

Systemprogrammierung

Grundlage von Betriebssystemen

Teil C – IX.1 Prozessverwaltung: Einplanungsgrundlagen

Wolfgang Schröder-Preikschat

22. Oktober 2015



Gliederung

Einführung

Programmfäden

Grundsätzliches

Fadenverläufe

Leistungsoptimierung

Arbeitsweisen

Ebenen

Ebenenübergänge

Verdrängung

Gütemerkmale

Benutzerdienlichkeit

Systemperformanz

Betriebsart

Zusammenfassung



Agenda

Einführung

Programmfäden

Grundsätzliches

Fadenverläufe

Leistungsoptimierung

Arbeitsweisen

Ebenen

Ebenenübergänge

Verdrängung

Gütemerkmale

Benutzerdienlichkeit

Systemperformanz

Betriebsart

Zusammenfassung



Lehrstoff

- **Programmfäden** als ein gängiges Mittel zum Zweck der Einplanung von Prozessen kennenlernen
 - Lauf- und Wartephänen von Fäden im Zusammenhang behandeln
 - phasenverschränkte Fadenausführung zur Leistungssteigerung nutzen
 - Uneindeutigkeit logischer Prozesszustände als Normalität sehen
- grundsätzliche **Arbeitsweisen** der Prozessplaner (*process scheduler*) verstehen und zu differenzieren
 - lang-, mittel- und kurzfristige Prozesseinplanung betrachten
 - Facetten der verdrängenden Prozesseinplanung beleuchten
 - Latenz und Determinismus in Zusammenhang bringen
- typische **Kriterien** zur und charakteristische (Güte-) **Merkmale** der Einplanung von Prozessen ansprechen
 - benutzer- und systemorientierte Kriterien unterscheiden
 - den Zusammenhang mit bestimmten Rechnerbetriebsarten erkennen



Gliederung

Einführung

Programmfäden

Grundsätzliches

Fadenverläufe

Leistungsoptimierung

Arbeitsweisen

Ebenen

Ebenenübergänge

Verdrängung

Gütemerkmale

Benutzerdienlichkeit

Systemperformance

Betriebsart

Zusammenfassung



© wosch

SP (SS 2015, C – IX.1)

2. Programmfäden

IX.1/5

Stoßbetrieb

burst mode

- einerseits die als **CPU-Stoß** (*CPU burst*) bezeichnete Laufphase
 - aktive Phase eines Fadens (auch: Rechenphase)
 - alle zur Ausführung erforderlichen Betriebsmittel sind verfügbar
 - der Faden ist eingelastet, ihm wurde der Prozessor zugeteilt
- andererseits der ggf. als Wartephase wirkende **E/A-Stoß** (*I/O burst*)
 - ggf. die inaktive Phase eines Fadens (auch: E/A-Phase)
 - nicht alle zur Ausführung erforderlichen Betriebsmittel sind verfügbar
 - Ein-/Ausgabe abwarten, **Betriebsmittel** [1, S. 10–11] zu erwarten
 - **konsumierbare Betriebsmittel**: Eingabedaten, Ausgabequittung (Signal)
 - **wiederverwendbare Betriebsmittel**: Puffer, Geräte, ..., Prozessor
 - die Betriebsmittel werden letztlich durch andere Fäden bereitgestellt
 - ein E/A-Gerät kann dabei als „externer Faden“ betrachtet werden²
- kontinuierlich einander abwechselnde Phasen eines jeden Prozesses
 - je nach Programm oder Progammmix, mit unterschiedlichen Stoßlängen

²Ein „externer Prozess“, der (a) durch Anweisung eines „internen Prozesses“ entsteht oder (b) entsprechend physikalischer Vorgaben autonom voranschreitet.



