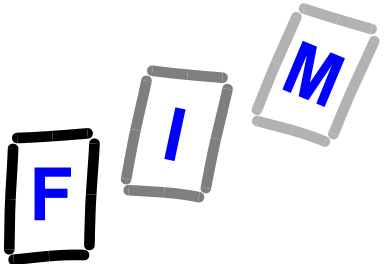
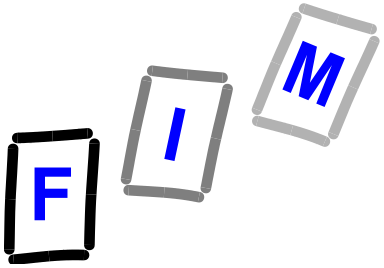


Betriebssysteme

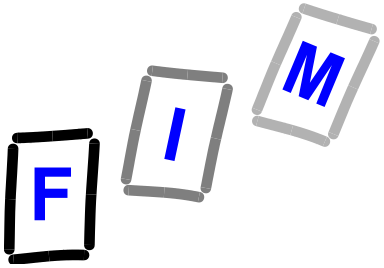
G: Parallele Prozesse

(Teil A: Grundlagen)



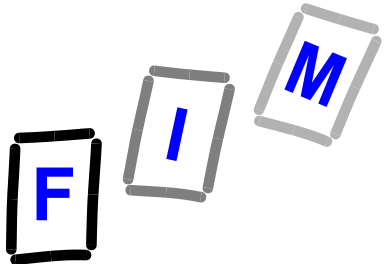
Prozesse

- **Bei Betriebssystemen stoßen wir des öfteren auf den Begriff „Prozess“**
  als wahrscheinlich am häufigsten verwendeter und am unklarsten definierter Fachausdruck!
- **Folgende Sicht:**
  - ➡ **Ein Prozess ist durch eine Menge von Registern / Hauptspeicherstellen bestimmt, welche ihre Werte nach bestimmten Regeln ändern.**
  - ➡ **Diese Regeln sind zumeist in einem Programm festgelegt.**
  - ➡ **Das Programm wird von einem Prozessor ausgeführt.**



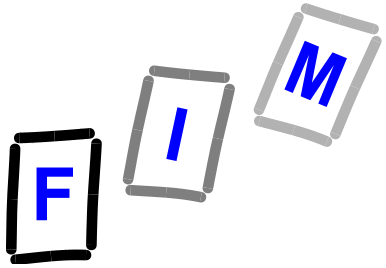
Prozesse

- **Unterscheide:**
  - ➡ **Nicht-deterministische Prozesse**
  - ➡ **Deterministische Prozesse**
- **Ein Prozess ist deterministisch (bezogen auf Ein-/Ausgabe), wenn seine Ausgabe alleine/nur von seiner Eingabe abhängig ist:**
  - ➡ **Sicherstellung der Wiederholbarkeit**
  - ➡ **Wird ein P mit selbem Input wieder gestartet, liefert P wieder dasselbe Ergebnis**
    - » **Real gar nicht so einfach …**



Sequentielle Prozesse

- ***Sequentieller* Prozess:**
  Zu jedem Zeitpunkt t wird genau eine Instruktion ausgeführt.
  - ➡ **Natürlich ist diese Sicht vom Abstraktions-grad abhängig**
    - » **ZB Methode, Anweisung, Maschinencode, Pipeline-Stufe**



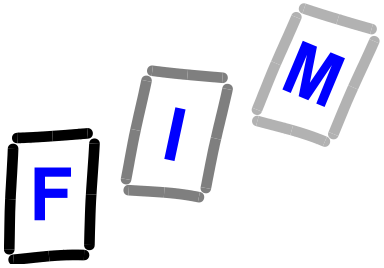
Gleichzeitig - Parallel

- **Die Begriffe „gleichzeitig“ (engl.: concurrent) und „parallel“ werden in der Fachliteratur nicht genau unterschieden.**

Allgemeine Definition: gleichzeitig - parallel

Zwei Prozesse laufen parallel (gleichzeitig), wenn die erste Instruktion des einen Prozesses ausgeführt wird, bevor die letzte Instruktion des anderen Prozesses beendet wurde.

