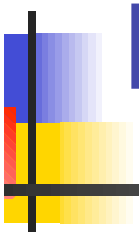


Betriebssysteme

Threads

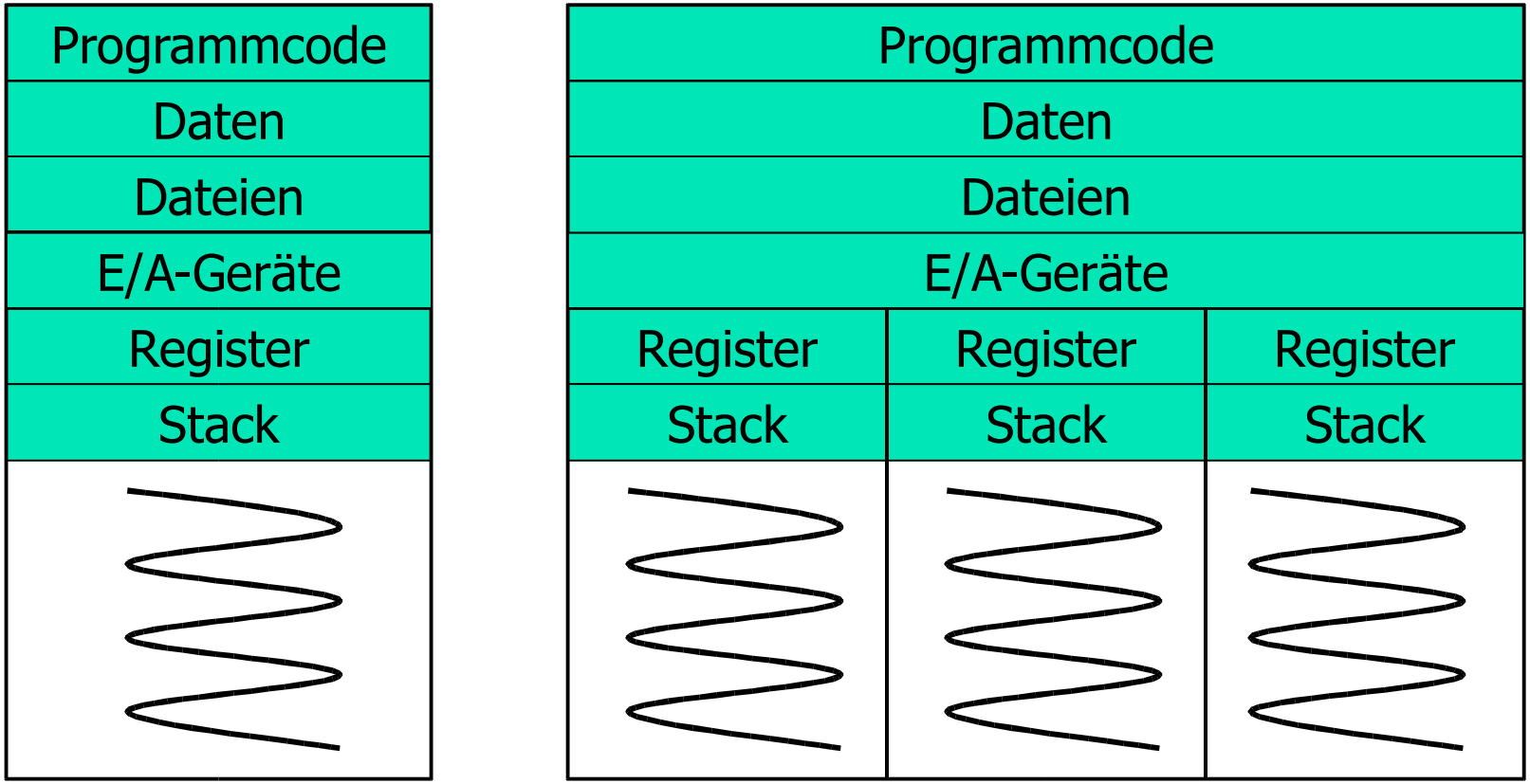
Überblick

- Prozesse versus Threads
- Beispiele für Multithreading
- Vorteile von Multithreading
- Anwender- und Kernel-Threads
- Multithreading-Modelle
- Implementierungsaspekte
- Solaris-2-Multithreading