© wosch

SP (SS 2015, C – IX.1)

2.2 Programmfäden – Fadenverläufe

IX.1/7

Prozessorzuteilungseinheit

- Einplanungseinheit für die Prozessorvergabe ist der **Faden**¹ (*thread*)
 - geplant wird, wann ein Faden wie lange ablaufen darf
- die Ablaufplanung der Fäden geschieht **betriebsmittelorientiert** und ist **ereignis- oder zeitgesteuert** ausgelegt
 - die Laufbereitschaft eines Fadens hängt von der Verfügbarkeit all jener Betriebsmittel ab, die für seinen Ablauf erforderlich sind
 - die Bereitstellung von Betriebsmitteln (ggf. durch andere Fäden) kann die sofortige Einplanung von Fäden bewirken oder
 - die Einplanung erfolgt in fest vorgegebenen Zeitintervallen
- dabei handelt es sich um Vorgänge im System, die voneinander ent- oder miteinander gekoppelt sein können
 - d.h., zeitversetzt oder zeitgleich zum Ablauf eingeplanter Fäden
- **Einplanung** eines Fadens ist nicht gleichzusetzen mit **Einlastung**:
 - Einplanung ist der Vorgang der Reihenfolgenbildung von Aufträgen
 - Einlastung ist der Moment der Zuteilung von Betriebsmitteln

¹Verbreitete Variante einer Prozessinkarnation [1, S. 6].



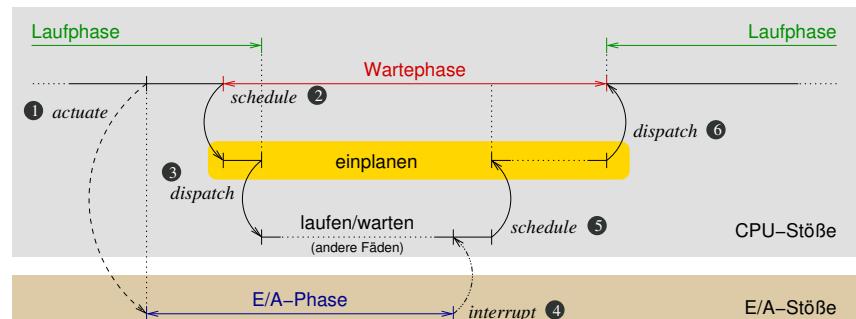
© wosch

SP (SS 2015, C – IX.1)

2.1 Programmfäden – Grundsätzliches

IX.1/6

Lauf-, E/A- und Wartephassen von Fäden (1)



- in Praxis ist der **Prozesszustand** [1, S. 24] „phasenuneindeutig“
 - **laufend** ■ Laufphase eines Fadens, direkt vor (logisch) oder nach (real) dem Prozesswechsel bei der Einlastung (3 bzw. 6)
 - **blockiert** ■ noch zur Laufphase des Fadens, vor der Prozessorabgabe (2–3)
 - physische Wartephase des Fadens bis zur Bereitstellung (3–5)
 - **bereit** ■ noch zur Laufphase, vor der Prozessorabgabe (2–3) und bei entsprechend kurzer E/A-Phase (1–5)
 - physische Wartephase des Fadens bis zur Einlastung (3–6)



© wosch

SP (SS 2015, C – IX.1)

2.2 Programmfäden – Fadenverläufe

IX.1/8

Lauf-, E/A- und Wartephassen von Fäden (2)

- **Betriebssystemkontrollfluss** zur Fadeneinplanung und -einlastung:
 1. der laufende Faden stößt einen E/A-Vorgang an (*actuate*)
 2. er wartet passiv auf die Beendigung der Ein-/Ausgabe (*schedule*)
 - Anforderung eines wiederverwend-/konsumierbaren Betriebsmittels
 3. und lastet einen eingeplanten, laufbereiten Faden ein (*dispatch*)
 4. die Beendigung der Ein-/Ausgabe wird signalisiert (*interrupt*)
 - Bereitsstellung des konsumierbaren Betriebsmittels „Signal“
 5. der auf dieses Ereignis wartende Faden wird eingeplant (*schedule*)
 6. der Faden wird eingelastet, sobald er an der Reihe ist (*dispatch*)
- Kreislauf bis zum **Leerlauf** (*idle state*) mangels laufbereiter Fäden
 - normalerweise wird der Prozessor dann in den Energiesparmodus versetzt
 - eher untypisch ist **aktives Warten** auf die Bereitstellung eines Fadens
 - durch (a) den einzigen „blockiert laufenden“ Prozess \ominus oder (b) einen speziell genau nur dafür vorgehaltenen Prozess (*idle process*) \ominus
 - im Fall von (a) kann der so wartende Prozess sich selbst bereitstellen — oder einlasten, wenn dies die Einplanungsstrategie (*full preemption*) zulässt
 - unabhängig davon, muss ein „blockierend laufender“ Prozess sich selbst „laufend bereit“ stellen können



