Tasks und Threads

Einführung

UNIX-Prozeßkonzept ist für viele heutige Anwendungen unzureichend

- in Multiprozessorsystemen werden häufig parallele Abläufe in einem virtuellen Adreßraum benötigt
- zur besseren Strukturierung von Problemlösungen sind oft mehrere Aktivitätsträger innerhalb eines Adreßraums nützlich
- typische UNIX-Server-Implementierungen benutzen die *fork*-Operation, um einen Server für jeden Client zu erzeugen

  ⇒ Verbrauch unnötig vieler System-Ressourcen (Datei-Deskriptoren, SKTs, Speicher, ...)

Lösungsansätze:

- derzeitige Multiprozessor-Betriebssysteme (z. B. Dynix) ermöglichen gemeinsame Speicherbereiche für mehrere Prozesse

  Nachteile:

  - viele Betriebsmittel zur Verwaltung eines Prozesses notwendig; Prozeßumschaltungen aufwendig → teuer
  - innerhalb einer solchen Prozeßfamilie wäre häufig ein anwendungsorientiertes Scheduling notwendig: schwierig realisierbar

- einige Anwendungen lassen sich mit Hilfe von Koroutinen (auf Benutzerebene) innerhalb eines Prozesses gut realisieren

  Nachteile:

  - Scheduling zwischen den Koroutinen schwierig (Verdrängung meist nicht möglich)
  - in Multiprozessorsystemen keine parallelen Abläufe möglich
  - wird eine Koroutine wegen eines *page faults* oder in einem Systemaufruf blockiert, ist der gesamte Prozeß blockiert

- Threads: leichtgewichtige Prozesse *(lightweighted processes)*
  - •• eine Gruppe leichtgewichtiger Prozesse nutzt gemeinsam eine Menge von Betriebsmitteln
  - •• jeder leichtgewichtige Prozeß ist aber ein eigener Aktivitätsträger
    - – eigener Programmzähler
    - – eigener Registersatz
    - – eigener Stack
  - •• Umschalten zwischen zwei leichtgewichtigen Prozessen einer Gruppe ist erheblich billiger als eine normale Prozeßumschaltung
    - – es müssen nur die Register und der Programmzähler gewechselt werden (entspricht dem Aufwand für einen Funktionsaufruf)
    - – Adreßraum muß nicht gewechselt werden
    - – alle Systemressourcen bleiben verfügbar

Abstraktionen in MACH: Tasks und Threads

Task: Betriebsumgebung (System-Resourcen) für Aktivitätsträger

- virtueller Adreßraum
- Zugriffsrechte
- Betriebsmittel-Informationen
- kein Programm-Ablauf und keine Register

Thread: Aktivitätsträger und seine Ablaufumgebung

- Registersatz
- Stack
- Programmzähler

- Innerhalb einer *Task* können beliebig viele *Threads* existieren
- In einer echten Multiprozessorumgebung können verschiedene *Threads* einer *Task* parallel auf mehreren Prozessoren ablaufen
- Ein herkömmlicher UNIX-Prozeß entspricht einer MACH-*Task* mit einem *Thread*