten ( → „Cache-Speicher“)
* Vor erneutem Zugriff wird geprüft, ob Daten bereits im Cache vorhanden sind → schnellerer Zugriff!
* Cache-Speicher ist unsichtbar für den Benutzer (Realisierung durch Hardware oder Betriebssystem)
* Ist vorteilhaft aufgrund des Lokalitätsverhaltens der meisten Programme (z. B. Schleifen etc.)

10. Was ist das Lokalitätsprinzip? Ist es stets erfüllt?

* Zeitliche Lokalität (temporal locality)
  + Wenn eine Adresse referenziert wurde, dann wird sie mit hoher Wahrscheinlichkeit bald wieder referenziert
* Räumliche Lokalität (spatial locality)
  + Wenn eine Adresse referenziert wurde, dann werden mit hoher Wahrscheinlichkeit die benachbarten Adressen bald referenziert
* Code wiederverwenden → Verringerung der Häufigkeit von Seitenladung
* Nein, abhängig wie das Programm sich verhält

Wie kann ich Lokalität erhöh en?

11. Wie schützt sich das BS gegen bösartige Benutzerprogramme? Wodurch unterstützt die HW diese Schutzmechanismen?

* Privilegierte Instruktionen
  + Schutz des vor fehlerhaften Programmen
  + → Verwendung Kernel-/Usermodusbit
  + → Nur BS ist im Kernel
* I/O-Zugriffsschutz
  + Schutz des Systems vor illegalen oder schädlichen I/O-Operationen
  + I/O-Operationen nur im Kernelmodus ausführbar → nur BS
  + Jede I/O-Operation in Anwendungen muss über System-Call laufen
* Speicherschutz
  + Schutz des Systems vor Zugriff auf unerlaubte Speicherbereiche
* CPU-Schutz
  + Wie kann garantiert werden, dass das Betriebssystem die CPU zurückerhält?
    - Timer + Interrupts

12. Wie funktioniert ein Systemaufruf (System Call)?

* Benutzung von Betribssystemdiensten des laufenden Programms
* Möglich Bibliotheks-Funktionen
* System Call erzeugt Software-Interrupt damit BS die CPU erhaählt
* Der Code aus dem Systemdienst wird über Register an Programm weitergegeben

13. Erläutern Sie die wichtigsten Betriebssystemdienste und geben Sie Beispiele für entsprechende System Calls an!

* Programmausführung
  + load, execute, end, abortproc
  + create
  + allocate
* Synchronisation und Kommunikation
  + wait for time, wait for event
  + send /receive
  + create / delete connection
* Dateiverwaltung
  + create, delete, open, close, read, write file
* Geräteverwaltung
  + request / release device
  + read, write, reposition
  + get devices attributes, set device attributes
* Allgemeine Information
  + get / set time or date
  + get / set system data

14. Welche möglichen Architekturmodelle für BS kennen Sie? Erläutern Sie jeweils die Grundidee!

* Monolithisch
  + BS aus einem Block
  + Ohne/Minimale Struktur
* Schichtenmodell
  + Unterste Schicht: Hardware
  + Nach außen hin: komplexer werdende Software
* Micro-Kernel
  + Minimaler Kern kommuniziert zwischen Benutzerprogramm und Managern (z. B. Prozessmanager)
  + und zwischen Hardware und Managern

15. Was ist das Prinzip der HW-Virtualisierung („virtuelle Maschine“)?

* Realisiert durch Hypervisor
  + Stellt alle Hardwareschnittstellen einer realen Maschine softwaretechnisch zur Verfügung
  + Privilegierte Maschinenbefehle eines Gast-Betriebssystems werden vom Hypervisor abgefangen und interpretiert

15a. Nennen Sie drei verschiedene Arten, um privilegierte Befehle eines Gast-Betriebssystems abzufangen!

* Typ-1-Hypervisor
  + Ein Hypervisor schirmt die komplette Hardware ab
  + Es gibt nur Gast-Betriebssysteme
* Typ-2-Hypervisor
  + Jedes Gast-Betriebssystem hat eigenen Hypervisor
  + Auf der Hardware läuft ein Host-Betriebssystem
* Typ-3-Hypervisor
  + Typ-1 – oder Typ-2- Hypervisor

15b. Erklären Sie den Unterschied zwischen Type-1-Hypervisor und Type-2-Hypervisor!