Prozesse und Threads (I)

- **Prozess**
  - Ein in Ausführung befindliches Programm
  - Benötigt Ressourcen: Prozessor, Speicher (Programmcode, Daten, Stack), Dateien, E/A-Geräte
  - Bislang betrachtet: sequentiell arbeitende Prozesse (nur ein Ausführungsstrang)
- **Thread**
  - Ein Ausführungsstrang innerhalb eines Prozesses
  - Benötigt: Prozessor, eigenen Stack
  - Nutzt: Programmcode, Daten, Dateien, E/A-Geräte des Prozesses
  - Mehrere Threads innerhalb eines Prozesses möglich

Prozesse und Threads (II)



Ein Prozess mit einem Thread Ein Prozess mit drei Threads

Beispiele für Multithreading

- Anwendungen mit graphischer Benutzeroberfläche, z.B. Textverarbeitung:
  - Texteingabe
  - Rechtschreibprüfung
  - Ausdruck
- Serversoftware, z.B. Webserver, DB-Server:
  - Administration
  - Simultane Bearbeitung vieler Anfragen

Vorteile von Multithreading

- **Kürzere Antwortzeiten**
  Bei interaktiven Anwendungen kann auch auf Benutzereingaben reagiert werden, während andere, langandauernde Aufgaben durchgeführt werden.
- **Gemeinsame Nutzung von Ressourcen**
  Auf gemeinsamen Speicher sowie gemeinsame Dateien und E/A-Geräte kann ohne weiteren Aufwand zugegriffen werden.
- **Wirtschaftlichkeit**
  Die Erzeugung eines neuen Threads und der Wechsel zwischen zwei Threads eines Prozesses verursacht erheblich weniger Aufwand (im Vergleich zur Prozesserzeugung/zum Prozesswechsel).
- **Nutzung von Multiprozessorarchitekturen**
  Auch ein einziger multithreading Prozess kann gleichzeitig mehrere Prozessoren nutzen.

