

Script generated by TTT

Title: Eingebettete_Systeme (06.11.2012)

Date: Tue Nov 06 14:50:13 CET 2012

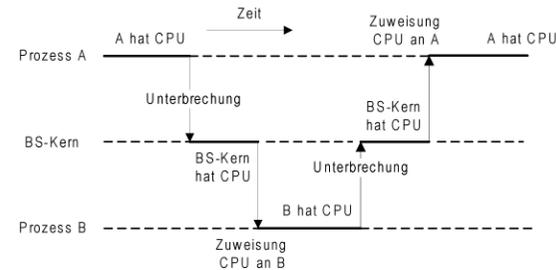
Duration: 75:10 min

Pages: 25

Eine wesentliche Aufgabe der Prozessorverwaltung besteht darin zu entscheiden, welcher der um den bzw. die Prozessor(en) konkurrierenden Prozesse (bzw. Threads) zu einem Zeitpunkt an den bzw. die Prozessor(en) gebunden wird. Dazu steht die BS-Komponente **Scheduler** zur Verfügung.

Prozessablauf besteht aus einer Sequenz von alternierenden CPU- und E/A-Perioden.

Zeitliche Verschränkung der Prozessbearbeitung bei einer CPU.



Kriterien

Scheduling-Strategien

Thread Scheduling

Mehrschichtiges Scheduling

Echtzeit Scheduling

Generated by Targeseam

Es werden zwischen zwei Klassen unterschieden: **nicht-unterbrechende** (nonpreemptive) und **unterbrechende** Strategien (preemptive).

nicht unterbrechend : Scheduling nur dann möglich, wenn der rechnende Prozess blockiert wird oder wenn er terminiert, d.h. Prozess behält CPU bis er sie selber abgibt.

Beispiel: Microsoft Windows 3.x; unterbrechende Strategien erst ab Windows 95

unterbrechend : Unterbrechung beim Eintreten von speziellen Ereignissen, u.a. Eintreffen eines Prozesses mit höherer Priorität oder Prozess geht in Wartezustand.

Zeitscheibenstrategie

Prioritäten

First-Come First-Served

Shortest-Jobs-First

Generated by Targeseam

Dieses nicht-unterbrechende Verfahren (FCFS) teilt einen Prozessor in der Reihenfolge des Auftragseingangs zu. Ready-Queue wird als FIFO-Liste verwaltet; Verfahren einfach zu realisieren.

Ein Kontextwechsel findet nur statt, wenn der Prozess eine blockierende Systemfunktion aufruft oder der Prozess die CPU freiwillig abgibt.

Es kann eine hohe CPU Auslastung erreicht werden.

Problem : Durchschnittliche Wartezeit ist hoch.

Beispiel: Prozesse P1,P2,P3 kommen nahezu gleichzeitig zum Zeitpunkt 0 an;

Dauer ihrer Berechnungszeiten: P1: 24 ms, P2: 3ms, P3: 3ms;

bei Reihenfolge P1, P2, P3: mittlere Wartezeit: $(0 + 24 + 27)/3 = 17$ ms

bei Reihenfolge P2, P3, P1 mittlere Wartezeit $(0+3+6)/3 = 3$ ms

Generated by Targeseam



Dieses Verfahren (SJF) führt die Prozessorzuteilung in der Reihenfolge der wachsenden Rechenphasen ("CPU-Burst") zu, d.h. Prozess mit kürzester, nächster Rechenphase erhält Prozessor als nächster.

anwendbar, falls die Dauer der nächsten Rechenphase bis E/A-Befehl, Interrupt etc. bekannt ist.

Beispiel: P1: 6ms, P2: 8ms, P3: 7ms, P4: 3ms

Schedule bei SFJ : P4, P1, P3, P2; Wartezeit: $(3+16+9+0)/4 = 7$ ms

bei FCFS: 10.25 ms (P1 vor P2 vor P3 vor P4)

Problem: Kenntnis über die Bedienzeiten erforderlich. Für Stapelbetrieb geeignet, da dort Information über Rechenzeiten zur Verfügung stehen (Benutzer geben bei Batch-Jobs Rechenzeit an).

Für interaktive Prozesse wird die Länge der nächsten Rechenphase ("Burst") geschätzt:

$$S_{n+1} = 1/n \sum_{i=1}^n T_i$$

T_i = Ausführungszeit der i-ten Rechenphase.

S_i = Schätzung für die i-te Rechenphase.