* Typ-1-Hypervisor hat nur Gast-Betriebssysteme / Typ-2-Hypervisor hat ein Host-Betriebssystem und weitere Gast-Betriebssysteme
* Typ-1-Hypervisor liegt direkt auf der Hardware / Typ-2-Hypervisor liegt auf dem Host-Betriebssystem

16. Welches Architekturmodell liegt Linux / Windows zugrunde?

* Linux:
  + monolithisch
  + mit Kernel-Modulen
* Windows
  + Modifizierter Microkernel

# Kapitel 2 - Prozesse

1. Was ist ein Prozess?

* Programm in Ausführung
* Eine Instanz eines Programms, das auf dem Computer gerade läuft
* Eine Einheit, der man den Prozessor zuweisen kann, und die vom Prozessor ausgeführt werden kann
* Eine Aktivitätseinheit, die beschrieben werden kann durch:
  + Einen Ausführungsfaden
  + Zustand
  + Zugewiesene Betriebsmittel

2. Wie sieht das typische Prozesslayout im Hauptspeicher aus? Erläutern Sie die einzelnen Bereiche!

Stack: Rücksprungadressen, lokale Variablen, Parameter

Daten: globale Variablen, Konstanten, dynamische Datenobjekte

3. In welchen Zuständen kann sich ein Prozess befinden? Wie sehen mögliche Übergänge aus?

4. Geben Sie einige typische Informationen an, die in einem Prozesskontrollblock (PCB) gespeichert werden!

* Allgemeine Register
* Priorität
* Prozess-ID
* Elternprozess
* Startzeit des Prozesses
* Zeiger auf Datensegemnt
* Benutzer-ID

5. Erläutern Sie den Ablauf eines Prozesswechsels von Prozess P0 auf Prozess P1!

Prozess 0 wird interruted  Seine Register werden in PCB0 gesichert  Der Scheduler wählt nächsten Prozess aus (hier Prozess 1)  Prozessor „bereinigen“  Register aus PCB1 laden  Prozess 1 beginnt zu rechnen

6. Erklären Sie die Prozesserzeugung unter Unix!

* Erster Prozess: init(PID 0)
* Rest: über fork als Kindprozesse

6a. Was macht der Systemaufruf fork, was waitpid?

* Fork: erstellt Kopie (Kind) des aktuellen Prozesses  Kopie, die eigenständig läuft
* Waitpid: Aktueller Prozess wartet so lange bis Kindprozess beendet ist

b. Geben Sie eine einfache Implementierung einer Shell an.

while (TRUE) {/\* Endlosschleife \*/

type\_prompt();/\* Prompt ausgeben \*/

read\_command(&command, &params); /\* Eingabezeile von Tastatur lesen \*/

PIDstatus = fork();/\* Kind erzeugen \*/

if (PIDstatus < 0) {

printf("Unable to fork");/\* Fehlerbedingung \*/

continue;/\* Schleife wiederholen \*/

}

if (PIDstatus > 0) {

waitpid(PIDstatus, &status, 0); /\* Elternprozess wartet auf Kind \*/

} else {

execve(command, params, 0); /\* Das Kind-Programm ausführen \*/

}

}

7. Was ist ein Thread?

* Prozess-„Faden“
* Leichtgewichtiger Prozess
* Unabhängiger Subprozess, der innerhalb eines normalen Prozesses läuft
* Verfügt über einen eigenen Programmzähler, Stack sowie Zustandsinformationen (eigener „Ausführungsfaden“)
* Alle Threads eines Prozesses laufen in demselben Adressraum, sie können deshalb globale Variablen gemeinsam nutzen
* Ein Prozess kann mehrere Threads enthalten
* Threads laufen wie die normalen Prozesse asynchron ab
* Threads können sich ebenso in einem der Zustände rechnend, bereit oder blockiert befinden
* Alle Threads haben Zugriff auf die Betriebsmittel des Prozesses (geöffnete Dateien, Kindprozesse, Signale usw.)

8. Erklären Sie den Unterschied zwischen einem Prozess und einem Thread!

* Thread sind Subprozesse eine Prozesses
* Threads innerhalb eines Prozesses haben einen Datenbereich
* 2 Prozesse haben immer unterschiedliche Datenbereiche  keine gemeinsamen globalen Variablen

8a. Welche Daten werden für jeden Thread gesondert gespeichert, auf welche Daten können alle Threads eines Prozesses gemeinsam zugreifen?

* Gesondert: Lokale Variablen
* Gemeinsam: Globale Variablen

8b. Gibt es pro Thread einen Stack? Begründen Sie Ihre Antwort.

* Ja

c. Welche Vorteile haben Threads gegenüber der ausschließlichen Verwendung von Prozessen?