Zwangsserialisierung von Programmfade

In Bezug auf ein Exemplar des wiederverwendbaren Betriebsmittels „CPU“ bzw. „Core“:

- die **absolute Ausführungsduauer** nach Ankunftszeit eingeordneter laufbereiter Fäden verlängert sich:
 - Ausgangspunkt seien n Fäden mit gleichlanger Bearbeitungsduauer k
 - der erste Faden wird um die Zeitdauer 0 verzögert
 - der zweite Faden um die Zeitdauer k , der i -te Faden um $(i - 1) \cdot k$
 - der letzte von n Fäden wird verzögert um $(n - 1) \cdot k$

Mittlere Fadenverzögerung

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

- die Vergrößerung der **mittleren Verzögerung** ist proportional zur Fadenanzahl



Fäden als Mittel zum Kaschieren von Totzeiten

- die **Überlappung** von Lauf- und Wartephassen verschiedener Fäden lässt eine Erhöhung der Rechnerauslastung erwarten
 - die Wartephase eines Fadens als Laufphase anderer Fäden nutzen
 - die Stöße anderer Fäden zum „Auffüllen“ von Wartephassen nutzen
- nicht nur die **Auslastung** der CPU kann sich steigern, sondern auch die der angeschlossenen Peripherie (E/A-Geräte)
 - eine CPU kann zu einem Zeitpunkt nur einen CPU-Stoß verarbeiten
 - parallel dazu können jedoch mehrere E/A-Stöße laufen
 - ausgelöst während eines CPU-Stoßes: in der Laufphase eines Fadens wurden mehrere E/A-Vorgänge gestartet
 - ausgelöst von mehreren CPU-Stoßen: die Wartephase eines Fadens wurde mit Laufphasen anderer Fäden gefüllt
 - Folge: CPU und E/A-Geräte sind andauernd mit Arbeit beschäftigt
- beachte: Fäden sind **Ausdrucksmittel** von (a) Mehrprogrammbetrieb oder (b) nicht-sequentiellen Programmen
 - bei weniger Prozessoren als Fäden, sind Fäden zu serialisieren



Subjektive Empfindung der Fadenverzögerung

Nur bis zu einer bestimmten Last, die sich unter anderem durch die Anzahl eingeplanter Fäden bestimmt.

- Startzeiten von Fäden verzögern sich im Mittel um: $\frac{n-1}{2} \cdot t_{cpu}$
 - mit t_{cpu} gleich der mittleren Dauer eines CPU-Stoßes
 - sofern $t_{cpu} \geq t_{ea}$, der mittleren Dauer eines E/A-Stoßes
 - die Praxis liefert als Regelfall jedoch ein anderes Bild: $t_{cpu} \ll t_{ea}$
- Wartephassen bei E/A-Operationen dominieren die Fadenverzögerung
 - zwischen CPU- und E/A-Stoßen besteht eine große Zeitdiskrepanz
 - der proportionale Verzögerungsfaktor bleibt weitestgehend verborgen
 - er greift erst ab einer bestimmten Anzahl von Programmfade
 - nämlich wenn zu einem Zeitpunkt gilt: $\sum_{i=1}^m t_{cpu}^i > \sum_{j=1}^n t_{ea}^j$
 - sehr häufig ist die Fadenverzögerung daher nicht wahrnehmbar
- beachte: **Überlast** durch zuviel eingeplante Fäden ist zu **vermeiden**
 - akkumulierte Länge der CPU-Stoße der jew. E/A-Auslastung anpassen