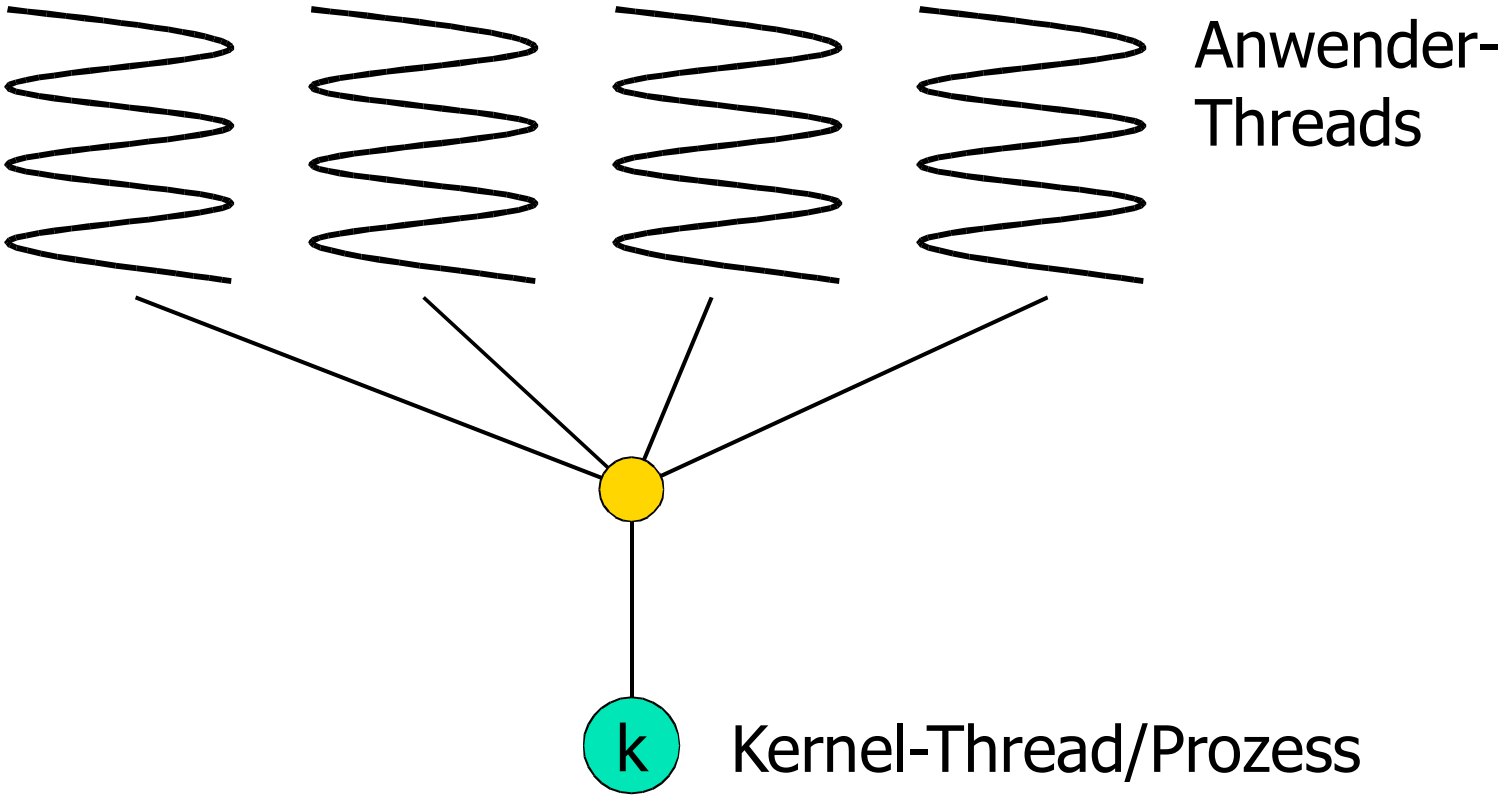
Anwender- und Kernel-Threads

- **Anwender-Threads**
  Erzeugung, Scheduling und Verwaltung der Threads erfolgt über spezielle Programm-Bibliotheken auf Ebene des Anwendungsprogramms. Für den Kernel besteht das Programm aus einem einzigen, single-threaded Prozess.
  Vorteil: effizient (Kernel muss nicht eingreifen)
  Nachteil: Muss ein Thread warten, müssen es alle.
- **Kernel-Threads**
  Erzeugung, Scheduling und Verwaltung der Threads werden durch das Betriebssystem unterstützt.
  Vorteile: Verteilung auf mehrere Prozessoren möglich; ein wartender Thread behindert die anderen Threads nicht.
  Nachteil: Etwas langsamer als Anwender-Threads.