* Eignen sich zur Parallelisierung
* Wechsel zwischen Thread und Thread innerhalb Prozess schneller als zwischen Prozessen

9. Was ist Kernel-Level / User Level Thread Scheduling? Diskutieren Sie Vor- und Nachteile des Kernel-Level-Scheduling von Threads!

* Kernel Level Threads
  + Threads aller Prozesse sind gleichwertig
  + Verwaltung einer vollstädnigen Thread-Tabelle durch den Kernel
  + Nach einem blockierenden Thread (z. B. E/A-Anforderung) kann direkt ein anderer Thread desselben Prozesses aktiv werden
* User Level Threads
  + Der Kernel kennt keine Threads
  + Verwaltung einer Thread-Tabelle durch den Benutzerprozess
  + Ein blockierender Thread blockiert den gesamten Prozess und damit alle anderen Threads dieses Prozesses

Vor- und Nachteile des Kernel-Level-Scheduling ????

10. Mit welcher Methode wird ein Java-Thread gestartet? Mit welcher Methode von außen unterbrochen?

* start()  führt im Thread die Methode run()
* interrupt() unterbricht den Thread

11. Erläutern Sie die Begriffe „preemptive“ und „non-preemptive“!

* preemptive  Prozesse können unterbrochen werden
* non-preemptive  Prozesse können nicht unterbrochen werden

12. Welche Ziele verfolgen alle Scheduling-Algorithmen?

* Alle Systeme
  + Fairness – jeder Prozess bekommt Rechenzeit der CPU
  + Policy Enforcement – Strategien werden sichtbar durchgeführt
  + Balance – alle Teile des Systems sind gleichmäßig ausgelastet
* Stapelverarbeitungssysteme
  + Durchsatz – maximiere nach Jobs pro Stunde
  + Turnaround-Zeit – minimiere die Zeit vom Start bis zur Beendigung
  + CPU-Belegung – belege die CPU konstant mit Jobs
* Interaktive Systeme
  + Antwortzeit – antworte schnellstmöglich auf Anfragen
  + Proportionalität – auf die Bedürfnisse des Nutzers eingehen
* Echtzeitsysteme
  + Meeting Deadlines – keine Daten verlieren
  + Predictability – z. B. Qualitätsverlust bei Multimedia vermeiden

13. Erläutern Sie Prozess-Scheduling – Algorithmen für Stapelverarbeitungssysteme!

* First-Come First-Served
  + Der Prozess der am längsten wartet bekommt CPU
  + Eine Warteschlange für Prozesse
* Shortest Job First
  + Der Prozess der, die kürzeste Jobdauer hat, wird zuerst bedient
* Shortest Remaining Time Next
  + Der Prozess der, die kürzeste verbleidende Jobdauer hat, wird zuerst bedient
  + Wie SJF nur unterbrechend

13a. Welcher Algorithmus ist “optimal” bzgl. der mittleren Gesamtwartezeit, wenn mehrere Jobs gleichzeitig anstehen? Warum?

* SJF da ich vorher planen kann und dadurch

b. Was macht man bei unterschiedlichen Ankunftszeiten?

14. Erläutern Sie Prozess-Scheduling – Algorithmen für Interaktive Systeme!

15. Erläutern Sie Prozess-Scheduling – Algorithmen für Echtzeitsysteme!

* Round-Robin
  + Festgelegte Zeitscheibe
  + Jeder Prozess bekommt Zeitscheibe  nach Beendigung warten bis alle anderen Zeitscheiben abgelaufen sind
* Scheduling mit Prioritätsklassen
  + Jeder Prozess gehört zu einer Prioritätsklasse
  + Prioritätsklasse wird ständig neu bestimmt, an Hand von Eigenschaften des Prozesses
  + Innerhalb einer Prioritätsklasse gibt es eine Warteschlange, die über Round-Robin geschedult wird

16. Erläutern Sie Prozess-Scheduling – Algorithmen für Multiprozessorsysteme!

* Time-Sharing mit gemeinsamen Prioritäts-Warteschlangen
  + Eine Prioritätswarteschlange für alle Kerne
  + Sobald CPU frei  Auswahl des Prozess mit der höchsten Priorität
* Time-Sharing mit eigenen Prioritäts-Warteschlangen
  + Jeder Prozess/Thread hat feste CPU
  + Jeder CPU hat eine eigene Prioritätswarteschlange
  + Nur wenn Warteschlange komplett leer, dürfen Prozesse „gestohlen“ werden
  + Dadurch können Cache-Blöcke erhalten werden
  + Dadurch trotzdem gleichmäßige Auslastung
* Gang-Scheduling
  + Zusammengehörige Threads werden als Gruppe (Gang) geschedult
  + Alle Gang-Mitglieder laufen gleichzeitig auf verschiedenen CPUs
  + Alle Threads auf allen CPU beginnen gleichzeitig

