**Prozessverwaltung**

Begriffe

Sequenzielles Programm - besitzt genau einen Kontrollfluss

Nebenläufiges Programm - mehrere Kontrollflüsse, dürfen parallel ausgeführt werden  
 - auf Mehrprozessorarchitektur Kontrollflüsse eigenen  
 Prozessoren zugeordnet und so parallel

Nebenläufigkeit - maximale Parallelität, die mit unbegrenzt vielen Prozessoren  
 erreichbar ist

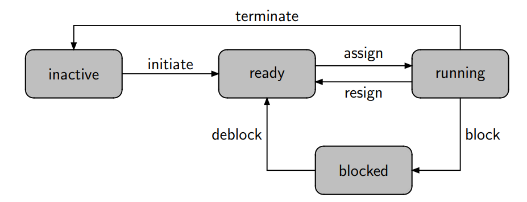
Parallelität - tatsächlich erreichte Grad an paralleler Ausführung  
 (begrenzt durch Prozessoranzahl)

Sequenzieller Prozess - stellt die Ausführungsumgebung für ein sequenzielles Programm  
 Dafür benötigt: Adressraum, das Maschinenprogramm, ein  
 Aktivitätsträger (Prozessorkern), eine Liste der belegten  
 Betriebsmittel

Nebenläufiger Prozess - stellt die Ausführungsumgebung für ein nebenläufiges Programm  
 Unterschied: mehrere Aktivitätsträger

Implementierung von Prozessen

Zustandsmodell Zustände:

- running Dem Prozess ist ein Prozessor zugeordnet

- ready Der Prozess beteiligt sich am Wettbewerb um  
 einen Prozessor (besitzt alle Betriebsmittel)

- blocked Prozess nicht ablaufbereit (wartet)

- inactive Prozess ist keinem Programm zugewiesen

Es liegt ein geschlossenes Modell vor, in dem Prozesse als Objekte auftreten. Zu jedem Zeitpunkt befindet sich ein Prozess in genau einem Zustand.

Übergangsfunktionen

block ( q ) Prozess wird blockiert und in Warteschlange q eingefügt

deblock ( q ) Prozess wird entblockt und in ready-Zustand überführt

assign ( ) Einen bereiten Prozess auswählen (Zuordnung des Prozessors)

resign ( ) Laufenden Prozess in ready überführen

initiate ( context ) Inaktiven Prozess auswählen (Initialisierung der Prozessumgebung  
 mit den Werten aus context) überführt in ready

terminate ( ) laufende Prozess gibt Prozessor ab, da Programmausführung beendet  
 Überführung in inactive

AUF BLOCK, RESIGN UND TERMINATE MUSS IMMER ASSIGN FOLGEN SONST PROZESSOR NICHT BELEGT!

Die Betriebssystemkomponente Prozessverwaltung

Dispatcher und Scheduler

- Einordnung des Prozesszustandsmodells  
 (gehört zur untersten Betriebssystemschicht (elementarer Dienst des Kerns))

Dispatcher  
 - Durchführung der Zustandsübergänge  
 - Implementierung der Übergangsfunktionen

Scheduler  
 - Aufruf beim Übergang von ready nach running  
 - Auswahl des nächsten Prozesses, der den Prozessor erhält

Prozesskontrollblock

- Darstellung eines Prozesses im Dispatcher

- Enthält: - Prozesszustand   
 - Sicherungsbereich für Registerinhalt  
 - Adressrauminformationen  
 - Scheduling-Informationen  
 - belegte Betriebsmittel (z.B. geöffnete Dateien)  
 - weitere Verwaltungsdaten

storeRegister() Abspeichern der aktuellen Registerwerte innerhalb des  
 Sicherungsbereichs des PCBs

loadRegister() Überschreiben der Registerinhalte des Prozessors mit den im PCB  
 gesicherten Werten

initPcb ( Context c ) Der Prozesskontrollblock wird mit allen notwendigen Informationen  
 initialisiert

put ( Pcb process ) Der Prozess wird an das Ende der Warteschlange angehängt

Pcb get () Der Prozess, der vorne in der Warteschlange steht, wird zurückgeliefert

Prozesswarteschlangen

Auf einen Prozess wird durch eine eindeutige PID (Process ID) verwiesen   
 Innerhalb der Prozessverwaltung wird die Referenz auf den PCB verwendet, der dem Prozess  
 zugeordnet ist.