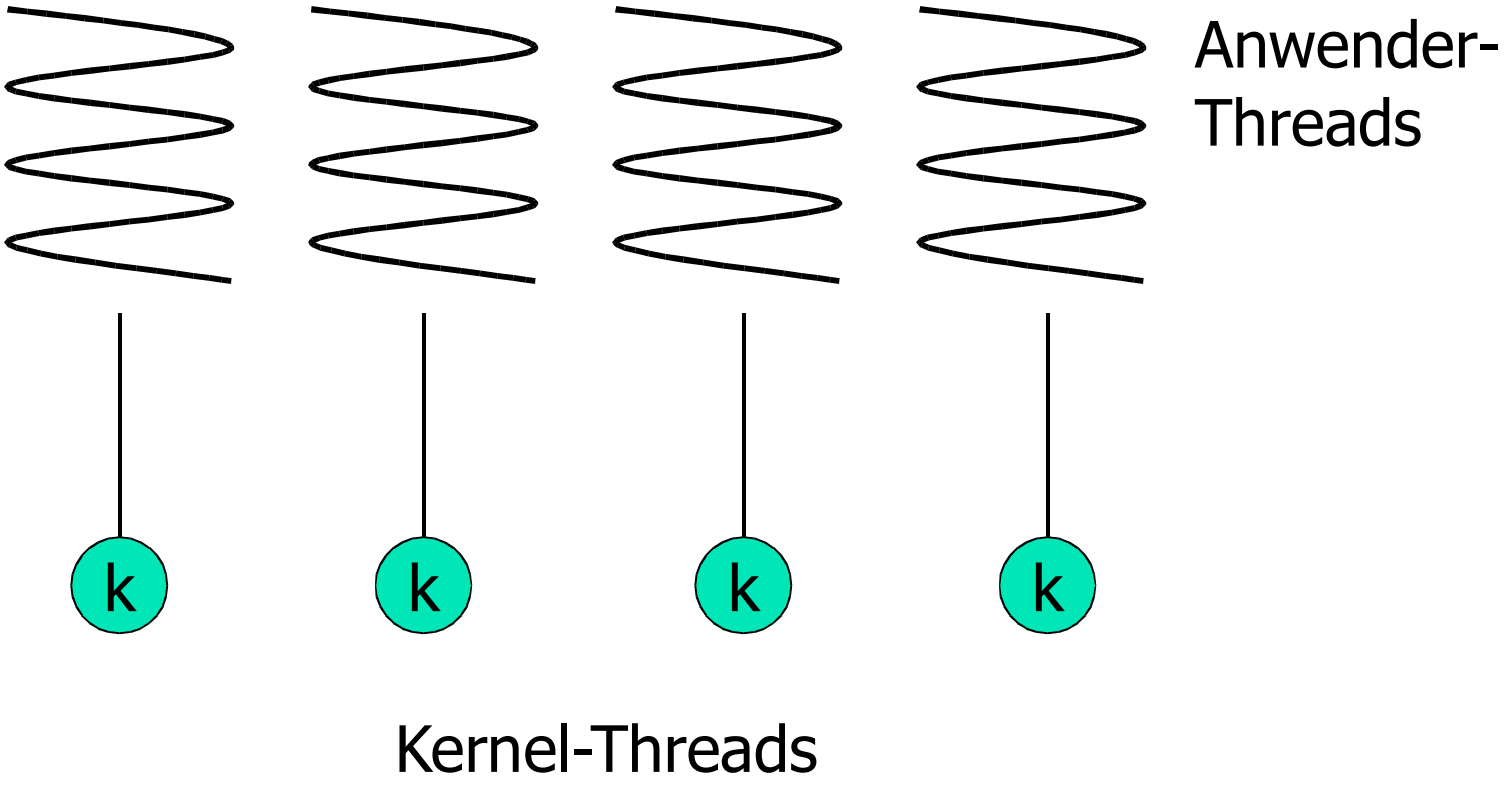
Multithreading-Modelle

- Many-to-One-Modell
  - Mehrere Anwender-Threads werden auf einen Kernel-Thread abgebildet.
  - Beispiele: Green-Thread-Library bei Solaris 2, POSIX Pthread-Library, Betriebssysteme ohne Thread-Unterstützung
- One-to-One-Modell
  - Jeder Thread eines Anwendungsprogramms wird auf genau einen Kernel-Thread abgebildet
  - Beispiele: Windows NT, Windows 2000, OS/2
- Many-to-Many-Modell
  - Die Threads der Anwendungsprogramme werden auf eine Anzahl von Kernel-Threads gemultiplext.
  - Beispiele: IRIX, HP-UX, Tru64 UNIX