17. Welche Algorithmen verwenden Unix / Windows (Prinzip reicht)?

* UNIX
  + Multilevel-Feedback-Queuing mit Round-Robin
* Windows
  + Multilevel-Feedback-Queuing mit Round-Robin

# Kapitel 3 - Synchronisation und –Kommunikation

1. Erläutern Sie die grundsätzlichen Problemstellungen der Prozess-Synchronisation!

* Wechselseitiger Ausschluss (z. B. Für Zugriff auf gemeinsam benutzte Betriebsmittel)
* Ablaufsteigerung bei Abhängigkeiten (Einhalten von Reihenfolgebedingungen)

2. Was ist ein „kritischer Abschnitt“? Wie sieht ein allgemeingültiges Lösungskonzept für den wechselseitigen Ausschluss aus?

* Abschnitt in dem:
  + nur 1 Prozess zur Zeit
  + Prozessausführungen nicht unterbrochen werden dürfen

3. Welche Synchronisationskonzepte kennen Sie?

* Aktives Warten
* Semaphore
* Monitore

4. Was ist „Aktives Warten“?

* Prozess prüft ständig ob er kritischen Bereich betreten darf
* Abfragen ist aktiv → aktives Warten

4a. Geben Sie eine zuverlässige SW-Lösung an

* 2 Flags:
  + Prozess A will in den Kritischen Abschnitt
  + Prozess B will in den Kritischen Abschnitt
* Eine Variable die Auskunft darüber gibt, wer zuletzt den Zustand geändert

4b. Geben Sie eine zuverlässige HW-Lösung an

* Es muss eine Variable in einemMaschinenbefehl abgefragt und–wenn frei (Wert 0) –gesetzt werden (auf Wert 1)

booleanTestAndSet(int\* i) {

if(\*i == 0) {

\*i = 1;

return TRUE;

} else{

return FALSE;

}

}

4c. Welche Nachteile sind mit aktivem Warten verbunden?

* Verschwendet unnötig Prozessorzeit
* Wettrennen werden nicht verhindert
* Verhungern möglich
* Deadlock möglich

4d. Erläutern Sie ein sinnvolles Einsatzszenario!

* Systemprogramme für sehr kurze kritische Abschnitte

5. Was ist ein Semaphor?

* Semaphor ist ein Sperrmechanismus, der Prozesse unter bestimmten Bedingungen blockiert oder wieder freigibt

5a. Geben Sie Pseudocode zur Definition der Semaphor-Operationen an!

P(S): while(S == 0 ) {

<blockieren und warten>;

}

S--;

V(S) : S++ ;

if(< mindestens ein Prozess wartet auf S >) {

< wecke einen wartenden Prozess >;

}

5b. Was ist ein Mutex?

* Ein wechselseitiger von Prozesses über dem kritischen Bereich

5c. Erläutern Sie eine Semaphor-Lösung für das Problem des wechselseitigen Ausschlusses!

* Durch einen Puffer auf den Beide zugreifen

5d. Erläutern Sie eine Semaphor-Lösung für das Erzeuger-Verbraucher-Problem mit beschränktem Puffer!

* Der Verbraucher wartet (passiv) wenn der Puffer leer ist
* Der Erzeuger erzeugt wenn ein Verbraucher wartet
* Nach dem erzeugen weckt der Erzeuger den Verbraucher
* Erzeuger wartet (passiv) wenn kein Verbraucher wartet
* Wenn der Verbraucher nicht wartet und etwas im Puffer ist, verbraucht er und weckt danach den Erzeuger

e. Welche Vor- und Nachteile haben Semaphore? Diskutieren Sie!

* ???

6. Was ist die Grundidee eines SW-Monitors?

* Monitorbereich, in dem sich nur 1 Prozess befinden kann
* Alle die zugreifen wollen, warten in Monitorwarteschlange

7. Erläutern Sie die JAVA-Implementierung des Monitorkonzepts!

* Aufruf einer synchronize-Methode
* Freigabe dieser nach dem Methode „fertig“ durchgelaufen ist

7a. In welchen Zuständen (in Bezug auf einen Monitor) kann sich ein Thread befinden?

* Außerhalb des Monitors
* Bereit für Monitoreintritt
* Im Monitor rechnend

7b. Durch welche Ereignisse werden welche Zustandsübergänge ausgelöst?