Probleme mit nebenläufigem Code

Doppelter Abruf/Bearbeitung

Unterbrechungssperren

- Unterbrechungen werden während der Ausführung des kritischen Abschnittes verboten  
 - Bei Mehrprozessorsystemen nicht ausreichend  
 ⇒ Anderen Prozessoren könnten weitere ausgeschlossene kritische Abschnitte ausführen

Nachteile  
 - Herabgesetzte Reaktionsfähigkeit auf externe Unterbrechungen  
 - Mögliche Folge: Verlust von Daten bei E/A-Operationen   
 - Unterbrechungen können nur im privilegierten Zustand gesperrt werden  
 - Fehleranfälligkeit: Vergisst ein Prozess, die Unterbrechungen wieder zuzulassen, führt dies  
 zu einer Monopolisierung des Prozessors.

Atomare Speicheroperationen

- Falls mehrere Prozessoren gleichzeitig versuchen, einen Wert in dasselbe Speicherwort zu  
 speichern, trifft die Hardware die Entscheidung, welche zuerst ausgeführt wird.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Dekker Algorithmus

Bewertung

+ Es funktioniert

- Nur zwei Prozesse

- nur kurze Abschnitte

Test&Set-Befehl

Grundlage für eine Realisierung des gegenseitigen Ausschlusses kurzer kritischer Abschnitte bei Mehrprozessorsystemen. ( → Spin Locks)

+ Eintritts- und Austrittsprotokoll sind unabhängig von der Prozessanzahl und  
 Prozessidentifikationen  
 + Durch die Parametrisierung mit einer globalen Variablen kann der gegenseitige Ausschluss  
 auf bestimmte kritische Abschnitte eingeschränkt werden

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung - Aktives Warten (Busy Waiting)  
 - Aushungern (Starvation) eines Prozesses kann auftreten

**Implementierung der Prozessverwaltung**

Annahmen:

* nur eine einzige ready-Warteschlange
* Alle Operationen der Prozessverwaltung werden innerhalb der Klasse ProcessManagement beschrieben (können direkt auf die Datenstrukturen der Prozessverwaltung zugreifen)

Struktur der Prozessverwaltung

* Operationen des Dispatchers:   
  Die Zustands-Übergangsfunktionen sind interne Operationen der Prozessverwaltung (Prozesse können nicht direkt auf diese Operationen zugreifen → Geheimnisprinzip)
* Schnittstellenoperationen der Prozessverwaltung (nach außen sichtbare System Calls): Erzeugung/Vernichtung von Prozessen; Semaphor-Operationen; Ein-/Ausgabeoperationen;
* Synchronisationskonzept  
  Alle Schnittstellenoperationen der Prozessverwaltung werden unter gegenseitigem Ausschluss ausgeführt  
  ⇒ Konsistenz der internen Datenstrukturen (z. B. Warteschlangen) ist sichergestellt
* Unterbrechungsbehandlungsroutinen (Interrupt-Handler):  
  Integration als Schnittstellenoperationen der Prozessverwaltung !Sie dürfen jedoch nicht direkt von Anwendungsprogrammen aufgerufen werden! Damit werden diese Operationen unter gegenseitigem Ausschluss ausgeführt

assign-Operation

* führt zum Verlust des Prozessors für den aufrufenden Prozess (P1)
* P1 läuft weiter, wenn ein anderer Prozess (P2) seinerseits assign aufgerufen hat und damit P1 die Beendigung seines assign-Aufrufes ermöglicht
* Die Wartezeit, bis (P1) den Prozessor wieder erhält, ist für (P1) transparent.

**Scheduling**

Zielsetzung

* Verwaltung des Betriebsmittels Prozessor
* Vermeidung der Prozessor-Monopolisierung durch eine Anwendung
* Unterstützung der verschiedenen Betriebsarten eines Betriebssystems:

- Dialogbetrieb schnelle Reaktion auf Benutzereingaben

* Auswahl von Prozessen, die nach dem Aufruf einer blockierenden Ein-/Ausgabeoperation durch eine Benutzereingabe wieder ausführbar sind

- Stapelbetrieb Hoher Durchsatz sowie hohe Auslastung der Hardware-Ressourcen

* Bevorzugte Auswahl von Prozessen, die gerade freie Betriebsmittel ben¨otigen

- Echtzeitbetrieb Einhaltung von Zeitgarantien

* Bevorzugte Auswahl von Prozessen, die gerade freie Betriebsmittel benötigen

Anwendungbereiche

* Kurzzeitablaufsteuerung
* Langzeitablaufsteuerung (Swapping/Auftragsverwaltung)

Mögliche Scheduling-Kriterien

* Prozessorauslastung
* Durchsatz
* Wartezeit
* Umlaufzeit
* Antwortzeit
* Echtzeitfähigkeit
* Leistungsfähigkeit