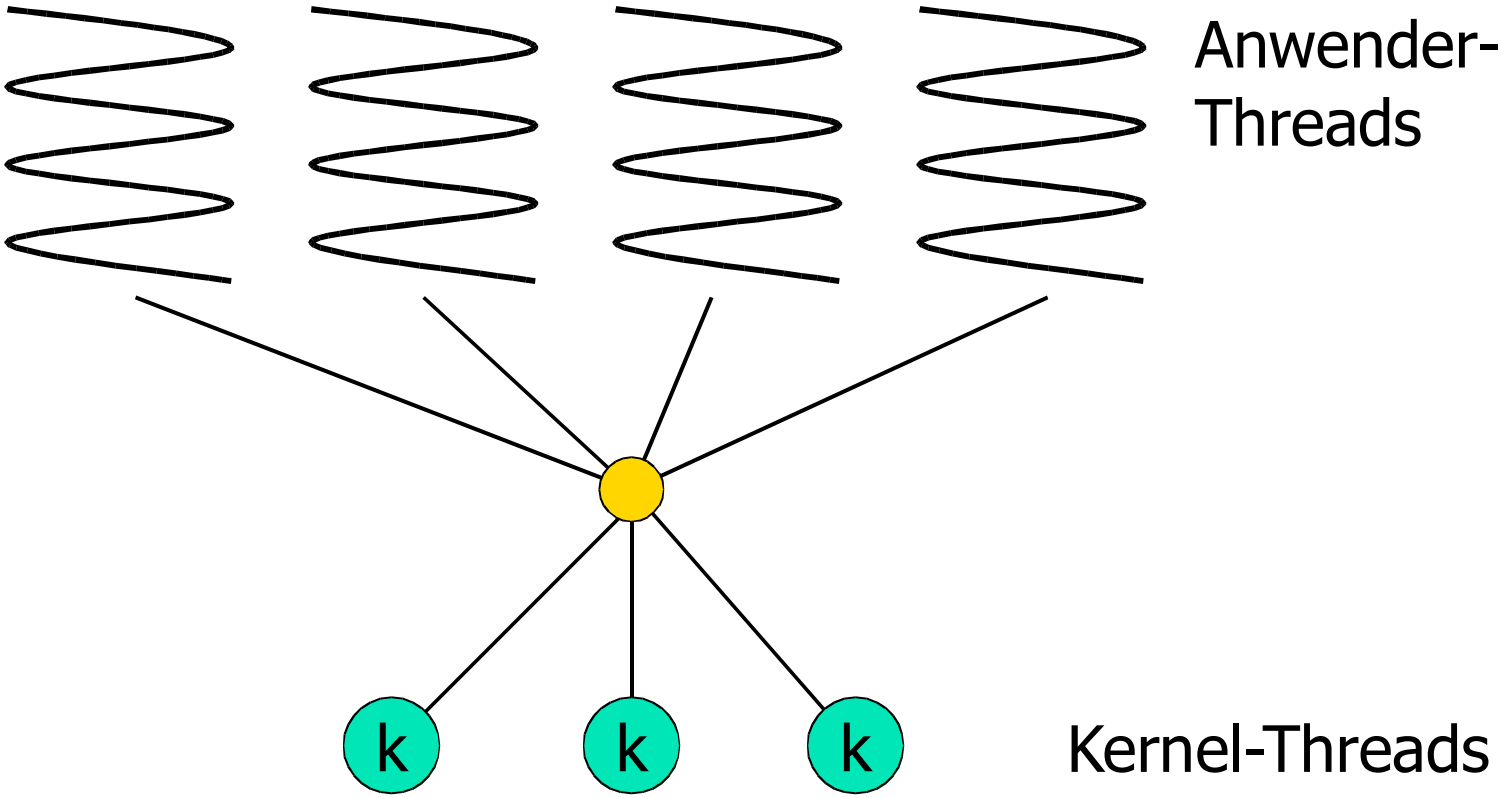
Multithreading-Modelle: Many-to-One

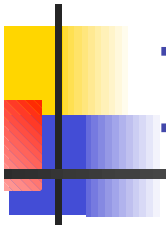


Multithreading-Modelle: One-to-One



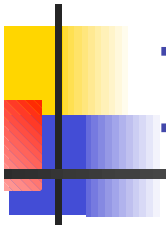
Multithreading-Modelle: Many-to-Many





Implementierungsaspekte (I)

- fork:
  - Alle Threads des Prozesses werden dupliziert
  - Nur der Thread, der fork aufgerufen hat, wird dupliziert
- Abbruch eines Threads:
  - Asynchroner Abbruch: Sofortige Beendigung des Threads
  - Verzögerter Abbruch: Der Thread überprüft regelmäßig, ob er sich beenden soll. Ist dies der Fall, führt er noch alle notwendigen Abschluss-arbeiten aus und beendet sich dann.



Implementierungsaspekte (II)

- Signalbehandlung
  - Das Signale wird dem zugehörigen Thread übermittelt (z.B. „division by zero“)
  - Das Signal wird an alle Threads übermittelt (z.B. „<control><c>“)
  - Das Signal wird an einen beliebigen Thread übermittelt (z.B. „window resize“)
  - Das Signal wird an einen bestimmten Thread übermittelt (z.B. „window resize“)
- Thread-Pools
  - Erzeugen einer begrenzten Zahl von Threads
  - Wiederverwerten nicht mehr benötigter Threads (Recycling ist schneller als neu erzeugen.)

Solaris-2-Threads