* Außerhalb des Monitors  Bereit für Monitoreintritt: Aufruf einer synchronize-Methode
* Bereit für Monitoreintritt  Im Monitor rechnend: Kein anderer Prozess verwendet Methode + Prozess vorderstes Element der Warteschlange
* Im Monitor rechnend  Außerhalb des Monitors: Beendigung des Methodenaufrufs

7c. Skizzieren Sie eine Java-Monitorlösung für das Erzeuger-Verbraucher-Problem mit beschränktem Puffer!

public synchronized void enter(E item) throws InterruptedException{

while (buffer.size() == bufferMaxSize) {

/\* Puffer ist voll \*/

this.wait(); // Ausführenden Thread blockieren

}

buffer.add(item); // Datenpaket in den Puffer legen

this.notifyAll(); // ggf. Verbraucher wecken

}

public synchronized E remove() throws InterruptedException{

E item;

while (buffer.size() == 0) {

/\* Puffer ist leer \*/

this.wait(); // Ausführenden Thread blockieren

}

/\* Datenpaket aus dem Puffer holen \*/

item = buffer.removeFirst();

this.notifyAll(); // ggf. Erzeuger wecken

return item;

}

8. Was ist ein Deadlock?

* = Verklemmung
* Jeder Prozess aus der Mengen wartet auf ein Ereignis, das nur ein anderer Prozess der Menge auslösen kann

9. Geben Sie ein Beispiel für einen Deadlock an!

* Philosophenproblem:
  + 5 Philosophen am runden Tisch
  + 5 Gabeln
  + Jeder Philosoph benötigt 2 Gabeln zum Essen
  + Es wird immer zuerst die linke und dann die rechte Gabel genommen
  + Wenn alle gleichzeitig Hunger haben  jeder nimmt linke  jeder wartet bis rechte Gabel frei wird alle verhungern

10. Welches sind die notwendigen und hinreichenden Bedingungen für das Vorliegen eines Deadlocks?

* Wechselseitiger Ausschluss
* No Preemption
* Hold and Wait
* Circular Wait

11. Durch welche Maßnahmen werden in der Praxis Deadlocks vermieden?

* Vor jeder Betriebsmittelanforderung wird überprüft wie hoch die Deadlock-Möglichkeit

# Kapitel 4 - Hauptspeicher-Verwaltung

1. Was versteht man unter Relokation von Programmen?

* Verschiebbarkeit von Programmen

2. Was ist der Unterschied zwischen einer logischen und einer realen (physikalischen) Adresse?

* Logische Adresse: Adresse die auf eine reale Adresse referenziert
* Reale Adresse: Tatsächliche Adresse im Hauptspeicher

3. Was ist eine MMU und wie wird sie verwendet?

* = Memory Management Unit
* CPU  gibt logische Adresse  MMU  gibt reale Adresse  Hauptspeicher

4. Erläutern Sie die Adressumsetzung durch Basis- und Limitregister anhand eines Beispiels!

* Ich habe eine Array welches 5 Register umfasst:
  + Basisregister: 0
  + Limitregister: 4

5. Erläutern Sie die Adressumsetzung beim Compilieren und Laden eines C-Programms (bestehend aus 2 Sourcecode-Dateien) in den Hauptspeicher (mit Basisregistertechnik)!

* ???

6. Erläutern Sie hinsichtlich der Hauptspeicheraufteilung bei Multiprogramming die Konzepte „Feste Partitionierung“ und „Dynamische Partitionierung“!

* Feste Partitionierung: es gibt eine feste Anzahl von Partitionen (mit evtl. unterschiedlicher Größe
* Dynamische Partitionierung: es gibt eine beliebige Anzahl von Partitionen mit unterschiedlicher Größe, freie Partitionen können zu neuen zusammengefasst werden

6a. Welchen Vorteil hat die dynamische Partitionierung gegenüber der festen Partitionierung?

* Verschnitt wird minimiert

6b. Geben Sie drei Algorithmen für die Platzierung von Prozessen bei dynamischer Partitionierung an und vergleichen Sie deren Leistung!

|  |  |
| --- | --- |
| Next-Fit | Gleichmäßiger aus First-Fit  Langsamer  Umsortieren nötig |
| Best-Fit | Langsamstes Verfahren  Umsortieren nötig  Wenigster Verschnitt |
| First-Fit | Schnell  Hinten noch Platz für Große Prozesse |

6c. Welche Probleme verbleiben bei der dynamischen Partitionierung?

