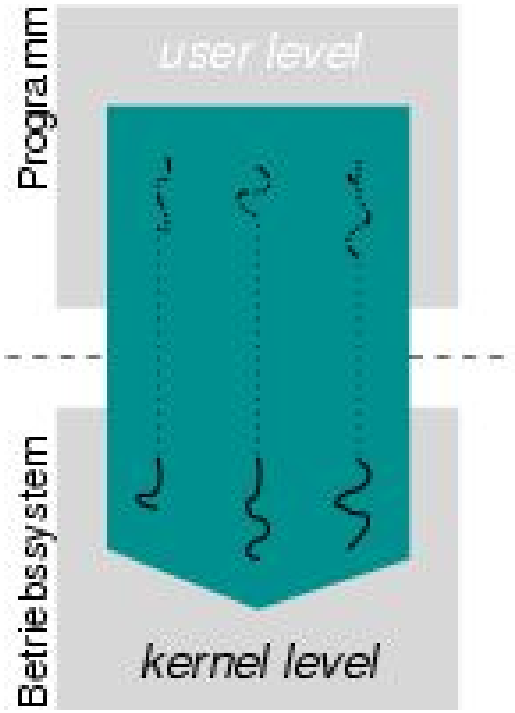
Programmfaden

thread ***of control*** 1. der Faden; der Zusammenhang. 2. durchziehen.

- Einplanungseinheit (*”unit of scheduling“*) für die Vergabe der CPU
  - die Ablaufplanung erfolgt
    - betriebsmittelorientiert
    - ereignisgesteuert
- auch bekannt als leichtgewichtiger Prozess (”light-weight process“, LWP)
  - der sich mit anderen F¨aden eine gemeinsame Ausführungsumgebung teilt

- Entsprechend der Ebene ihres Vorkommens bzw. ihrer Verwaltung werden Fäden in zwei Hauptgruppen unterteilt:
  - Fäden auf Benutzerbene (”Benutzerfäden“)
  - Fäden auf Kernebene (”Kernfäden“).

Programmfaden - Kernebene

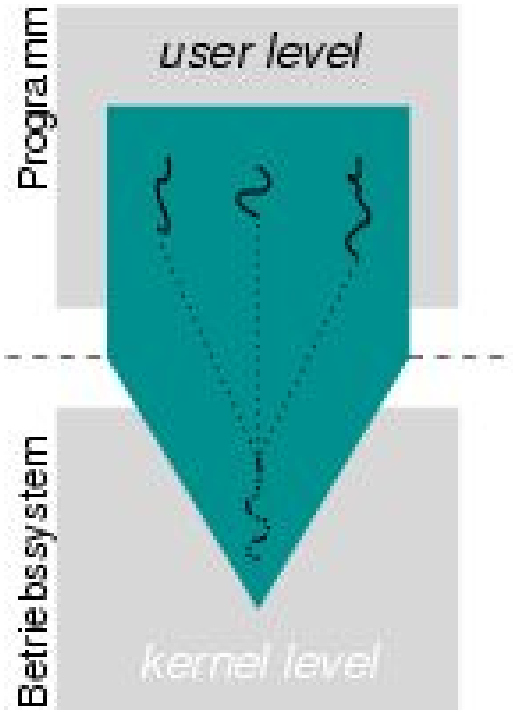


kernel-level thread Prozesskonzept der klassischen Form zur Implementierung nicht-sequentieller Programme

- Prozesse sind Fäden des Programms ”Betriebssystem“
- Benutzerfäden werden Kernf¨aden exklusiv zugeordnet
  - jeder Benutzerfaden besitzt (s)einen Kernfaden
- nicht jeder Kernfaden ”trägt“ einen Benutzerfaden
- Ablaufplanung ist Funktion des Betriebssystem (kern)s

Operationen auf Benutzerfäden sind im Betriebssystemkern implementiert, was zur Konsequenz hat, dass Fadenwechsel innerhalb desselben Benutzerprogramms auch über den Betriebssystemkern verlaufen
$\Rightarrow$ Laufzeitaufwand

Programmfaden - Anwendungsebene



user-level thread Prozesskonzept, das Implementierung und Verwaltung von Fäden ganz in ”Benutzerhände“ legt

