Script **generated by TTT**

Title: Eingebettete_Systeme (30.10.2012)

Date: Tue Oct 30 16:22:08 CET 2012

Duration: 61:47 min

Pages: 18

Prozessverwaltung

Dieser Abschnitt behandelt das Prozesskonzept, Datenstrukturen zur Beschreibung des aktuellen Prozesszustandes sowie Dienste zum Aufruf von Systemfunktionen.

Prozesse repräsentieren eine Aktivität; sie haben ein Programm, Eingaben, Ausgaben und einen Zustand.

Aufgabe des Dispatchers: Realisieren der Zustandsübergänge zwischen ***rechnend*** und ***rechenwillig*** : Prozessor binden und entbinden. Dazu ist ein Kontextwechsel erforderlich.

Kontextwechsel

CPU wird entzogen und einer anderen Aktivität zugeteilt; ein Kontextwechsel ist erforderlich, falls der rechnende Prozess P1 in den Zustand ***wartend*** oder z.B. durch Prozessorentzug in den Zustand ***rechenwillig*** übergeführt wird.

Problem

aktueller Ausführungskontext des Prozesses muss gesichert werden und Kontext des nächsten rechenbereiten Prozesses muss geladen werden.

Achtung : je umfangreicher ein PCB ist, desto "teurer" sind Prozesswechsel, d.h. das Umschalten der CPU zwischen den Prozessen.

Threads

Threads haben einen sehr viel kleineren Kontext ⇒ Umschalten zwischen Threads innerhalb eines Prozesses sehr schnell, da Adressraum und andere Ressourcen (z.B. Dateien) gemeinsam genutzt werden.

Ziel für den Betrieb von Rechensystemen: kontrollierter Zugriff auf Hardwarekomponenten nur durch BS.

Lösung: alle Zugriffe auf Hardware nur über ***privilegierte*** Befehle zulässig;

Frage: wer darf privilegierte Befehle ausführen ⇒ Antwort: Prozesse in einem privilegierten Modus.

Herkömmlich unterscheidet man zwischen dem Benutzer- (engl. user mode) und dem Systemmodus (engl. kernel mode).

Benutzermodus

nur nicht privilegierten Befehle; Zugriff auf Prozessadressraum und unkritische Register (Befehlszähler, Indexregister); Benutzerprozesse im Benutzermodus.

Systemmodus

Es sind auch die privilegierten Befehle verfügbar (z.B. Anhalten der Maschine, Verändern der Ablaufpriorität). Die Dienste des Betriebssystemkerns werden im Systemmodus ausgeführt.

Nutzung der Hardware-Komponenten nur über Dienste des BS: Aufruf eines BS-Dienstes über spezielle Befehle: den ***Systemaufruf*** .

Systemaufrufe

Ein Systemaufruf ist eine Funktion, die von einem Benutzerprozess aufgerufen wird, um einen BS-Kerndienst aufzuführen.

1. Der Systemaufruf überprüft die übergebenen Parameter und bildet daraus eine Datenstruktur, um die Daten an den BS-Kern weiterzureichen.

2. Danach wird eine spezielle Instruktion, ein Software Interrupt (Trap), ausgeführt. Diese Instruktion identifiziert über einen Operanden den gewünschten Systemdienst.

3. Bei Ausführung der Trap-Instruktion wird der Zustand des Benutzerprozesses gerettet und es findet ein Wechsel in den Systemmodus statt.

Beispiel

Beispiel

Lesen von Daten aus einer Datei und Kopieren in eine andere Datei. Dabei treten die folgenden Systemaufrufe auf:

(1) Schreiben des Prompts auf Bildschirm: Angabe der Dateinamen

(2) Lesen der Tastatureingabe (bzw. Analoges bei Mouse-Eingabe)

(3) Öffnen der zu lesenden Datei (open)

(4) Erzeugen der neuen Datei

(5) ggf. Fehlerbehandlung: Nachricht auf Bildschirm

(6) Schleife: Lesen von Eingabedatei (ein Systemaufruf) und schreiben in zweite Datei (auch Systemaufruf)

(7) Schließen beider Dateien

(8) Ausgabe auf Bildschirm

Durch die Verwendung von Laufzeitroutinen ergibt sich eine höhere Abstraktionsebene ⇒ Aufruf einer Routine, die die notwendigen Systemaufrufe durchführt.

Prozessverwaltung

Dieser Abschnitt behandelt das Prozesskonzept, Datenstrukturen zur Beschreibung des aktuellen Prozesszustandes sowie Dienste zum Aufruf von Systemfunktionen.

Prozesse repräsentieren eine Aktivität; sie haben ein Programm, Eingaben, Ausgaben und einen Zustand.