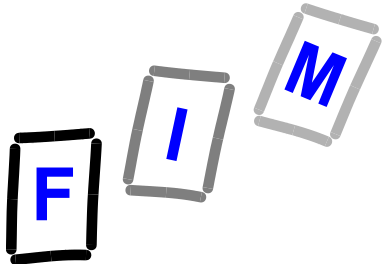
- **Anmerkungen:**
  - ➡ **Diese Definition ist unabhängig von der Anzahl der verfügbaren Prozessoren (CPUs).**
  - ➡ **Sie stimmt auch für eine CPU:**
    Die (eine) Überschneidung auf der Zeitachse ist entscheidend.



Gleichzeitige Prozesse

- **Überschneiden sich zeitlich.**
- **Werden als unabhängig (engl.: disjoint) bezeichnet, wenn jeder Prozess nur auf eigene (engl.: private) Daten zugreift.**
- **Sind abhängig / kooperierend (engl.: interacting) bezeichnet, wenn sie auf gemeinsame (engl.: shared) Daten zugreifen.**
  - ➡ **Beachte wieder: Granularität!**
    - » **„Selbe Festplatte“ vs. „Selbe Datei“ vs. „Selbes Byte in selber Datei“**

Per Brinch Hansen, 1973



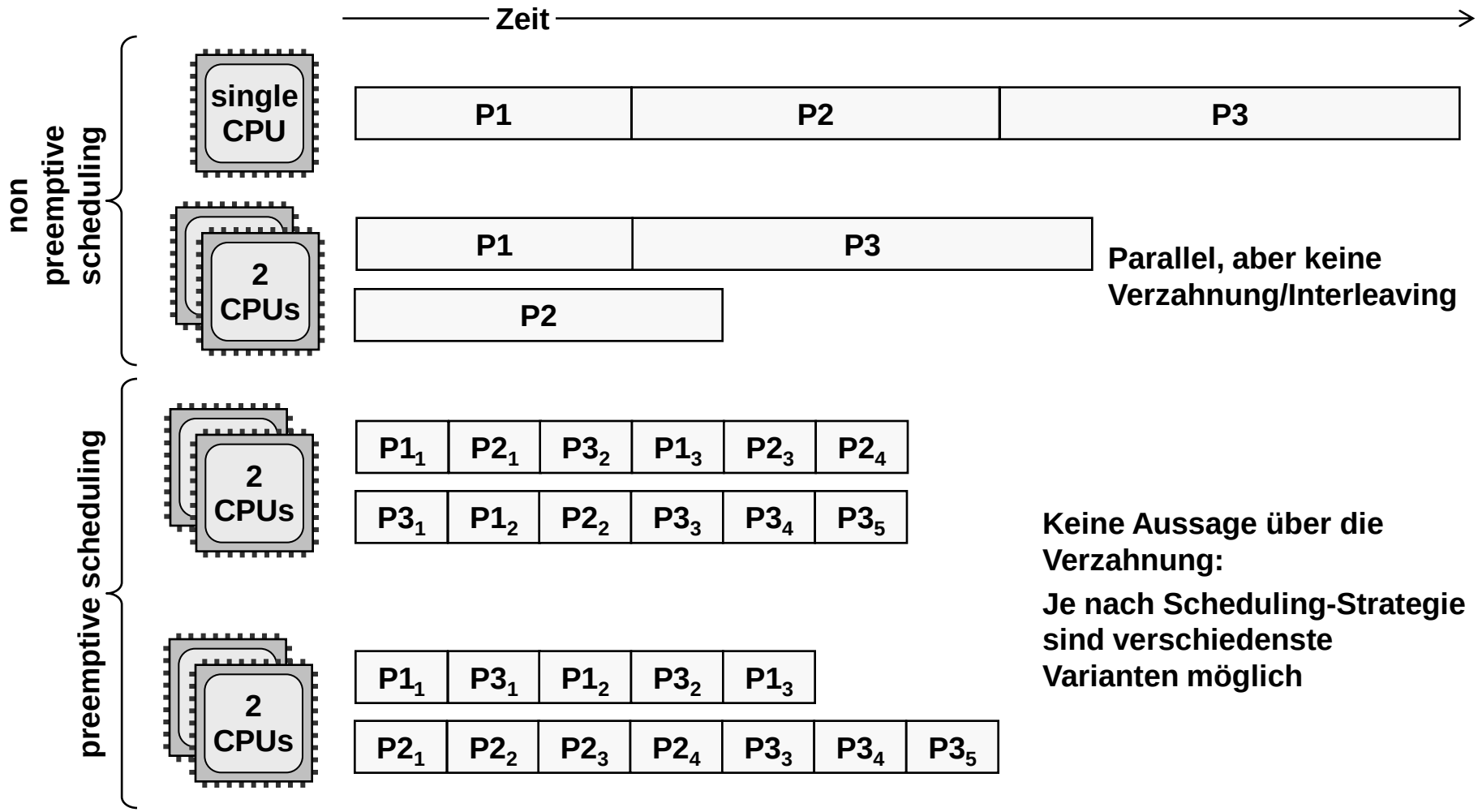
cobegin
coend

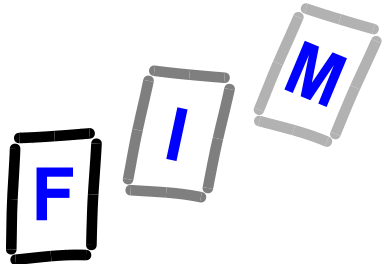
Zur Darstellung von Gleichzeitigkeit / Parallelität verwendet man zB folgende Notation:

```
COBEGIN
    P1;
    P2;
    ...;
    Pn;
COEND;
```

P_1, ..., P_n laufen dabei zueinander parallel ab

Verzahnung (engl.: interleaving)





Keinerlei Annahmen über die Reihenfolge

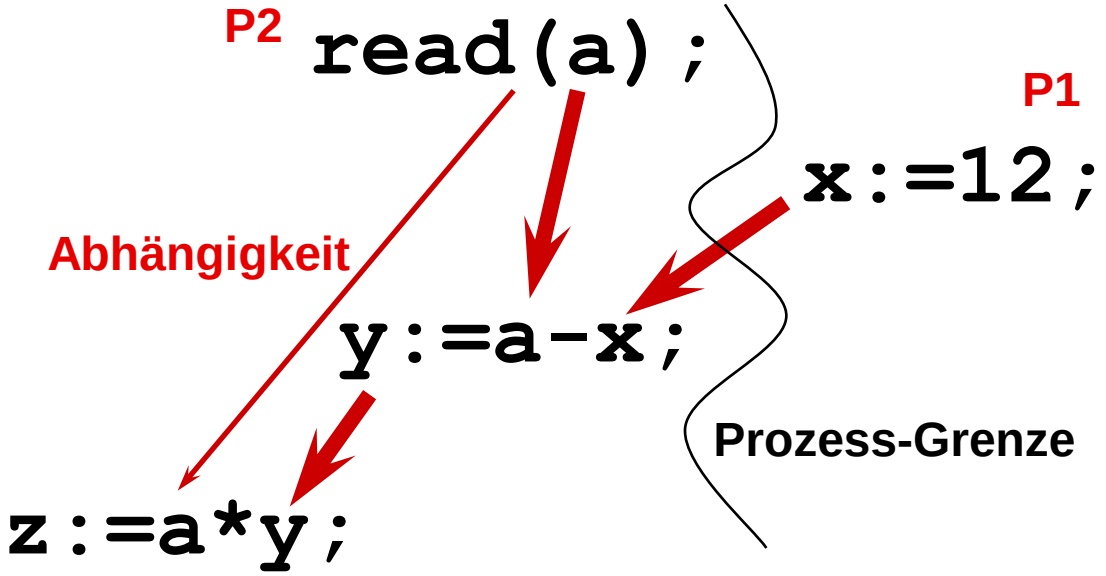
- Seien A=<a1;a2> und B=<b1;b2> zwei sequentielle Prozesse, die zueinander parallel ablaufen.