- Prozesse sind mehrfädige (Anwendungs-) Programme
- zur Ausführung werden virtuelle Prozessoren verwendet
  - die Prozessoren können als Kernfäden realisiert sein
  - der Kern reicht ”scheduler activations“ nach oben
- Ablaufplanung ist Funktion jedes Anwendungsprogramms

Existenz und Anzahl von Benutzerfäden sind dem Betriebssystemkern unbekannt, weshalb über die Auslastung der virtuellen Prozessoren exakte Aussagen nicht getroffen werden können
$\Rightarrow$ Prozessorauswahl

Fadenverläufe – Im Stoßbetrieb

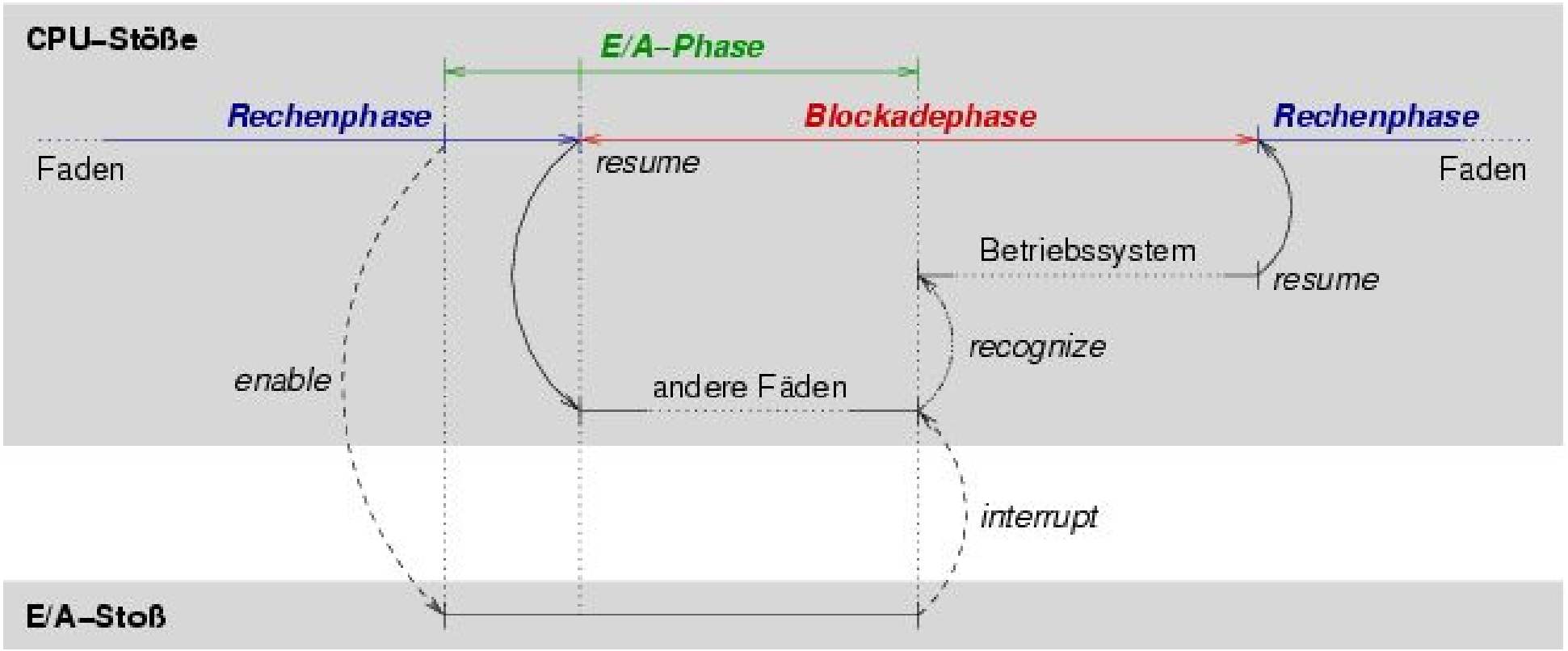
Rechenphase ”CPU-Stoß“ (*CPU burst*)

- aktive Phase eines Programmfadens: die CPU führt Instruktionen aus

Blockadephase ”E/A-Stoß“ (*I/O burst*)