* Programm kann zu groß für Hauptspeicher sein
* Löcher im Speicher

7. Was bedeutet „Overlay-Technik“?

* Prozesse > Hauptspeicher in den Prozessor bekommen

8. Was versteht man unter „Swapping“?

* Ein- und Auslagerung eines kompletten Prozesses

9. Erläutern Sie die Konzeptidee des Virtuellen Speichers (Anforderungen)!

* Aufteilung des Hauptspeichers in viele (kleine) Partitionen
  + Feste Größe: Seiten ( Paging)
  + Unterschiedliche Größe: Segmente (\*)
* Jeder Prozess wird auf mehrere Partitionen verteilt!
* Partitionen können einzeln auf die Platte ausgelagert werden
* Jeder Prozess benutzt eigene logische („virtuelle“) Adressen
* Jeder virtuelle Adressraum kann größer als der physikalische Hauptspeicher sein (nur durch Plattenkapazität)

10. Erläutern Sie die Konzeptidee des „Paging“!

* Linearer Adressraum mit virtuellen Adressen
  + Virtuelle Adressen…
    - … beginnen für jeden Prozess bei 0
    - … werden fortlaufend durchnummeriert
    - … täuschen einen virtuellen Speicher vor
  + Aufteilung des virtuellen Adressraums in Seiten
    - Seite = Partition fester Größe
    - Jeder Seite wird ein zusammenhängender realer Speicherbereich zugeordnet, auch Seitenrahmen („Page Frame“) genannt
    - Abbildung von virtuellen auf reale Adressen (virtuelle Seiten  Seitenrahmen) durch eine Seitentabelle

11. Erläutern Sie die Aufgabe, Struktur und Verwendung einer **Seitentabelle** beim Paging-Verfahren (inkl. Adressabbildung)!

* Eine Seitentabelle pro Prozess
* Ein Eintrag beschreibt eine Seite (Index = virtuelle Seitennummer)
* Aufbau eines Eintrags in einer Seitentabelle:
  + Seitenrahmennummer
  + Valid-Bit: 1 = im Hauptspeicher, 0 = auf Platte ( Zugriff löst seitenfehler aus  Interrupt zum Einlesen von der Platte) [Present-/Absent-Bit]
  + Zugriffsrechte (Schreiben/Lesen)
  + Verwaltungs-Flags: Referenced-Bit (R-Bit), Modifiied-Bit (M-Bit) werden bei Lese-Schreib-Zugriffen (R-Bit) oder Schrieb-Zugriffen (M-Bit) auf die Seite von der Hardware gesetzt

12. Welche Lösungsmöglichkeiten gibt es, um große Seitentabellen effizient zu implementieren? Erläutern Sie die Konzepte!

* TLB
  + Kürzlich genutzte Daten in Cache Speicher
* Aufteilung einer Seitentabelle in mehrere Tabellen mit mehrstufigem Zugriff
  + Nur wenige Tabellen müssen zeitgleich im Speicher sein
* Invertierte Seitentabellen (warum invertiert?)
  + Adressumrechnung anders
  + TLB + alle physischen Seitenrahmen des Hauptspeicher in Invertierte Seitentabelle

13. Welche Seitenersetzungsalgorithmen kennen Sie? Erläutern Sie die Konzepte!

* FIFO
  + Es wird immer die Seite ersetzt, die am längsten im Hauptspeicher ist
  + Einfaches Verfahren
  + Verwaltet die Hauptspeicher-Seiten eines Prozesses in einer List (Kopf: älteste Seite, Ende: jüngste Seite)
  + Berücksichtigt nicht, ob die Seite gerade jetzt häufig benutzt wird
    - Könnte sofort nach Ersetzung wieder gebraucht werden
  + Ineffizientes Verhalten
* Least Recently Used (LRU)
  + Annäherung an optimale Strategie
  + Ersetzt die Seite, die am längsten nicht benutzt wurde
  + Nach dem Lokalitätsprinzip ist die Wahrscheinlichkeit hoch, dass die Seite auch in Zukunft lange nicht benutzt wird
  + Implementierung; Jede Seite müsste mit einem Zeitstempel für den letzten Zeitpunkt der Benutzung versehen werden
    - Das ist in der Realität aber extrem aufwendig
* Clock-Algorithmus
  + Ziel: Leicht implementierbare Annäherung an LRU
  + Führt eine zirkulare Liste („Clock“) mit Zeigern auf Seitentabelleneinträge für alle Seiten des Prozesses, die sich im Hauptspeicher befinden (mit Speicherung der letzten Zeigerposition)
  + Verwendet das Referenced-Bit (R-Bit)
    - Wenn die Seite geladen wird: R-Bit = 0
    - Wenn auf die Seite zugegriffen wird: R-Bit = 1 (durch die MMU)
  + Seitenfehler  Zeiger durchläuft die Clock-Liste ab der letzten Position (im „Uhrzeigersinn“)
    - Seite mit R-Bit = 1  R-Bit zurücksetzen auf 0
    - Erste gefundene Seite mit R-Bit = 0  Seite ersetzen (auch in der Clock-Liste)!