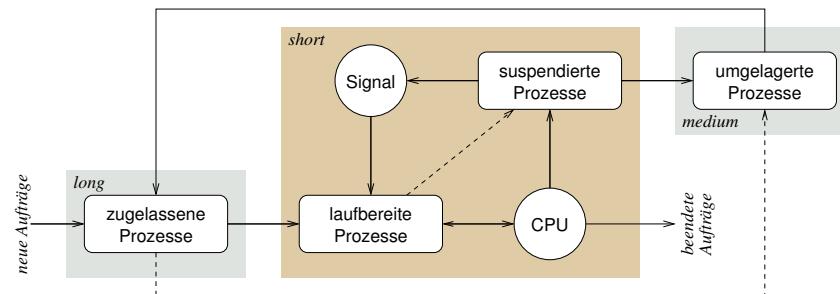


Gliederung

- Einführung
- Programmfaden
 - Grundsätzliches
 - Fadenverläufe
 - Leistungsoptimierung
- Arbeitsweisen
 - Ebenen
 - Ebenenübergänge
 - Verdrängung
- Gütemerkmale
 - Benutzerdienlichkeit
 - Systemperformance
 - Betriebsart
- Zusammenfassung



Phasen der Prozesseinplanung



- Voraussetzung für Mehrprozessbetrieb ist die **kurzfristige Einplanung**
 - laufbereite Prozesse erwarten die Zuteilung des wiederverwendbaren Betriebsmittels „CPU“, d.h. den Start ihrer Laufphase
 - suspendierte Prozesse erwarten die Zuteilung eines konsumierbaren Betriebsmittels „Signal“, d.h. das Ende ihrer Wartephase
- lang- und mittelfristige Einplanung regeln den Mehrprogrammbetrieb
 - Umlagerung macht Hauptspeicher frei für weitere Prozessexemplare
 - im Ergebnis könnten neue Prozesse (z.B. *login*) zugelassen werden



Dauerhaftigkeit von Zuteilungsentscheidungen

- langfristige Einplanung (long-term scheduling)** [s – min]
 - Lastkontrolle**, Grad an Mehrprogrammbetrieb einschränken
 - Programme laden und/oder zur Ausführung zulassen
 - Prozesse der mittel- bzw. kurzfristigen Einplanung zuführen
- mittelfristige Einplanung (medium-term scheduling)** [ms – s]
 - Teil der **Umlagerungsfunktion (swapping)**
 - Programme vom Hinter- in den Vordergrundspeicher bringen
 - Prozesse der langfristigen Einplanung zuführen
- kurzfristige Einplanung (short-term scheduling)** [μ s – ms]
 - Einlastungsreihenfolge** der Prozesse festlegen — obligatorisch

Logische Ebenen der Prozesseinplanung, die, mit Ausnahme der untersten Ebene, nicht in jedem Betriebssystem ein physisches Äquivalent haben.



Prozesszustand vs. Einplanungsebene

- Prozesse haben in Abhängigkeit von der Einplanungsebene (S. 14) zu einem Zeitpunkt einen **logischen Zustand**:
 - kurzfristig (short-term)**
 - bereit, laufend, blockiert
 - mittelfristig (medium-term, mid-term)**
 - schwebend bereit, schwebend blockiert
 - langfristig (long-term)**
 - erzeugt, gestoppt, beendet

*Jedoch legen die **Anwendungsfälle** fest, welche dieser Ebenen von einem Betriebssystem wirklich zur Verfügung zu stellen sind, nicht umgekehrt.*

- ein **Universalbetriebssystem** implementiert eher alle Ebenen, ein **Spezialbetriebssystem** dagegen eher nur die unterste Ebene
 - kurzfristige Prozesseinplanung, obligatorisch bei Mehrprozessbetrieb