- inaktive Phase eines Programmfadens, z. B. als Folge von Ein-/Ausgabe:
  - eine E/A-Operation wurde auf Anweisung des Faden in Gang gesetzt
  - der Faden muss die Beendigung der E/A-Operation abwarten
- allgemein: ein Programmfaden erwartet (passiv) ein Ereignis
  - das Ereignis wird von einem anderen Faden signalisiert
  - ein E/A-Gerät kann dabei als ”externer Faden“ betrachtet werden

CPU Stoß vs. I/O Stoß



Fäden als Leistungsoptimierung

- der CPU-Stoß von Faden$_x$ verläuft parallel* zum E/A-Stoß von Faden$_y$
  - ggf. werden CPU- bzw. E/A-Stöße weiterer Fäden zum ”Auffüllen“ genutzt
- die Auslastung wird verbessert durch die Überlappung der verschiedenen Stöße
  - in einem Monoprozessorsystem kann immer nur ein CPU-Stoß aktiv sein
  - parallel dazu können jedoch viele E/A-Stöße (anderer Fäden) laufen
  - als Folge sind CPU und E/A-Geräte andauernd mit Arbeit beschäftigt
- bei nur einen Prozessor (CPU, E/A-Gerät) sind die Fäden zu serialisieren

*Ein E/A-Stoß ist zu einem Zeitpunkt zwar genau einem Programmfaden zugeordnet, er wird jedoch von einem separaten *”I/O processor“* (IOP) ausgeführt — dem E/A-Gerät. Dadurch ergibt sich echte Parallelität auf Stoßebene.

- die *absolute* Ausführungsdauer später ”eintreffender“ Fäden verlängert sich:
  - Ausgangspunkt seien *n* Fäden mit gleichlanger Bearbeitungsdauer *k*
  - der erste Faden wird um die Zeitdauer *0* verzögert
  - der zweite Faden um die Zeitdauer *k*, der *i*-te Faden um *(i - 1) · k*
  - der letzte von n Fäden wird verzögert um *(n - 1) · k*

$$\frac{1}{n}\cdot\sum_{i=1}^{n}(n-1)\cdot k=\frac{(n-1)}{2}\cdot k$$

- die mittlere Verzögerung wächst (subjektiv) proportional mit der Fadenanzahl

Subjektive Empfindung der Verzögerung

- die mittlere Verzögerung eines Fadens ergibt sich zu: $2 \cdot t_{CPU}$
  - mit t_{CPU} gleich der mittleren Dauer eines CPU-Stoßes
  - bei genügend vielen asynchron ablaufenden E/A-Operationen
- zwischen CPU- und E/A-Stößen besteht eine große Zeitdiskrepanz
- die Verzögerung durch E/A-Operationen ist dominant
  - der proportionale Verzögerungsfaktor bleibt weitestgehend verborgen
  - er greift erst ab einer bestimmten Anzahl von Programmfäden
  - viele Anwendungen/Benutzer nehmen die Verzögerung daher nicht wahr
- die ”Totzeit“ bei E/A-Stößen wird für CPU-Stöße laufbereiter Fäden genutzt

Monopolisierung der CPU durch Programmfäden

- mit erfolgter Prozessorzuteilung gewinnen Fäden die Kontrolle über die CPU
  - die CPU führt nur noch Anweisungen aus, die das Benutzerprogramm vorgibt
- das Betriebssystem kann die Kontrolle nur bedingt zurückgewinnen:
  - die Fäden müssten {synchrone,asynchrone} Programmunterbrechungen erfahren
- synchrone Programmunterbrechungen sind ein eher schwaches Instrument
  - die Fäden müssten sich kooperativ dem Betriebssystem gegenüber erweisen
  - ”böswillige“ Programme können schnell die Kooperative gefährden/auflösen

sporadische Unterbrechung bei Beendigung eines E/A-Stoßes -

- der E/A-Stoß musste vorher von einem Faden erst ermöglicht werden
- wann und ob überhaupt ein E/A-Stoß ausgelöst wird ist ungewiss
- ebenso ungewiss ist die E/A-Stoßdauer und damit der Interrupt-Zeitpunkt

periodische Unterbrechung durch Einsatz eines Zeitgebers

- der Zeitgeber wird je nach Bedarf vom Betriebssystem programmiert
- er sorgt in der Regel für zyklische Unterbrechungen (*timer interrupts*)
- mit Ablauf der vorgegebenen Zeit wird das Betriebssystem reaktiviert

$\Rightarrow$ Zugriffe auf Zeitgeber und Interrupt-Maske sind *privilegierte Operationen* !