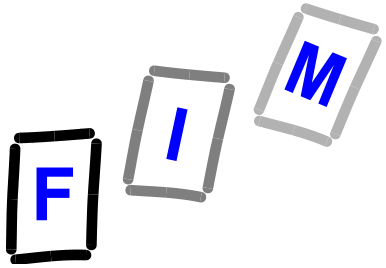
Dann sind folgende Reihenfolgen des Ablaufes möglich:

<a1;a2;b1;b2>	<b1;b2;a1;a2>
<a1;b1;b2;a2>	<b1;a1;a2;b2>
<a1;b1;a2;b2>	<b1;a1;b2;a2>

Ablauf-Reihenfolge

- Eine spezielle Ablauffolge ist notwendig, wenn P1 Daten erzeugt, die von P2 verwendet werden.
- Dies ist aber nicht immer notwendig.

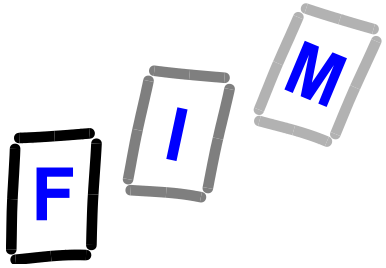




Gründe für parallele Prozesse (1)

Überblick:

- **Gemeinsame Verwendung (engl. sharing) von Informationen und Ressourcen**
- **Schnellere Berechnungen**
  - ➡ **Gleichzeitig, zB mehrere CPUs/VMs**
  - ➡ **Interleaving, zB gleichzeitige Internet-Anfragen, die erst nach einigen Sekunden beantwortet werden**
    - » **Wartezeiten sinnvoll nutzen**
- **Modularität: Siehe zB Webservices**
- **Komfort**

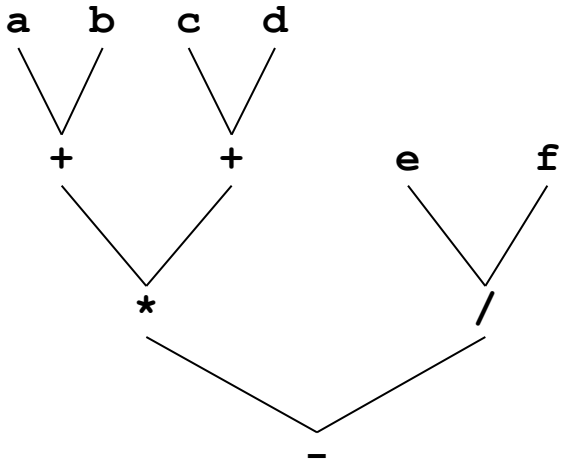


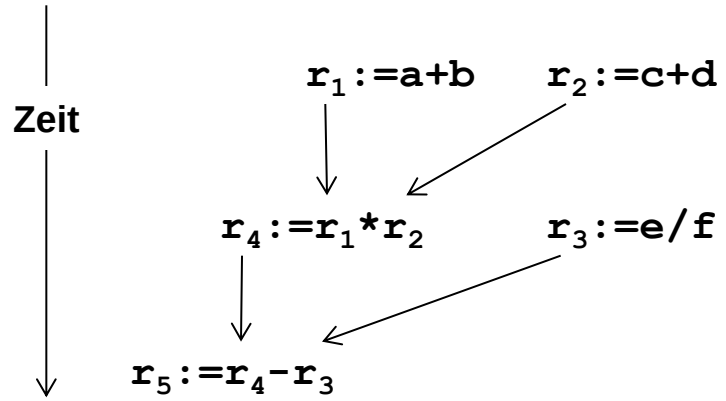
Gründe für parallele Prozesse (2)