Kurzfristige Einplanung

Prozessorzuteilungsreihenfolge

- das Betriebssystem bietet **Mehrprozessbetrieb** (*multi-processing*) auf Basis der Serialisierung von Programmfäden:
 - bereit** (*ready*) zur Ausführung durch den Prozessor (die CPU)
 - der Prozess ist auf der Bereitliste (*ready list*)
 - das Einplanungsverfahren bestimmt die Listenposition
 - laufend** (*running*), erfolgte Zuteilung des Betriebsmittels „CPU“
 - der Prozess vollzieht seinen CPU-Stoß
 - zu einem Zeitpunkt pro CPU nur ein laufender Prozess
 - blockiert** (*blocked*) auf ein bestimmtes Ereignis
 - der Prozess erwartet die Zuteilung eines Betriebsmittels
 - zusätzlich zum Betriebsmittel „CPU“
 - die Zuteilung löst ein anderer (ggf. externer) Prozess aus
- gleichzeitige Vorgänge innerhalb eines Betriebssystems bewirken ggf. **uneindeutige logische Prozesszustände** (S. 8)
 - ein Prozess, der blockiert wird, ist zeitweilig „laufend blockiert“
 - ein solcher Prozess kann dann auch „laufend bereit“ gestellt werden
 - insb. ein Prozess, der den Prozessor im Leerlauf betreibt \mapsto *idle process*



© wosch

SP (SS 2015, C – IX.1)

3.2 Arbeitsweisen – Ebenenübergänge

IX.1/17

Langfristige Einplanung

Zulassungsreihenfolge

- das Betriebssystem verfügt über Funktionen zur **Lastkontrolle** und steuert den Grad an Mehrprogrammbetrieb:
 - erzeugt** (*created*) und fertig zur Programmverarbeitung
 - das **Prozessexemplar** eines Programms wurde geschaffen
 - ggf. steht die Speicherzuteilung jedoch noch aus
 - gestoppt** (*stopped*) und erwartet seine Fortsetzung/Beendigung
 - der Prozess wurde angehalten (z.B. `~Z` bzw. `kill(2)`)
 - Gründe: Überlast, **Verklemmungsvermeidung**, ...
 - beendet** (*ended*) und erwartet seine Entsorgung
 - der Prozess ist terminiert, Betriebsmittelfreigabe erfolgt
 - ggf. muss ein anderer Prozess den „Kehraus“ vollenden
- „gestoppt“ werden können auch bereite, laufende, blockierte Prozesse
 - durch entsprechende Aktionen anderer Prozesse, über Systemaufrufe
 - sofern das Betriebssystem die dazu notwendigen Mechanismen bietet



© wosch

SP (SS 2015, C – IX.1)

3.2 Arbeitsweisen – Ebenenübergänge

IX.1/19

Mittelfristige Einplanung

Umlagerungsreihenfolge

- das Betriebssystem implementiert die **Umlagerung** (*swapping*) von kompletten Programmen bzw. logischen Adressräumen:
 - schwebend bereit** (*ready suspend*)
 - das **Exemplar** eines Prozesses ist ausgelagert
 - verschoben in den Hintergrundspeicher
 - „swap-out“ ist erfolgt
 - „swap-in“ wird erwartet
 - die Einlastung des Prozesses ist außer Kraft
 - genauer: aller Fäden seines Programms
 - schwebend blockiert** (*blocked suspend*)
 - ausgelagerter ereigniserwartender Prozess
 - Ereigniseintritt \mapsto „schwebend bereit“
- Prozesse im Zustand „laufend“ werden normalerweise nicht umgelagert
- im Falle langfristiger Einplanung konkurrieren „schwebend bereite“ Prozesse mit neu zuzulassenden Prozessen
 - gleichwohl genießen in dem Fall erstere (zumeist) Vorrang vor letzteren