S_1 = Schätzung für die erste Rechenphase.

Anpassung der Berechnung

$$S_{n+1} = (1/n) * T_n + (n-1)/n * S_n$$

Generated by Targeteam



Die Prozessorzuteilung von Threads hängt von deren Art der Realisierung ab.

User-Threads

Realisierung der Threads im Benutzeradressraum \Rightarrow Kern hat keine Kenntnis bzgl. der Threads. BS-Scheduler wählt nur Prozess aus. Laufzeitsystem des Prozesses wählt rechenwilligen Thread des Prozesses aus; es kann ein beliebiges [Scheduling-Verfahren](#) für Prozesse verwendet werden.

Java Virtual Machines verwenden unterbrechendes Prioritäten-Scheduling für Threads;

10 ist die höchste und 1 die niedrigste Priorität;

Kernel-Threads

Realisierung der Threads im Systemadressraum \Rightarrow BS-Scheduler wählt den nächsten auszuführenden Thread aus.

a) Ist ausgewählter Thread demselben Prozess zugeordnet wie der vorher rechnende Thread \Rightarrow geringer Kontextwechsel.

b) Ist ausgewählter Thread nicht demselben Prozess zugeordnet wie der vorher rechnende Thread \Rightarrow aufwendiger Kontextwechsel.

Generated by Targeteam



Es werden zwischen zwei Klassen unterschieden: **nicht-unterbrechende** (nonpreemptive) und **unterbrechende** Strategien (preemptive).

nicht unterbrechend : Scheduling nur dann möglich, wenn der rechnende Prozess blockiert wird oder wenn er terminiert, d.h. Prozess behält CPU bis er sie selber abgibt.

Beispiel: Microsoft Windows 3.x; unterbrechende Strategien erst ab Windows 95

unterbrechend : Unterbrechung beim Eintreten von speziellen Ereignissen, u.a. Eintreffen eines Prozesses mit höherer Priorität oder Prozess geht in Wartezustand.

Zeitscheibenstrategie

Prioritäten

First-Come First-Served

Shortest-Jobs-First

Generated by Targeteam



Einlagern eines rechenwilligen Prozesses von der Platte in den Arbeitsspeicher ist aufwendig; deshalb Mehr-Schichten Scheduling.

Short-Term-Scheduler (CPU Scheduler)

Auswahl eines geeigneten Prozesses aus der Ready-Queue; wird häufig aufgerufen; Verfahren siehe oben.

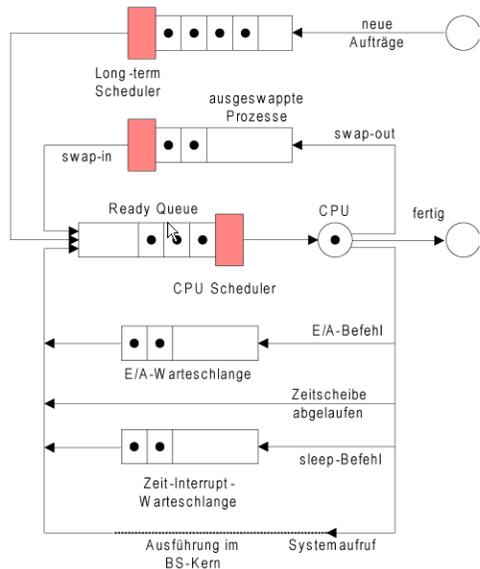
Long-Term-Scheduler

Auswahl rechenwilliger neuer Aufträge (meist Jobs aus dem Hintergrundbetrieb (batch)) und Einlagerung in den Arbeitsspeicher; Einfügen der Prozesse in die Ready-Queue.

Graphische Darstellung

Generated by Targeteam





Generated by Targeteam

Prozesse in Multimedia-Anwendungen führen oft zeitkritische Operationen aus. Bzgl. des Scheduling existieren zwei gegensätzliche Ziele:

- a) ein **unkritischer Prozess** sollte nicht dauerhaft blockiert werden, weil zeitkritische Prozesse eine Ressource gänzlich auslasten.
- b) **zeitkritische Prozesse** dürfen nicht durch Scheduling-Verfahren (Round-Robin oder Prioritäten) am zeitkritischen Fortschritt gehindert werden ⇒ Einhaltung von Zeitvorgaben.

Zuordnung von Kenngrößen zu zeitkritischen Prozessen

Bereitzeit (ready time): frühestmöglicher Ausführungsbeginn einer Aktivität.