Einplanung von Programmfäden

Wiederholung

- Scheduling stellt sich allgemein zwei grundsätzlichen Fragestellungen:

  1. Zu welchem Zeitpunkt sollen Prozesse ins System eingespeist werden?

  2. In welcher Reihenfolge sollen Prozesse ablaufen?

- Ein Scheduling-Algorithmus verfolgt das Ziel, den von einem Rechnersystem zu leistenden Arbeitsplan so aufzustellen (und zu aktualisieren), dass ein gewisses Maß an Benutzerzufriedenheit maximiert wird.

$\Rightarrow$ Prozess == Faden

Ebenen der Prozesseinplanung

long-term scheduling kontrolliert den Grad an Mehrprogrammbetrieb [s – min]

- Benutzer Systemzugang gewähren, Programme zur Ausführung zulassen
- Prozesse dem *medium-* bzw. *short-term scheduling* zuführen

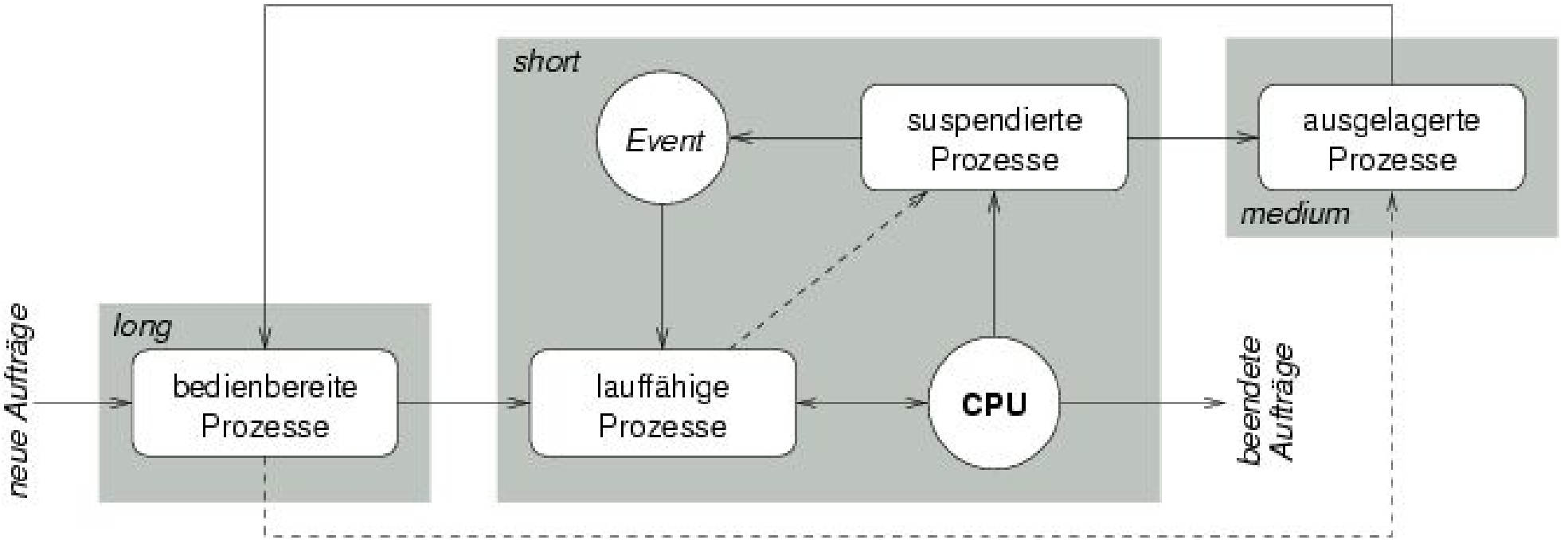
medium-term scheduling als Teil der Ein-/Auslagerungsfunktion [ms – s]

- Programme zwischen Vorder- und Hintergrundspeicher hin- und herbewegen
- *swapping*: auslagern (*swap-out*), einlagern (*swap-in*)

short-term scheduling regelt die Prozessorzuteilung an die Prozesse [µs – ms]