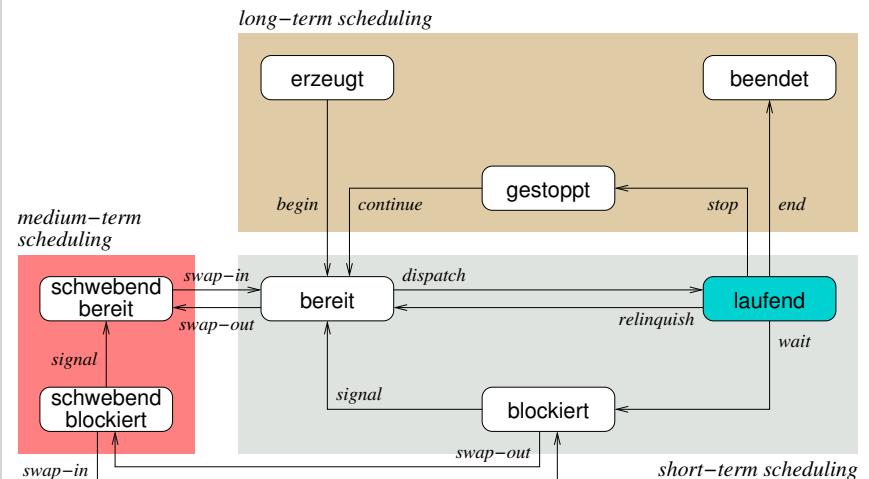
© wosch

SP (SS 2015, C – IX.1)

3.2 Arbeitsweisen – Ebenenübergänge

IX.1/18

Abfertigungszustände im Zusammenhang



- je nach Betriebssystem/-art gibt es weitere „Zwischenzustände“



© wosch

SP (SS 2015, C – IX.1)

3.2 Arbeitsweisen – Ebenenübergänge

IX.1/20

Einplanungs-/Auswahlzeitpunkt

- **Einplanung (scheduling)** bzw. **Umplanung (rescheduling)**:
 - nachdem ein Prozess erzeugt worden ist: *begin*
 - wenn ein Prozess freiwillig die CPU abgibt: *relinquish*
 - falls das von einem Prozess erwartete Ereignis eingetreten ist: *signal*
 - sobald ein Prozess wieder aufgenommen werden kann: *continue*
- Übergänge in den Zustand „bereit“ aktualisieren die **Bereitliste**
 - eine Entscheidung über die Einlastungsreihenfolge wird getroffen
 - eine Funktion der Einplanungsstrategie wird ausgeführt
- die Bereitliste ist je nach Betriebsart ggf. unterschiedlich ausgelegt:
 - als Tabelle, Warteschlange, Vorrangwarteschlange, ...
 - eine Datenstruktur pro Prozessor oder für mehrere Prozessoren
- Prozesse können dazu gedrängt werden, die CPU freiwillig abzugeben
 - sofern **verdrängende (preemptive) Prozesseinplanung** erfolgt



© wosch

SP (SS 2015, C – IX.1)

3.3 Arbeitsweisen – Verdrängung

IX.1/21

Latenzzeiten und Determinismus

Verdrängung als Querschnittsbelang von Betriebssystemen.

- **Einplanungslatenz (scheduling latency)**, die Zeitdauer der Ein- oder Umplanung eines Prozesses
 - ist ggf. vorhersehbar (*predictable*) und **deterministisch**:
 - zu jedem Zeitpunkt ist der nachfolgende Schritt eindeutig festgelegt, unabhängig von Systemlast/-aktivitäten in dem Moment
 - die Latenzzeit ist konstant oder mit fester oberer Schranke variabel
 - sollte kurz sein, um „Hintergrundrauschen“ klein zu halten
- **Einlastungslatenz (dispatching latency)**, die Zeitspanne zwischen Einplanung und Prozessorzuteilung eines Prozesses
 - ereignisbasierte Betriebssysteme lassen Einlastung nur an bestimmten Stellen zu („programmierte Verdrängung“) \sim größere Latenz
 - an einem ausgewählten **Verdrängungspunkt (preemption point)**
 - prozessbasierte Betriebssysteme lassen Einlastung jederzeit zu, sie können **voll verdrängend (full preemptive)** arbeiten \sim kleinere Latenz
 - sofern sie frei von Unterbrechungs- oder Verdrängungssperren sind
 - die Zeitdauer der Einlastung ist i.A. vorhersehbar und deterministisch



© wosch

SP (SS 2015, C – IX.1)

3.3 Arbeitsweisen – Verdrängung

IX.1/23

Verdrängende Prozesseinplanung

Ereignis \rightarrow Signalisierung \rightarrow Einplanung \rightarrow Einlastung \rightarrow Reaktion