- **Gemeinsame Verwendung (engl.: sharing) von Informationen und Ressourcen**
  - ➡ **Mehrere Prozesse/Benutzer können an der gleichen Information (und damit auch ev. an den gleichen Ressourcen) interessiert sein**
  - ➡ **Z.B. gemeinsame Nutzung von Speicherplatz oder von Dateien**
- **Die Kommunikation und der Datenaustausch zwischen Prozessen werden über gemeinsame (engl.: shared) Variablen (Speicher) durchgeführt.**

Gründe für parallele Prozesse (3)

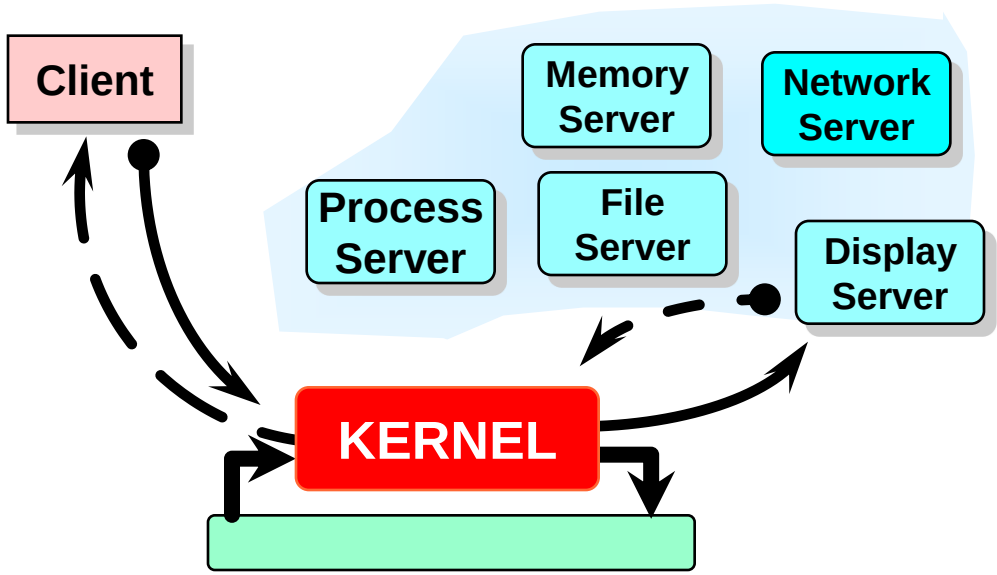
- **Schnellere Berechnungen (speed up)**
  - ➡ **Aufteilung einer Aufgabe in Teilaufgaben,**
  - ➡ **welche dann z.T. „echt“ parallel zueinander ausgeführt werden.**
  - ➡ **Voraussetzungen: >1 CPU, Ein-/Ausgabe-Kanäle, ...**





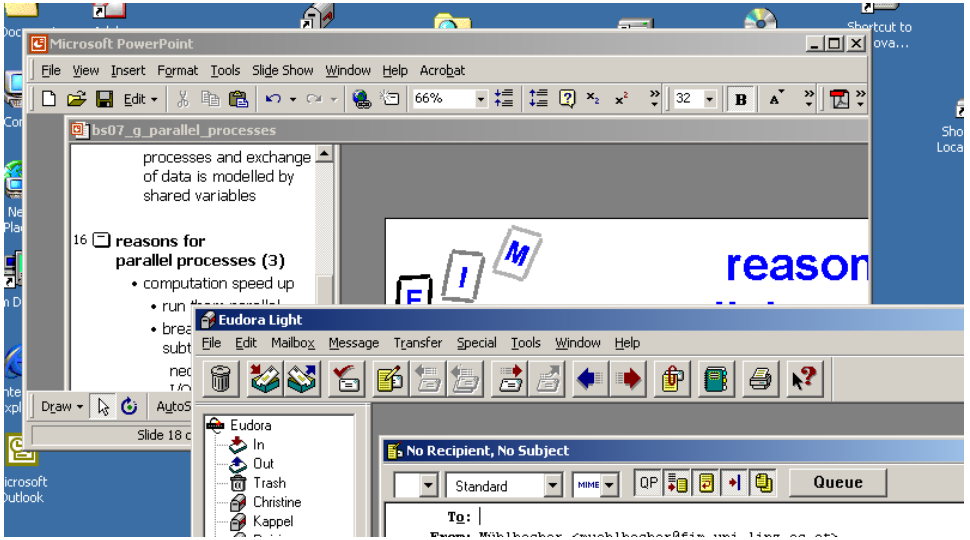
Gründe für parallele Prozesse (4)