Frist (deadline): spätester Zeitpunkt für die Beendigung einer Aktivität.

Ausführungszeit: worst-case Abschätzung für das zur vollständigen Ausführung einer Aktivität notwendige Zeitintervall.

[Earliest Deadline First \(EDF\)](#)

[Rate-Monotonic Scheduling](#)

Generated by Targeteam

Der Prozessor wird immer dem Prozess mit der am nächsten in der Zukunft liegenden Frist zugeordnet. Es existieren die beiden Varianten: nicht-unterbrechend und unterbrechend.

nicht-unterbrechend

Eine Prozessorzuordnung bleibt bis der Prozess eine blockierende Systemfunktion aufruft oder freiwillig die CPU abgibt. Neue eintreffende Prozesse mit kürzeren Fristen werden erst beim nächsten Scheduling berücksichtigt.

unterbrechend

Diese Variante führt einen Kontextwechsel durch, wenn ein Prozess mit einer kürzeren Frist rechenwillig wird.

Generated by Targeteam

Rate-Monotonic Scheduling (RMS) ist für periodische Prozesse; RMS ordnet Prioritäten in Abhängigkeit von der Periode zu.

- 1) Prozesse mit der höchsten Frequenz (kleinste Periode) erhalten die höchste Priorität.
- 2) Prozesse mit der geringsten Frequenz (längste Periode) erhalten die niedrigste Priorität.

Auswahl der Prozesse anhand ihrer Priorität.

hochfrequente Prozesse werden minimal verzögert.

Zerstückelung niederfrequenter Prozesse, da sie häufig wegen hochfrequenter Prozesse unterbrochen werden.

Generated by Targeteam



Ein Prozess ist der Ablauf eines Programms in einem Rechner. Dieser Ablauf ist eine Verwaltungseinheit im jeweiligen Betriebssystem.

Fragestellungen

Dieser Abschnitt gibt eine kurze Einführung in eine der wichtigen Verwaltungsaufgaben eines Betriebssystems:

Verwaltung von Prozessen.

Verwaltung des Prozessors, d.h. Zuteilung der CPU an rechenbereite Prozesse (Scheduling).

Unterbrechungskonzept.

Prozessverwaltung

Prozessorverwaltung

Unterbrechungskonzept

Generated by Targeteam



Die optimale Ausnutzung und Auslastung aller Geräte eines Rechner-Systems legt Mehrprogrammbetrieb nahe. Zerlegung der Ausführungsphasen eines Programms in viele Einzelteile;

- Aufbrechen der Ausführung eines Prozesses in mehrere Phasen: u.a. Rechnen, E/A, Synchronisieren mit Partner.

- Die Ausführung der Programme wird in der Regel mehrfach unterbrochen.

Motivation

Unterbrechungsarten

Behandlung externer Unterbrechungen

Konflikte

Generated by Targeteam



Ursachen für Unterbrechungen

zugeeilte Prozessorzeit ist aufgebraucht;

benötigte Ressourcen stehen aktuell nicht zur Verfügung;

ein E/A-Gerät meldet sich zurück;

ein Fehler tritt auf, z.B. Division durch 0;

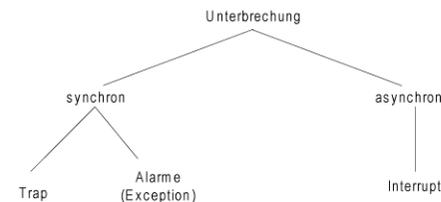
Systemaufruf (wurde bereits als spezielle Unterbrechung eingeführt);

Bei einer Unterbrechung wird ein gerade aktiver Prozess unterbrochen und eine **Unterbrechungsbehandlung** durchgeführt.

Es ist erforderlich, bei der Unterbrechung den CPU Status des gerade aktiven Prozesses für die spätere Fortsetzung zu speichern.

Forderung: Eine Unterbrechung muss so **kontrolliert** erfolgen, dass ein definierter Prozessstatus festgehalten werden kann.

Generated by Targeteam



Externe, asynchrone Unterbrechungen

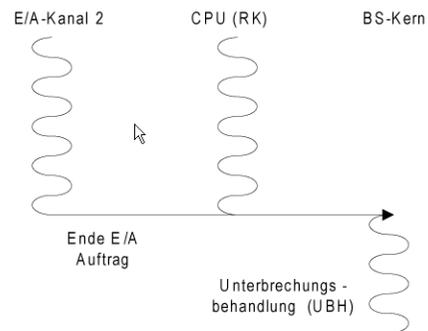
Interne, synchrone Unterbrechungen

Generated by Targeteam

Externe, asynchrone Unterbrechungen

Dies sind Unterbrechungen (Interrupts), die von außerhalb des zu unterbrechenden Prozesses ausgelöst werden, z.B. E/A-Kanal-"Endmeldung".