- **Verdrängung (preemption)** des laufenden Prozesses bedeutet:
 1. ein Ereignis tritt ein, dessen Behandlungsverlauf zum Planer führt
 - ggf. wird ein Prozess von „blockiert“ in den Zustand „bereit“ überführt³
 2. der (vom Ereignis unterbrochene) **laufende** Prozess wird eingeplant
 - d.h., vom Zustand „laufend“ in den Zustand „bereit“ überführt
 3. der **einzuLASTende** Prozess wird ausgewählt & (wieder) aufgenommen
 - ggf. handelt es sich dabei um den unter 1. eingeplanten Prozess
- Einplanung und Einlastung von Prozessen erfolgt nicht immer zeitnah zum Ereignisbeginn bzw. Moment der Verdrängungsaufforderung
 - ereignisbasierte Betriebssystem(kern)architektur
 - keine kernel threads (vgl. [1, S. 20]) \sim einen Stapel pro Betriebssystemkern
 - Unterbrechungs-/Verdrängungssperre zum Schutz kritischer Abschnitte
- ggf. entstehende **Latenzzeiten** können Anwendungen beeinträchtigen

³Nur bei Zuteilung eines konsumierbaren Betriebsmittels, nicht jedoch bei Ablauf der Zeitscheibe eines Prozesses.



© wosch

SP (SS 2015, C – IX.1)

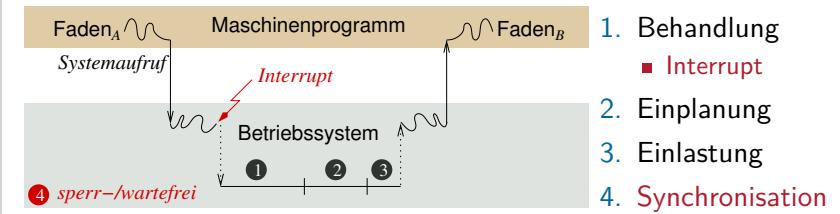
3.3 Arbeitsweisen – Verdrängung

IX.1/22

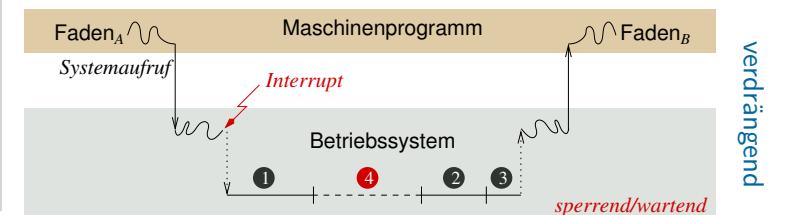
Latenzzeiten in Bezug zum Betriebsmodus

Asynchrone Programmunterbrechungen als Quelle der Ungewissheit.

voll verdrängend



1. Behandlung
 - **Interrupt**
2. Einplanung
3. Einlastung
4. **Synchronisation**



verdrängend



© wosch

SP (SS 2015, C – IX.1)

3.3 Arbeitsweisen – Verdrängung

IX.1/24

Gliederung

- Einführung
- Programmfaden
 - Grundsätzliches
 - Fadenverläufe
 - Leistungsoptimierung
- Arbeitsweisen
 - Ebenen
 - Ebenenübergänge
 - Verdrängung
- Gütemerkmal
 - Benutzerdienlichkeit
 - Systemperformanz
 - Betriebsart
- Zusammenfassung

Benutzerorientierte Kriterien

- charakteristische **Anforderungsmerkmale**:
 - 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.
- je nach **Anwendungsdomäne** mit unterschiedlicher Wichtung



Dimensionen der Prozesseinplanung

- Kriterien zur Aufstellung einer Einlastungsreihenfolge von Prozessen
 - benutzerorientierte Kriterien**
 - fokussieren auf **Benutzerdienlichkeit**
 - d.h. das vom jeweiligen Benutzer wahrgenommene Systemverhalten
 - bestimmen im großen Maße die Akzeptanz des Systems
 - bedeutsam für die Anwendungsdomäne in technischer Hinsicht
 - z.B. Einhaltung und Durchsetzung von Gütemerkmalen
 - systemorientierte Kriterien**
 - fokussieren auf **Systemperformanz**
 - d.h. die effektive und effiziente Auslastung der Betriebsmittel
 - bestimmen im großen Maße die „Rentabilität“ des Systems
 - bedeutsam für die Anwendungsdomäne in kommerzieller Hinsicht
 - z.B. Amortisierung hoher Anschaffungskosten von Großrechnern
- Ausschlusskriterien sind dies nicht, vielmehr **Schwerpunktsetzung**:
 - gute Systemperformanz ist auch der Benutzerdienlichkeit förderlich