14. Welche Speicherzuteilungsstrategien kennen Sie? Erläutern Sie die Konzepte!

* Feste Seitenanzahl und lokale Strategie
  + Einem Prozess steht für seine Ausführung eine feste Anzahl von Hauptspeicherseiten zur Verfügung
  + Tritt ein Seitenfehler auf, dann muss das Betriebssystem entscheiden, welche andere Seite dieses Prozesses ersetzt werden muss
    - Das wesentliche Problem liegt in der Festlegung der Seitenzahl ( x % der gesamten Prozessgröße?)
* Variable Seitenzahl und globale Strategie
  + Den im Hauptspeicher befindlichen Prozessen stehen eine variable Anzahl von Sieten zur Verfügung
  + Seitenfehler  Zuweisung einer freien Seite, falls verfügbar
  + Keine freien Seiten mehr verfügbar  Seite zur Ersetzung auswählen (beliebiger Prozess!)
  + ZU viele Prozesse im Hauütspeicher  u. U. sind die zur Verfügnug stehenden Speicherbereiche nicht mehr ausreichend und die Seitenfehlerrate wird sehr groß
  + Jedes Programm, das zusätzliche Seiten benötigt, erhählt diese auf Kosten von Seiten anderer Programme. Da diese aber ebenfalls noch benötigt werden, werden in immer schnellerer Folge weitere Seitenfehler erzeugt. (Seitenflattern, engl. Thrashing). Die Prozesse verbrauchen dann für das Seitenwechseln mehr Zeit als für ihre Ausführung
* Variable Seitenzahl und lokale Strategie: Working Set-Strategie  Working Set – Modell
  + Der Working Set WS(t, δ) eines Prozesses zur Zeit t ist die Menge aller Seiten, die er bei den letzten δ Speicherzugriffen angesprochen hat.
  + Der Umfang der Working Set ist abhängig von δ. Im Extremfall umfasst der Working Set das gesamte Programm
  + Je größer der Working Set, umso geringer die Seitenfehlerrate

# Kapitel 5 - Externe Geräte & Dateisysteme

1. Was ist die Struktur und die Aufgabe eines „Controllers“?

* Aufgabe:
  + Schnittstelle: Kommunikation zwischen Prozessor/Hauptspeicher
  + Schnittstelle: Steuerung des Geräts
* Struktur:
  + Puffer
  + Register

2. Welche Techniken zur Signalisierung und zum Datenaustausch zwischen Controller und BS kennen Sie?

* Signalisierung
  + Registerzugriff und aktives Warten
  + Interrupts
* Datenaustausch
  + Das BS greift auf Controller-Register über spezielle Befehle zu
  + Contriller-Speicher wird in com BS reservierte Hauptspeicheradressen eingeblendet
* Direct Memory Access
  + Das BS übergibt Anfangsadresse und Größe des Hauptspeicherbereichs an den DMA-Controller
  + Der DMA-Controller steuert den Datentransfer zwischen Gerätecontroller und Hauptspeicher
  + Der DMA-Controller informiert das BS mittels Interrupt über das Ende der Operation

3. Was ist ein Device driver („Treiber“)?

* Eigenschaften
  + Herstellerspezifische Betriebssystemerweiterung
  + Steuert den Controller
* Aufgaben
  + Initialisierung des E/A-Geräts beim Systemstart
  + Vorbereitung einer E/A-Operation und Aktivierung des Geräts für den Datentransfer
  + Reaktion auf Geräte-Interrupts
  + Fehlerbehandlung
* Implementierung als
  + Eigener Prozess und/oder
  + Betriebssystem-interne Funktion (Aufruf aus Bibliothek)
* Einbindung in das BS durch
  + Integration in den Kernel (unix)
  + Laden aus Datei bei Systemstart (Windows, Unix)
  + Laden zur Laufezit (z. B.  USB/FireWire-Unterstützung)

4. Welche Funktionen hat ein Dateisystem?

* Bereitstellung von Dateioperationen
* Verwaltung von Verzeichnissen („Directories“), um Ort und Eigenschaften aller Dateien zu speichern
* Überprüfung von Zugriffsrechten
* Abbildung von logischen Dateizugriffen (Dateioperationen) auf physikalische Einheiten (Blöcke)
* Verwaltung von Plattenblöcken  externe Datenträger