Der Ablauf im Rechnerkern (RK) wird unterbrochen und eine Unterbrechungsanfangsbehandlung des Betriebssystemkerns wird aktiviert.

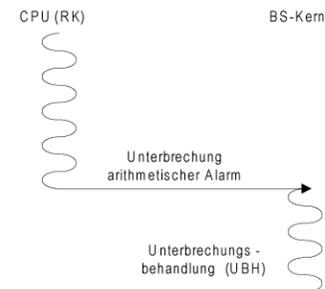


Generated by Targem.com

Interne, synchrone Unterbrechungen

Dies sind Unterbrechungen (Alarme, Exceptions), die durch den zu unterbrechenden Prozess selbst ausgelöst werden, z.B. Division durch 0.

Der Ablauf im RK wird unterbrochen und eine Unterbrechungsanfangsbehandlung des Systemkerns wird aktiv.



Beispiele von internen Unterbrechungen

Speicherschutzalarm: Prozess greift auf Speicherbereich zu, der nicht vorhanden ist oder auf den er nicht zugreifen darf.

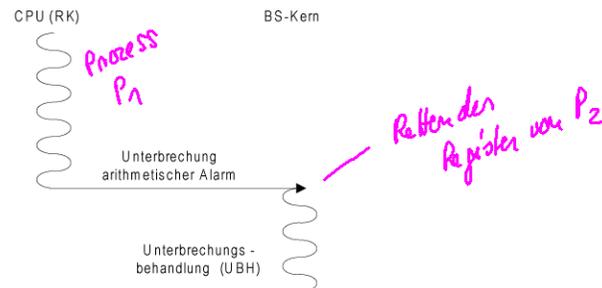
Befehlsalarm: dem Operationscode des Maschinenbefehls ist keine Operation zugeordnet.

Seitenehltalarm: Seite der virtuellen Adresse ist nicht im Arbeitsspeicher.

arithm. Alarm: arithm. Operation kann nicht ausgeführt werden, z.B. Division durch 0.

Interne, synchrone Unterbrechungen

Der Ablauf im RK wird unterbrochen und eine Unterbrechungsanfangsbehandlung des Systemkerns wird aktiv.



Beispiele von internen Unterbrechungen

Speicherschutzalarm: Prozess greift auf Speicherbereich zu, der nicht vorhanden ist oder auf den er nicht zugreifen darf.

Befehlsalarm: dem Operationscode des Maschinenbefehls ist keine Operation zugeordnet.

Seitenehltalarm: Seite der virtuellen Adresse ist nicht im Arbeitsspeicher.

arithm. Alarm: arithm. Operation kann nicht ausgeführt werden, z.B. Division durch 0.

Systemaufruf: kontrollierter Übergang in das Betriebssystem.

Generated by Targem.com

Ablauf

Geräte-Controller meldet Unterbrechung über spezielle Interrupt-Leitung an CPU.

CPU prüft im Befehlszyklus nach jeder Befehlsausführung, ob eine Unterbrechung gemeldet wurde.

Falls Unterbrechung vorliegt: sichern u.a. des aktuellen Befehlszählers, des Programmstatusworts und Sprung zu einer Unterbrechungsanfangsbehandlung, die an festgelegter Speicheradresse steht.

Routine untersucht Unterbrechungsursache, die vom Controller über Datenleitung gemeldet wird (Unterbrechungsnummer).

über Unterbrechungsnummer erfolgt die Auswahl der benötigten Unterbrechungsbehandlungsroutine; Nummer ist i.a. Index in eine Tabelle, dem **Unterbrechungsvektor**.

Vektor enthält Speicheradresse der Unterbrechungsbehandlungsroutine.

Generated by Targem.com



Konflikte bei Unterbrechungen treten z.B. in folgenden Situationen auf:

- (1) während einer Unterbrechungsbehandlung treten weitere Unterbrechungen auf;
- (2) es treffen gleichzeitig mehrere Unterbrechungswünsche ein.