- ereignisgesteuerte Ablaufplanung: Unterbrechungen, Systemaufrufe, Signale
- Blockierung bzw. Verdrängung des laufenden Prozesses

Ebenen der Prozesseinplanung (2)



Verfahrensweisen (1) kooperativ vs. präemptiv

cooperative scheduling voneinander abhängiger Prozesse

- Prozesse müssen die CPU freiwillig abgeben, zugunsten anderer Prozesse
- die Programmausführung muss (direkt/indirekt) Systemaufufe bewirken

die Systemaufufe müssen (direkt/indirekt) den Scheduler aktivieren

preemptive scheduling voneinander unabhängiger Prozesse

- Prozessen wird die CPU entzogen, zugunsten anderer Prozesse
- der laufende Prozess wird ereignisbedingt von der CPU *verdrängt**
- die Ereignisbehandlung aktiviert (direkt/indirekt) den Scheduler

*Beispielsweise als Folge einer Programmunterbrechung, ggf. nur zur Durchsetzung von CPU-Schutz.

Verfahrensweisen (2) deterministisch vs. probabilistisch

deterministic scheduling bekannter, exakt vorberechneter Prozesse

- Prozesslaufzeiten/-termine sind bekannt, sie wurden ggf. ”offline“ berechnet
- die genaue Vorhersage der CPU-Auslastung ist möglich
- das System garantiert die Einhaltung der Prozesslaufzeiten/-termine
- die Zeitgarantien gelten unabhängig von der jeweiligen Systemlast !

probabilistic scheduling unbekannter Prozesse

- Prozesslaufzeiten/-termine bleiben unbestimmt
- die CPU-Auslastung kann lediglich abgeschätzt werden
- das System kann Zeitgarantien weder geben noch einhalten
- Zeitgarantien sind durch Anwendungsmaßnahmen bedingt erreichbar

Verfahrensweisen (3) statisch vs. dynamisch

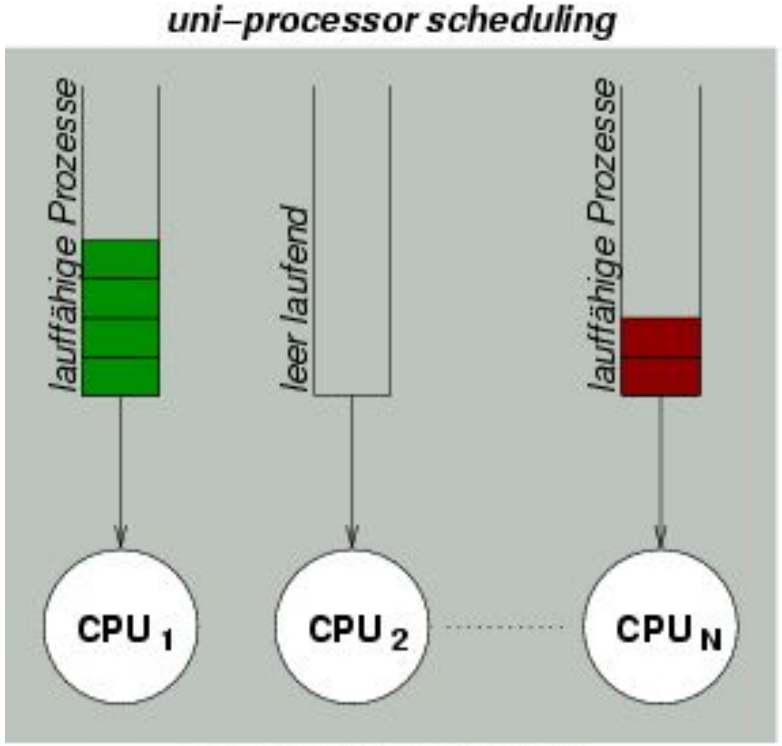
offline scheduling statisch, vor der eigentlichen Programmausführung

- wenn die Komplexität eine Ablaufplanung im laufenden Betrieb verbietet
  - Einhaltung aller Zeitvorgaben garantieren: ein NP-vollständiges Problem
  - kritisch, wenn auf jede abfangbare katastrophale Situation zu reagieren ist
- Ergebnis der Vorberechung ist ein vollständiger Ablaufplan (in Tabellenform)
  - (semi-) automatisch erstellt per Quelltextanalyse spezieller ”Übersetzer“
  - oft zeitgesteuert abgearbeitet/ausgeführt als Teil der Prozessabfertigung
- die Verfahren sind zumeist beschränkt auf strikte Echtzeitsysteme