- **Modularität**
  - ➡ **Vergleiche: Module eines Softwaresystems**
  - ➡ **Teile ein Gesamtsystem (z.B. das BS!) in einzelne unabhängige und kooperierende Prozesse.**

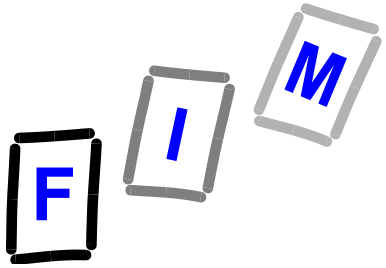


Gründe für parallele Prozesse (5)

- **Komfort / Benutzerfreundlichkeit**
  - ➡ **Anwender kann mehrere Aufgaben zur gleichen Zeit ausführen lassen**
    - » **Text editieren**
    - » **Ausdruck**
    - » **Senden / Empfangen von Emails**
    - » **...**

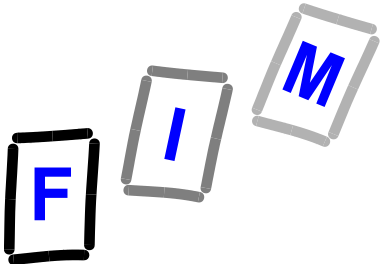


- ➡ **Siehe auch Services/Dienste!**



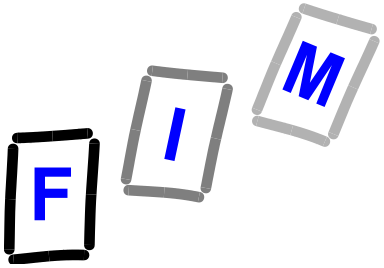
Wechsel des Prozesskontextes

- Während der Ausführung eines Prozesses werden diesem Ressourcen zugeordnet
  - ➡ Speicherbereiche (z.B. Daten des Prozesses)
  - ➡ Dateien
  - ➡ Zugriffsrechte, ...
- Der Wechsel der CPU auf einen anderen Prozess bedingt einen „Wechsel des Prozesskontextes“ (engl.: context switch).
  - ➡ Reiner Overhead
  - ➡ Zeitaufwendig



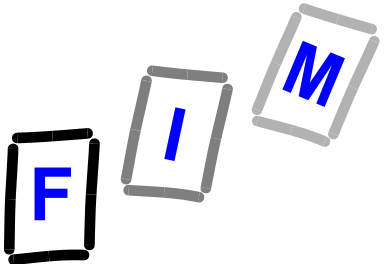
Threads

- Wenn die CPU zwischen Prozessen umschalten muss, welche sich Speicherbereiche und andere Ressourcen teilen, lässt sich dieser Aufwand reduzieren durch
  - ➡ leichtgewichtige (light-weight) Prozesse,
  - ➡ auch „Threads“ genannt.
- Ein normaler (schwergewichtiger) Prozess ist ein Task mit nur einem Thread.
- Jede Task (Prozess) besteht aus zumindest einem Thread.
- Unterschied:
  - ➡ Prozess: Eigener Speicherbereich
  - ➡ Thread: Alle Threads eines Prozesses teilen sich Speicher