Klassifikation von Scheduling-Verfahren

* Kooperatives Scheduling (Cooperative Scheduling)  
  Die schedule-Operation kann nur ausgeführt werden, wenn ein Prozess von sich aus den Prozessor abgibt
* Verdrängendes Scheduling (Preemptive Scheduling)  
  Die System-Software kann einem Prozess den Prozessor zu entziehen

Scheduling-Verfahren

* First-Come-First-Served (FCFS)
  + Auswahl des Prozesses, der bereits am längsten im ready-Zustand ist
  + Keine Verdrängung → der Prozess läuft, bis er sich blockiert oder terminiert
  + Einfache Implementierung → Verwendung einer FIFO-Warteschlange
  + Bei der Auswahl des nächsten Prozesses wird insb. die aktuelle Systemauslastung nicht berücksichtigt
* Shortest-Processing-Time-First (SPTF)
  + Auswahl des Prozesses mit der kürzesten Rechenzeit bis zum nächsten blocked/inactive-Zustand ( → kürzester CPU-Burst)
  + Strategie minimiert die mittlere Gesamtwartezeit der Prozesse
  + Problem: Wie wird der CPU-Burst abgeschätzt?
* Shortest-Remaining-Time-First (SRTF)
  + Wird ein anderer Prozess mit einem kürzeren CPU-Burst ablaufbereit, so erhält dieser den Prozessor
  + Kürzerer CPU-Burst heißt hier, dass dessen CPU-Burst kürzer als die Restbearbeitungszeit des CPU-Burst des gerade laufenden Prozesses ist
* Round Robin (RR)
  + Ziel: Gleichmäßige Aufteilung der verfügbaren Rechenzeit auf alle ablaufbereiten Prozesse
  + Verwendung des Zeitscheibenverfahrens: Die Hardware muss ein Zeitgeberregister besitzen
  + Laden des Registers durch die Operation loadTimer (timeSlice)
  + Der Registerinhalt wird automatisch in bestimmten Zeitabständen dekrementiert
  + Beim Wert 0 wird eine Zeitscheibenunterbrechung (Timer Interrupt) erzeugt
* Highest-Priority-First (HPF)
  + Jedem Prozess wird eine Priorität zugeordnet
  + Kleinere Zahlenwerte repräsentieren dabei üblicherweise eine höhere Priorität
  + Der Prozess mit der höchsten Priorität erhält den Prozessor
  + Existier verdrängend und nicht verdrängen, mit und ohne Dynamik in der Priorität

**Ein-/Ausgabeoperationen an der Schnittstelle zur Prozessverwaltung**

* Aufgabenstellung
  + Integration der Steuerung von E/A-Geräten in das Betriebssystem
  + Bereitstellung eines stark vereinfachten E/A-Modells
* Randbedingungen
  + Geschwindigkeitsunterschiede zwischen CPU und E/A-Geräten ⇒ Faktor 103 bis 105
  + Parallelarbeit von CPU und E/A-Geräten notwendig
* Lösungskonzept
  + Ein-/Ausgabe wird auf Hardware-Ebene fast immer asynchron ausgeführt
  + E/A-Geräte werden als unabhängige Prozessoren aufgefasst
  + Ablauf:  
    E/A-Gerät durch speziellen E/A-Befehl anstoßen

Eigenständige Operationsausführung des E/A-Gerätes

Meldung des E/A-Endes per Unterbrechung (Interrupt) an CPU

* Realisierungsstruktur
  + Verwendung einer Warteschlange für E/A-Aufträge zur Pufferung der angeforderten E/A-Operationen
  + Sequenzialisierte Abarbeitung der E/A-Aufträge

Das Semaphor-Konzept

Semaphor-Definition

Objekt, auf dem die beiden Operationen p (auch: up) und v (auch: down) existieren

Interne Komponenten eines Semaphors sem:

* Wartezustand für Prozesse (blocked-Queue)
* Wert des Semaphors: sem.ctr
* Interpretation von sem.ctr:
  + positiv: sem.ctr = Anzahl der noch verfügbaren Betriebsmittel
  + negativ: |sem.ctr| = Anzahl der wartenden Prozesse
* p und v sind atomar (Einsatz von Unterbrechungssperren und atomaren Speicheroperationen)

Gegeben sei eine Menge von kritischen Abschnitten, die unter gegenseitigem Ausschluss bearbeitet werden sollen

Verwendung eines Semaphors, das mit 1 initialisiert wird (auch: binäres Semaphor)

Jeder der kritischen Abschnitte wird mit einem Eintritts- und einem Austrittsprotokoll geklammert:

p-Operation: Eintrittsprotokoll

v-Operation: Austrittsprotokoll