Beispiel

Mögliche Konfliktlösungen

Integration der Unterbrechungsbehandlung in den Befehlszyklus der CPU

prüfen ob interne Unterbrechung aufgetreten,

fals ja, Behandlung der Unterbrechung



sonst : prüfen ob externe Unterbrechung mit höherer Priorität. Wenn ja wähle eine mit höchster Priorität.

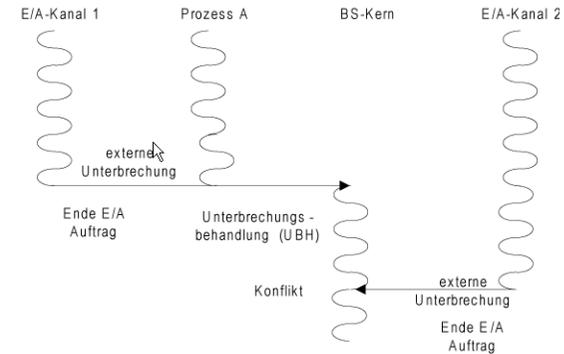
Bei Unterbrechung: sichere alten Zustand, stelle neuen Zustand her und führe ersten Befehl der Unterbrechungsbehandlungsroutine aus.

Generated by Targeteam

Beispiel



E/A-Kanal 1 und E/A-Kanal 2 erledigen beide Aufträge für Prozess A.



Generated by Targeteam



Konflikte bei Unterbrechungen treten z.B. in folgenden Situationen auf:

- (1) während einer Unterbrechungsbehandlung treten weitere Unterbrechungen auf;
- (2) es treffen gleichzeitig mehrere Unterbrechungswünsche ein.

Beispiel

Mögliche Konfliktlösungen

Integration der Unterbrechungsbehandlung in den Befehlszyklus der CPU

prüfen ob interne Unterbrechung aufgetreten,

fals ja, Behandlung der Unterbrechung

sonst : prüfen ob externe Unterbrechung mit höherer Priorität. Wenn ja wähle eine mit höchster Priorität.

Bei Unterbrechung: sichere alten Zustand, stelle neuen Zustand her und führe ersten Befehl der Unterbrechungsbehandlungsroutine aus.

Generated by Targeteam

Mögliche Konfliktlösungen



Andere Unterbrechungen nicht zulassen, d.h. **Maskierung** von Unterbrechungen; anstehende Unterbrechung ignorieren oder vorläufig zurückstellen; Problem: u.a. **Rechtzeitigkeit** der Unterbrechungsbehandlung.

Interne Unterbrechungen erfolgen stets sofort und geben der zugehörigen Unterbrechungsbehandlung dieselbe Priorität, wie sie der unterbrochene Ablauf hatte.

Externe Unterbrechungen erhalten **Prioritäten** z.B. (0,...,31) zugeordnet. Die aufgerufene Unterbrechungsbehandlung erhält die **Priorität** (Ablaufpriorität) der auslösenden Unterbrechung.

Behandlung neu eintreffender externer Unterbrechung, falls **Priorität höher als Ablaufpriorität**; andernfalls zurückstellen.

Generated by Targeteam



Konflikte bei Unterbrechungen treten z.B. in folgenden Situationen auf:

- (1) während einer Unterbrechungsbehandlung treten weitere Unterbrechungen auf;
- (2) es treffen gleichzeitig mehrere Unterbrechungswünsche ein.

Beispiel

Mögliche Konfliktlösungen

Integration der Unterbrechungsbehandlung in den Befehlszyklus der CPU

prüfen ob interne Unterbrechung aufgetreten,

falls ja, Behandlung der Unterbrechung

sonst : prüfen ob externe Unterbrechung mit höherer Priorität. Wenn ja wähle eine mit höchster Priorität.

Bei Unterbrechung: sichere alten Zustand, stelle neuen Zustand her und führe ersten Befehl der Unterbrechungsbehandlungsroutine aus.

Generated by Targeteam



Teil I: Betriebssysteme und Systemsoftware

- Prof. J. Schlichter
 - Lehrstuhl für Angewandte Informatik / Kooperative Systeme, Fakultät für Informatik, TU München
 - Boltzmannstr. 3, 85748 Garching

Email: schlichter@in.tum.de

Tel.: 089-289 18654

URL: <http://www11.informatik.tu-muenchen.de/>

Übersicht

Einführung

Synchronisation und Verklemmungen

Prozess- und Prozessorverwaltung

Speicherverwaltung

Prozesskommunikation

Zusammenfassung

Generated by Targeteam