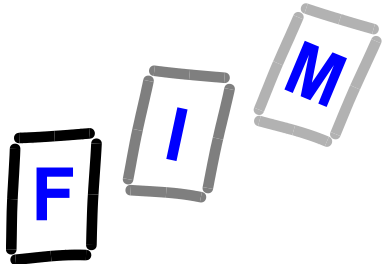
Scheduling zwischen Threads

- **Threads verhalten sich in vieler Hinsicht wie Prozesse:**
  - ➡ **Sie teilen sich die CPU.**
  - ➡ **Status: bereit, rechnen, warten, beendet.**
- **Moderne BS führen die CPU-Aufteilung (auch) auf Ebene der Threads durch.**
  - ➡ **Z.B. Prioritätssteuerung (engl.: priority scheduling) zwischen Threads.**
    - » **Die Threads eines Prozesses besitzen anfangs die gleiche Priorität**
  - ➡ **Time-Sharing zwischen Threads gleicher Priorität.**



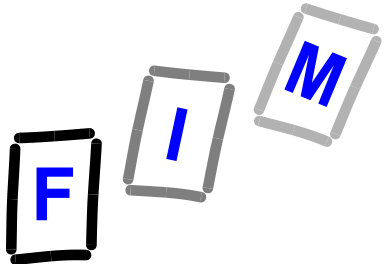
Erzeugen von Threads in JAVA

- **Das Programm erzeugt zwei Threads**
  - ➡ **Haupt-Thread**
    - » **Macht nichts anderes, als den zweiten Thread *Thread1* zu starten.**
    - » **Main1.java ist dabei die Hauptklasse, welche die Methode *main()* enthält.**
  - ➡ **Der zusätzlich gestartete Thread erweitert die Thread-Klasse:**
    - » **Wird durch *t1 = new Thread1* erzeugt und mittels *t1.start()* gestartet.**
    - » **Schreibt dann einfach Text auf den Bildschirm.**



Erzeugen von Threads in JAVA (1)

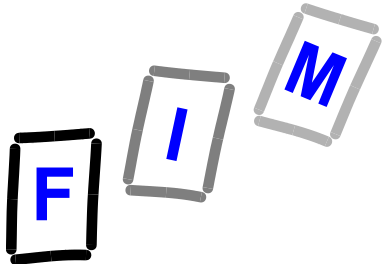
```
Main1.java
public class Main1 {
  public static Thread1 t1;
  // Deklaration der Thread-Variablen
  public static void main(String args[ ]) {
    t1=new Thread1(); // Erzeuge Thread
    t1.start();       // Starte Thread
    /* Neu / Modifikation! */
    while (true) {
      System.out.println("Main");
     } /* while */
  } /* main */
} /* Main1 */
```



Erzeugen von Threads in JAVA (2)

Thread1.java

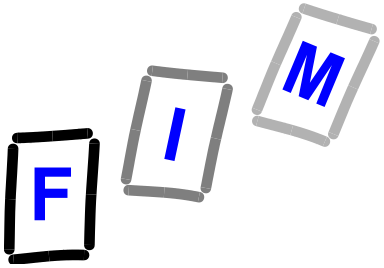
```
class Thread1 extends Thread {

  public void run(){

    /* Neu / Modifikation! */
    while (true) {
      System.out.println("Thread1");
    } /* while */

  } /* run */

} /* Thread1 */
```



Erzeugen von Threads in JAVA (3)

- **Ausgabe: Nicht deterministisch**
  - ➡ **Warum? Gemeinsam genutzte Ressource (Terminal), aber kein Synchronisation!**
- **Beispiel:**
  - ➡

```
Main
Main
Main
Thread1
Thread1
Main
Main
```