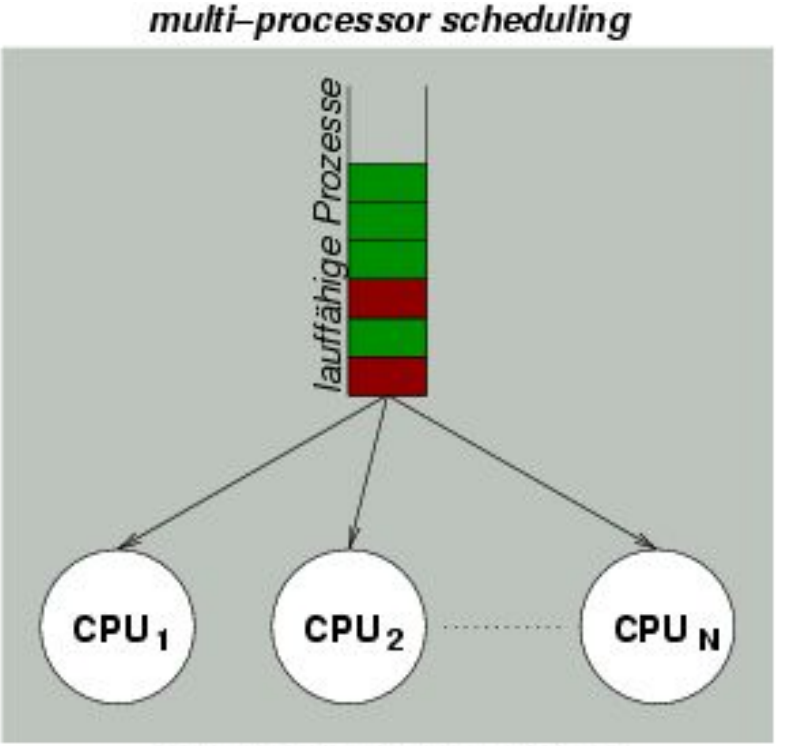
online scheduling dynamisch, während der eigentlichen Programmausführung

- interaktive- und Stapelsysteme, aber auch schwache Echtzeitsysteme

Exkurs - Mehrprozesssysteme



separate Warteschlangen



gemeinsame Warteschlange

Dimensionen der Prozesseinplanung

- die Kriterien, nach denen Einplanung betrieben wird, sind unterschiedlich:

  benutzerorientierte Kriterien betrachten die Benutzerdienlichkeit

  - d. h. das vom jeweiligen Benutzer wahrgenommene Systemverhalten
  - bestimmen im großen Maße die Akzeptanz des Systems beim Benutzer

  systemorientierte Kriterien betrachten die Systemperformanz

  - d. h. die effektive und effiziente Auslastung der Betriebsmittel
  - sind von Bedeutung bei kommerziellen Dienstleistungsanbietern

- die Benutzerdienlicheit ist auch stark durch die Systemperformanz bedingt

Benutzerorientierte Kriterien

Antwortzeit Minimierung der Zeitdauer von der Auslösung eines Systemaufrufs bis zur Entgegennahme der Rückantwort, bei gleichzeitiger Maximierung der Anzahl interaktiver Prozesse**.**

Durchlaufzeit Minimierung der Zeitdauer vom Starten eines Prozesses bis zu seiner Beendigung, d. h., der effektiven Prozesslaufzeit und aller anfallenden Prozesswartezeiten.

Termineinhaltung Starten und/oder Beendigung eines Prozesses (bis) zu einem fest vorgegebenen Zeitpunkt.

Vorhersagbarkeit Deterministische Ausführung des Prozesses unabhängig von der jeweils vorliegenden Systemlast.

Systemorientierte Kriterien

Antwortzeit Minimierung der Zeitdauer von der Auslösung eines Systemaufrufs bis zur Entgegennahme der Rückantwort, bei gleichzeitiger Maximierung der Anzahl interaktiver Prozesse.

Durchlaufzeit Minimierung der Zeitdauer vom Starten eines Prozesses bis zu seiner Beendigung, d. h., der effektiven Prozesslaufzeit und aller anfallenden Prozesswartezeiten.