Systemorientierte Kriterien

- wünschenswerte Anforderungselemente:
 - Durchsatz** Maximierung der Anzahl vollendeter Prozesse pro vorgegebener Zeiteinheit, d.h., der (im System) geleisteten Arbeit.
 - Prozessorauslastung** Maximierung des Prozentanteils der Zeit, in der die CPU Programme ausführt, d.h., „sinnvolle“ Arbeit leistet.
 - Gerechtigkeit** Gleichbehandlung der Prozesse; Zusicherung, ihnen innerhalb gewisser Zeiträume die CPU zuzuteilen.
 - Dringlichkeiten** Vorzubehandlung des Prozesses mit der höchsten (statischen/dynamischen) Priorität.
 - Lastausgleich** Gleichmäßige Betriebsmittelauslastung; ggf. auch Vorzubehandlung der Prozesse, die stark belastete Betriebsmittel eher selten belegen.
- sollten verträglich zu der jeweiligen Anwendungsdomäne ausgelegt sein



Betriebsart vs. Einplanungskriterien

- Prozesseinplanung impliziert die Rechnerbetriebsart und umgekehrt:
 - **allgemein** Gerechtigkeit, Lastausgleich
 - **Durchsetzung der jeweiligen Strategie**
 - Stapelbetrieb** Durchsatz, Durchlaufzeit, Prozessorauslastung
 - Dialogbetrieb** Antwortzeit
 - Echtzeitbetrieb** Dringlichkeit, Termineinhaltung, Vorhersagbarkeit
 - oft im Konflikt mit Gerechtigkeit/Lastausgleich

Proportionalität

Für bestimmte Prozesse ein Laufzeitverhalten „simulieren“, das nicht unbedingt dem technischen Leistungsvermögen des Rechensystems entspricht:

- es kommt gelegentlich vor, dass Nutzer eine inhärente Vorstellung über die Dauer bestimmter Aktionen des Betriebssystems haben
- aus Gründen der **Nutzerakzeptanz** sollte diesen entsprochen werden



Gliederung

Einführung
Programmfaden
Grundsätzliches
Fadenverläufe
Leistungsoptimierung

Arbeitsweisen
Ebenen
Ebenenübergänge
Verdrängung

Gütemerkmale
Benutzerdienlichkeit
Systemperformanz
Betriebsart

Zusammenfassung



Resümee

... Prozesseinplanung ist querschneidender Belang

- **Einplanungseinheit** für die Prozessorvergabe ist der Faden
 - seine Lauf- und Wartephänen betreiben einen Rechner stoßartig
 - Fäden sind Mittel zum Kaschieren von Totzeiten anderer Fäden
- Betriebssysteme treffen **Zuteilungsentscheidungen** auf drei Ebenen:
 - long-term scheduling* Lastkontrolle des Systems
 - medium-term scheduling* Umlagerung von Programmen
 - short-term scheduling* Einlastungsreihenfolge von Prozessen
- die **Entscheidungskriterien** haben verschiedene **Dimensionen**:
 - Benutzer** Antwort-/Durchlaufzeit, Termine, Vorhersagbarkeit
 - System** Durchsatz, Auslastung, Gerechtigkeit, Dringlichkeit, Lastausgleich
- Durchsetzung der Kriterien impliziert eine bestimmte **Betriebsart**
 - umgekehrt: Betriebsarten erwarten die Durchsetzung gewisser Kriterien



Literaturverzeichnis

- [1] KLEINÖDER, J. ; SCHRÖDER-PREIKSCHAT, W. :
Prozesse.
In: LEHRSTUHL INFORMATIK 4 (Hrsg.): *Systemprogrammierung*.
FAU Erlangen-Nürnberg, 2015 (Vorlesungsfolien), Kapitel 6.1