Threads bei JAVA VM

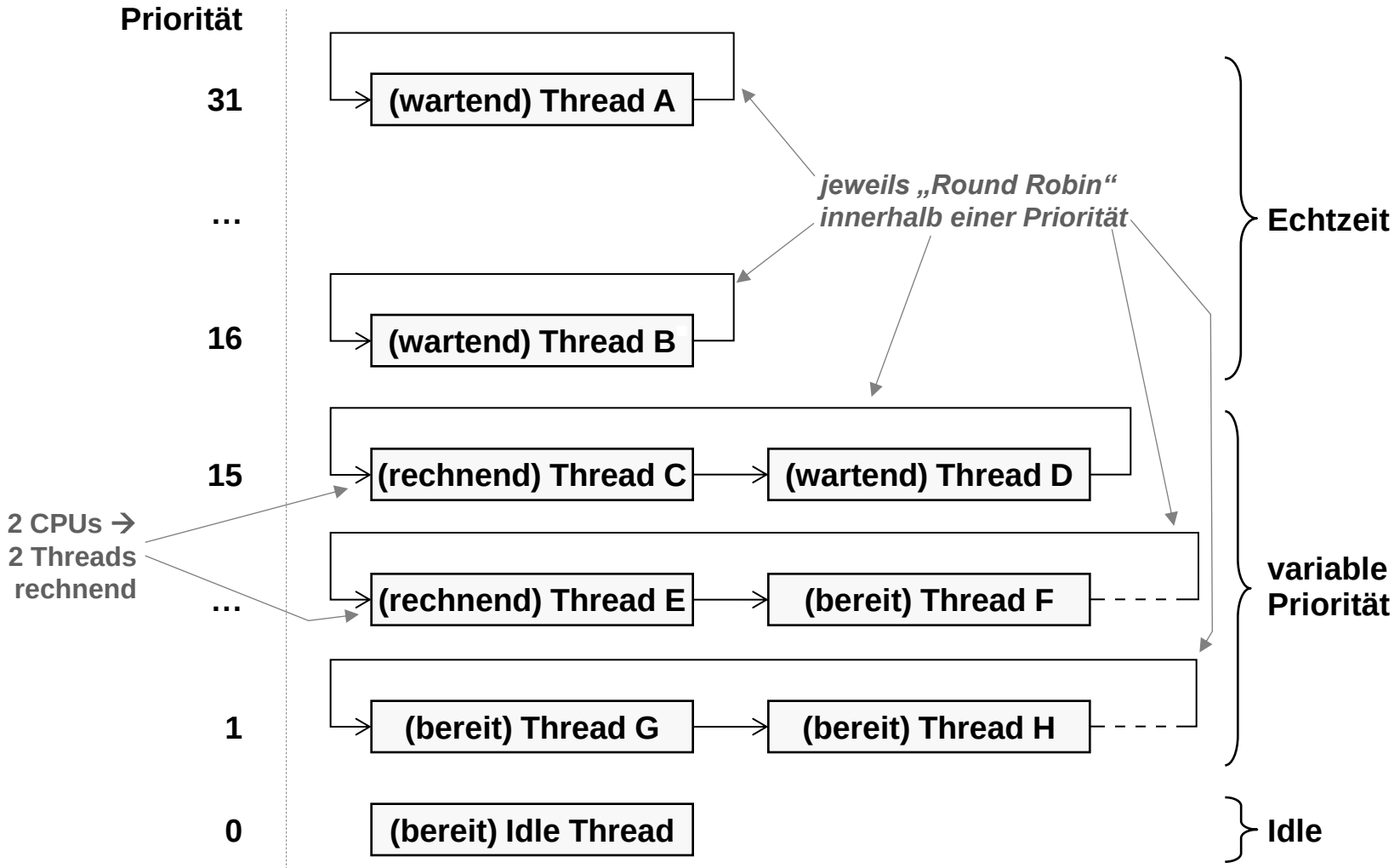
- **Green Threads: Scheduling von der JVM autonom und unabhängig vom OS**
  - » **Heute nicht mehr üblich**
  - » **Eigener Scheduler innerhalb JVM notwendig**
- **Native Threads: Werden auf die Threads im Host-System „aufgesetzt“**
  - » **Java-Spezifikation:**

    Keine Aussage über Scheduling von Threads gleicher Priorität

    Daher bei Native Threads:

    Preemptive scheduling des Host-Systems wirksam

Thread-Prioritäten und Scheduling in XP-Familie



„Ungerechte/gerechte(re) Situation“