Termineinhaltung Starten und/oder Beendigung eines Prozesses (bis) zu einem fest vorgegebenen Zeitpunkt.

Vorhersagbarkeit Deterministische Ausführung des Prozesses unabhängig von der jeweils vorliegenden Systemlast.

Betriebsart vs. Einplanungskriterien

allgemein: Durchsetzung (der Strategie); Gerechtigkeit, Lastausgleich

Stapelbetrieb: Durchsatz, Durchlaufzeit, Prozessorauslastung

interaktiver Betrieb: Antwortzeit; Proportionalität:

- Benutzer haben meist eine inhärente Vorstellung über die Dauer bestimmter Aktionen. Dieser (oft auch falschen) Vorstellung sollte das System aus Gründen der Benutzerakzeptanz möglichst entsprechen.

Echtzeitbetrieb: Dringlichkeit, Termineinhaltung, Vorhersagbarkeit

- steht meist im Widerspruch zu Gerechtigkeit und Lastausgleich

Prozessabfertigung - *Dispatching*

- der aktuell laufende Prozess muss dem System ”jederzeit“ bekannt sein
  - zur prozessbezogenen Abrechnung der Inanspruchnahme von Betriebsmitteln
    - Benutzer-, System- und Stoßzeiten
    - Energiebedarf, Speicherbelegung, Adressraumgröße
    - geöffnete Dateien, genutzte Geräte, . . .
  - zur prozessbezogenen Überprüfung der Zugriffsrechte auf Betriebsmittel
- Buch über diesen Prozess führt der ”Prozessabfertiger“ (process dispatcher)
  - ausgedrückt als ”Zeiger“ auf den PD des aktuell laufenden Prozesses
  - ein Fadenwechsel hat die Aktualisierung dieses Zeigers zur Konsequenz
- den Zeigerwert liefert eine je nach Systemkonzept variierende Funktion

Einplanung vs. Abfertigung

process scheduler trifft strategische Entscheidungen

- betrachtet wird immer eine Menge lauffähiger Prozesse
  - die PDs der betreffenden Prozess sind in einer Warteschlange aufgereiht
  - welche Position darin ein PD einnimmt, obliegt der Einplanungsstrategie
- der aktuell laufende Prozess ist immer von der Entscheidung mit betroffen
  - dazu muss der PD des laufenden Prozesses ”jederzeit greifbar“ sein
  - bei der Prozesabfertigung wird entsprechend Buch über diesen PD geführt

process dispatcher setzt die Entscheidungen durch

- schaltet um zum jeweils ausgewählten Prozess und vermerkt seinen PD

Race Condition „Verdrängung“

- verdrängende Ablaufplanung ist ”Querschnittsbelang“ eines ganzen Systems:

  Abfertigung muss atomar erfolgen; nur ein Ansatz, bei dem ein Umsetzen des

  - Stapelzeigers gleichzeitig ein Umsetzen des PD-Zeigers bedeutet, verläuft *implizit koordiniert*. PD ist ”aktueller Parameter“

  Einplanung muss atomar erfolgen; Operationen auf die zur Implementierung der Warteschlange(n) verwendeten dynamischen Datenstrukturen sind zu koordinieren.

  Anwendung muss ggf. atomar erfolgen; mehrfädige Programme bzw. einfädige Programme, die sich mit anderen Programmen gemeinsame Variablen teilen, sind zu koordinieren.

- unterlassene, schlechte oder falsche Koordinierung verursacht ”Albträume“

Zusammenfassung

- eine Prozessinkarnation ist ein Programmfaden des Betriebssystems
  - Programmfäden (threads) sind eine spezialisierte Form von Koroutinen
  - Koroutine: autonomer Kontrollfluss mit kooperativem Operationsprinzip
- zwei Fadenarten kommen vor: *kernel-level threads* vs. *user-level threads*
  - beide Ausprägungsformen von Fäden können gleichzeitig vorhanden sein
  - Kernfaden + ggf. ein abstrakter Prozessor für Benutzerfäden
- Fadenverläufe schreiten stoßweise voran: CPU-Stoß vs. E/A-Stoß
  - E/A-Stöße wartender Fäden kommen CPU-Stößen lauffähiger Fäden zugute
  - die Einplanungs- und Abfertigungsverfahren sind ggf. zu koordinieren