Prozesskonzept

Dispatcher

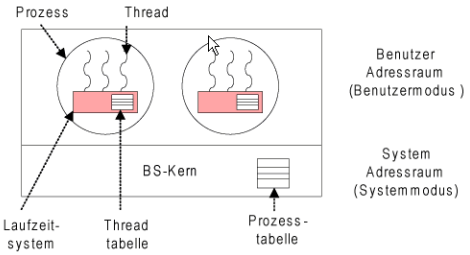
Arbeitsmodi

Systemaufrufe

Realisierung von Threads

im Benutzer-Adressraum

Der BS-Kern verwaltet nur single-threaded Prozesse.



Threads werden durch Laufzeitsystem im Benutzeradressraum verwaltet. Eine Thread-Tabelle speichert Informationen (Register, Zustand, etc.) über Threads pro Prozess.

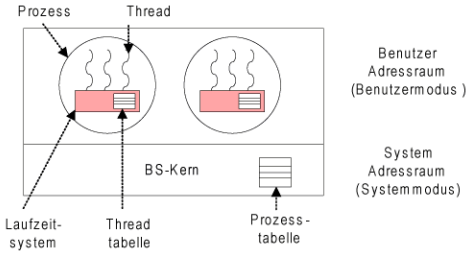
Prozessorzuteilung im BS-Kern erfolgt an Prozesse. Laufzeitsystem bestimmt, welcher Thread rechnend gesetzt wird.

Problem: Systemaufruf eines Threads blockiert die anderen Threads des Prozesses.

Problem: wie wird einem Thread die CPU entzogen?

im Benutzer-Adressraum

Der BS-Kern verwaltet nur single-threaded Prozesse.



Threads werden durch Laufzeitsystem im Benutzeradressraum verwaltet. Eine Thread-Tabelle speichert Informationen (Register, Zustand, etc.) über Threads pro Prozess.

Prozessorzuteilung im BS-Kern erfolgt an Prozesse. Laufzeitsystem bestimmt, welcher Thread rechnend gesetzt wird.

Problem: Systemaufruf eines Threads blockiert die anderen Threads des Prozesses.

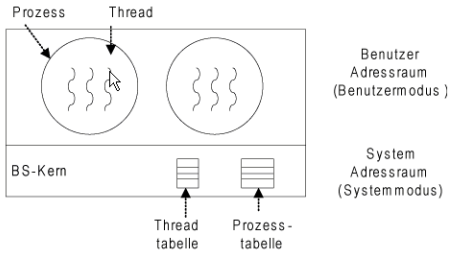
Problem: wie wird einem Thread die CPU entzogen?

Es existieren zwei grundlegende Ansätze, Threads in einem Rechensystem zu realisieren: im ***Benutzer-Adressraum*** (Benutzermodus) oder im ***System-Adressraum*** (Systemmodus).

im Benutzer-Adressraum

im System-Adressraum

Neben den Prozessen werden im BS-Kern auch alle Threads verwaltet.



Thread-Tabelle speichert Informationen (Register, Zustand, etc.) über Threads.

Prozessorzuteilung im BS-Kern erfolgt an Threads.

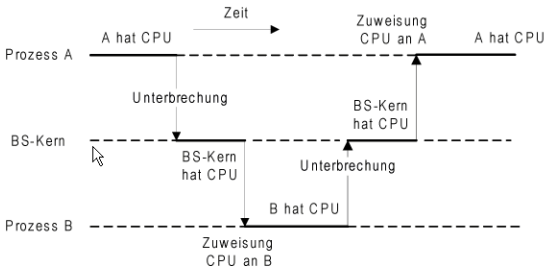
Der Systemaufruf eines Threads blockiert nicht die anderen Threads des Prozesses.

Prozessorverwaltung

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

Der Scheduler wählt aus der Menge der rechenwilligen Prozesse den nächsten auszuführenden Prozess aus. Es existieren unterschiedliche Verfahren die von der jeweiligen Prozessorverwaltungsstrategie abhängen. Mögliche ***Leistungskriterien*** für ein Schedulingverfahren:

- Fairness.
- Effizienz, Prozessorauslastung.
- Antwortzeit für interaktive Benutzer (Dialogverarbeitung).
- Wartezeit, insbesondere für Batch-Jobs (Stapelverarbeitung).
- Ausführungszeit, d.h. Zeitspanne von Auftragsbeginn bis Auftragsende.
- Abschlusszeit, insbesondere für Realzeitsysteme.
- Durchsatz, Anzahl der Aufträge pro Zeiteinheit.

Kriterien der Betriebsarten

Kriterien der Betriebsarten

Die Ziele der Schedulingverfahren hängen von der Umgebung und den Betriebsarten des Rechensystems ab.

Alle Systeme

- Fairness - jeder Prozess bekommt Rechenzeit der CPU.
- Balance - alle Teile des Systems sind ausgelastet.
- Policy Enforcement - Scheduling Policy wird nach außen sichtbar durchgeführt.

Stapelbetrieb

- Durchsatz - maximiere nach Aufträge pro Zeiteinheit.
- Ausführungszeit - minimiere die Zeit von Start bis zur Beendigung.
- CPU-Belegung - belege CPU konstant mit Aufträgen.

