# 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
* Ist es stets erfüllt ?????

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

**Stack**

**frei**

**Daten ("Heap“)**

**Programmcode**

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.

* ???

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?

* SRTN????

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?

7. Erläutern Sie die JAVA-Implementierung des Monitorkonzepts!

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

b. Durch welche Ereignisse werden welche Zustandsübergänge ausgelöst?

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

8. Was ist ein Deadlock?

9. Geben Sie ein Beispiel für einen Deadlock an!

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

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