5. Welche Möglichkeiten gibt es, Dateien zu strukturieren?

* Byte-Folge
* Folge von Datensätzen
* Baum

6. Was ist ein Verzeichnis (Directory)?

* Leistet die Abbildung zwischen den Dateinamen und den entsprechenden Dateien des Verzeichnisses
* Ist selbst eine Datei (mit einer speziellen Funktion)
* Enthält eine Liste mit Verwaltungsinformationen über alle Dateien
  + Name, Speicherort, Attribute
* Ist bei den meisten Dateisystemen Teil einer hierarchischen Baumstruktur

7. Mit welchen „Belegungsmethoden“ kann ein BS verwalten, welche Blöcke zu einer Datei gehören? Erläutern Sie die verschiedenen Ansätze!

* Zusammenhängende Speicherung
  + Alle Blöcke einer Datei werden in physiklaisch zusammenhängenden Speicherbereichen abgelegt
  + Der Dateieintrag im Verzeichnis enthält die Startadresse des ersten Blocks und die Anzahl der kontinuierlich belegten Blöcke
* Verkettete Listen
  + Die Blöcke einer Datei können auf der Platte verstreut sein (nicht zusammenhängend gespeichert)  Verkettung der Blöcke ist nötig
  + Das erste Wort eines jeden Plattenblocks wir dlas Zeiger auf den nächsten Block der Datei benutzt (enthält dessen Plattenblockadresse)
  + Der Dateieintrag im Verzeichnis enthält einen Ziger auf den ersten Block der Datei: („Datei A“, 4), der letzte Block hat den Zeiger -1
* Verkettete Listen mit Zeigertabelle: File Allocation Table (FAT)
  + Idee: „Herausziehen“ der Zeiger aus den Plattenblöcken
  + Alle Plattenblöcke werden durch eine sperate Zeigertabelle im Hauptspeicher beschrieben („File Allocation“ = Fat) mit Tabellenindex = Plattenblockadresse
    - Der Verzeichniseintrag für eine Datei enthhält die erste Plattenblockadresse = Index des ersten Eintrags in der FAT
    - Jeder Eintrag in der FAT ist die Aresse des nächsten Plattenblocks
    - Der FAT-Eintrag für den letzten Block enthält -1 oder EOF (= „End Of File“)
* Belegung der separate Zeigerlisten
  + Idee: Alle Zeiger auf die Blöcke einer Datei (Plattenblockadressen) werden in einer eigenen Datenstruktur (I-Node) zusammengefasst
  + Der Dateieintrag im Verzeichnis enthhält nur den Dateinamen und die Adresse des I-Nodes dieser Datei
  + Der i-te Eintrag in der I-Node –Zeigerliste zeigt auf den i-ten Block der Datei
  + Falls bei fester Größe des I-Node die Anzahl der Einträge nicht ausreicht, kann auf eine weitere Liste verwiesen werden

8. Was sind die Vorteile von I-Nodes gegenüber einer FAT?

* Nur die benötigten I-Nodes müssen im Hauptspeicher liegen, nicht die gesamte FAT (Problem bei großen Festplatten!)
* Schnellerer zugriff auf weiter hinten liegende Blöcke
* Geringere Auswirkung von Lesefhlern (i.d.R. ist nur ein Block/ eine Datei betroffen)
* Einrichten von „Hard Links“ ist möglich (mehrere Verzeichniseinträge für die dieselbe Datei)
* Direkte Zuordnung von Datei-Informationen ist möglich

9. Was passiert beim Bootvorgang?

* Das BIOS (im ROM) liest den MBR ein und führt das dort gespeicherte Programm aus
* Das MBR-Programm ermittelt aus der Partitionstabelle die aktive Partition, lädt deren Bootblock in den Hauptspeicher und führt ich aus
* Das Bootblock-Programm lädt das Betriebssystem, welches in der Partition enthalten ist
* Das BS holt sich sämtliche Informationen über das Dateisystem aus dem Superblock (Typ des Dateisystems, Anzahl Blöcke etc.)

~~10. Welche UNIX-Prüfalgorithmen zur Konsistenzprüfung eines Dateisystems kennen Sie? Erläutern sie deren Arbeitsweise.~~

~~11. Welche Ziele verfogen RAID-Systeme?~~

~~12. Erläutern Sie das Prinzip von RAID Level 0, 1, 4, 5 anhand eines Beispiels!~~