Dialogbetrieb - interaktive Systeme

- Antwortzeit - antworte schnellstmöglich auf Anfragen.
- Proportionalität - Erwartungshaltung der Nutzer berücksichtigen.

Echtzeitsysteme

- Abschlusszeit - kein Verlust von Daten.
- Vorhersagbarkeit - Qualitätsverlust bei Multimedia vermeiden.

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

Zeitscheibenstrategie

Die ***Zeitscheibenstrategie*** (Round Robin) ist unterbrechend. Ziel ist die gleichmäßige Verteilung der Rechenzeit auf rechenwillige Prozesse.

Es werden die Prozesse an den Prozessor jeweils für ein festgelegtes Zeitquantum q gebunden und spätestens nach dem Ablauf dieser Zeitspanne wird den Prozessen der Prozessor wieder entzogen.

zyklisches Bedienen der Prozesse (Round Robin).

Ready-Queue (Liste der rechenwilligen Prozesse) als zyklische Warteschlange realisiert.

Wahl des Zeitquantums:

- falls q zu klein: viele unproduktive Kontextwechsel.
- falls q zu groß: Round Robin wird zu einem reinen FCFS Scheduling (First Come First Served), da die Wahrscheinlichkeit für einen Aufruf eines blockierenden Systemdienst steigt.
- Typische Werte für q: 10 bis 100 Millisekunden.

Für q = 100 ms gilt bei 1 MIPS Maschine (Million Instructions/Second): ca. 100.000 Instruktionen/q.

Diese Strategie ist i.a. unterbrechend. Sie basiert darauf, an die Prozesse ***Prioritäten*** zu vergeben. Die Prioritätenvergabe kann dabei statisch oder dynamisch sein.

Prioritäten sind i.a. ein festgelegter Zahlenbereich, z.B. 0, ..., 7.

Statische Prioritätenvergabe

jeder Prozess besitzt für die Dauer seiner Existenz eine feste Priorität.

Problem: Gefahr des Verhungerns von Prozessen mit niedriger Priorität

Lösung: Erhöhung der Priorität von lange wartenden Prozessen, d.h. dynamische Prioritäten.

Dynamische Prioritätenvergabe

die Prioritäten der Prozesse können sich dynamisch verändern, d.h. sie werden in gewissen Zeitabständen neu berechnet.

Idee: lange Wartezeiten berücksichtigen (Erhöhen die Priorität).

Prozesse mit großem CPU-Verbrauch sinken in Priorität.

E/A-intensive Prozesse steigen in Priorität (damit E/A-Geräte und CPU parallel genutzt werden).

Zeitscheibenstrategien und Prioritätenvergabe können zu effizienten Verwaltungsstrategien kombiniert werden.

Windows XP Scheduling

Windows XP nutzt eine Prioritäten-basierte, unterbrechende Strategie. Prozess/Thread mit höchster Priorität wird ausgeführt, bis

er terminiert, oder

er seine Zeitscheibe ausgeschöpft hat, oder

eine Blockierung (Systemaufruf, E/A) auftritt.

Es werden Prioritäten von 0 - 31 unterschieden, die in verschiedene Klassen eingeteilt werden

	Echtzeit	hoch	über normal	normal	unter normal	idle
zeit kritisch	31	15	15	15	15	15
höchste	26	15	12	10	8	6
über normal	25	14	11	9	7	5
normal	24	13	10	8	6	4
unter normal	23	12	9	7	5	3
niedrigst	22	11	8	6	4	2
idle	16	1	1	1	1	1

Prioritäten

Diese Strategie ist i.a. unterbrechend. Sie basiert darauf, an die Prozesse ***Prioritäten*** zu vergeben. Die Prioritätenvergabe kann dabei statisch oder dynamisch sein.

Prioritäten sind i.a. ein festgelegter Zahlenbereich, z.B. 0, ..., 7.

Statische Prioritätenvergabe

jeder Prozess besitzt für die Dauer seiner Existenz eine feste Priorität.

Problem: Gefahr des Verhungerns von Prozessen mit niedriger Priorität

Lösung: Erhöhung der Priorität von lange wartenden Prozessen, d.h. dynamische Prioritäten.

Dynamische Prioritätenvergabe

die Prioritäten der Prozesse können sich dynamisch verändern, d.h. sie werden in gewissen Zeitabständen neu berechnet.

Idee: lange Wartezeiten berücksichtigen (Erhöhen die Priorität).

Prozesse mit großem CPU-Verbrauch sinken in Priorität.

E/A-intensive Prozesse steigen in Priorität (damit E/A-Geräte und CPU parallel genutzt werden).

Zeitscheibenstrategien und Prioritätenvergabe können zu effizienten Verwaltungsstrategien kombiniert werden.

Windows XP Scheduling

Generated by Targeteam