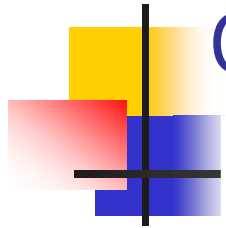


Proseminar KVBK : Scheduler unter Linux



Valderine Kom Kenmegne



Gliederung

- **1. Einführung**
- **2. Einplanungsstrategien im Betriebssystem**
 - 2.1 Ziel der Einplanungsstrategien
 - 2.2 Beispiele von Einplanungsstrategien
- **3. Linux Scheduler**
 - 3.1 Prozessarten in Linux
 - 3.2 Prozesszustände
 - 3.3 Prozessumschaltung
 - **3.4 Praktisches Beispiel: Linux 2.4 Einplanungsstrategie**
 - 3.4.1 Echtzeitprozesse
 - 3.4.2 Normalprozesse
 - 3.4.3 Prozessauswahl
 - 3.4.4 Nachteile
- **4. Zusammenfassung**



1. Einführung

Um alltägliche Aufgaben zu erledigen braucht man einen genauen Zeitplan. Ebenso benötigt das Betriebssystem eine exakte Steuerung. Deshalb beschäftigen wir uns heute mit dem Thema ***Scheduler***.



2. Einplanungsstrategie im Betriebssystem

■ 2.1 Ziel der Einplanungsstrategien

Es gibt verschiedene Ziele die sich manchmal widersprechen:

- Maximale Prozessorausnutzung
- Rasche Bedienung von dringenden Prozessen
- Minimierung der Antwortzeit
- Gerechtigkeit



2.2 Beispiele von Einplanungsstrategien

- *First-Come–First-Served (FCFS)*: „Wer zuerst kommt malt zuerst“ Nicht-unterbrechendes Scheduling.
- *Round Robin*: Jeder Prozess bekommt eine Zeitscheibe. Unterbrechendes Scheduling.
- *Shortest-Process-First*: Prozess mit der kürzesten Bearbeitungsdauer wird zuerst ausgeführt. Man muss die Dauer der Ausführung vorher wissen.



Gliederung

- 1. Einführung
- 2. Einplanungsstrategien im Betriebssystem
 - 2.1 Ziel der Einplanungsstrategien
 - 2.1 Beispiele von Einplanungsstrategien
- ➔ ■ **3. Linux Scheduler**
 - 3.1 Prozessarten in Linux
 - 3.2 Prozesszustände
 - 3.3 Prozessumschaltung
 - **3.4 Praktisches Beispiel: Linux 2.4 Einplanungsstrategie**
 - 3.4.1 Echtzeitprozesse
 - 3.4.2 Normalprozesse
 - 3.4.3 Berechnung der Prozesspriorität
 - 3.4.4 Nachteile
- **4. Zusammenfassung**



3.1 Prozessarten in Linux

- **Interaktivprozesse:** Sie interagieren vielfach mit dem Benutzer z.B. *Texteditoren, Shellbefehle ...*
- **Batchprozesse:** Sie interagieren **nicht** mit dem Benutzer und laufen im Normalfall im Hintergrund z.B. *Suchmaschine, Compiler*
- **Echtzeitprozesse:** Sie brauchen eine schnelle Antwortzeit mit geringer Abweichung. Z.b. *Robotercontroller, Video und Tonübertragung ...*

3.2 Prozesszustände

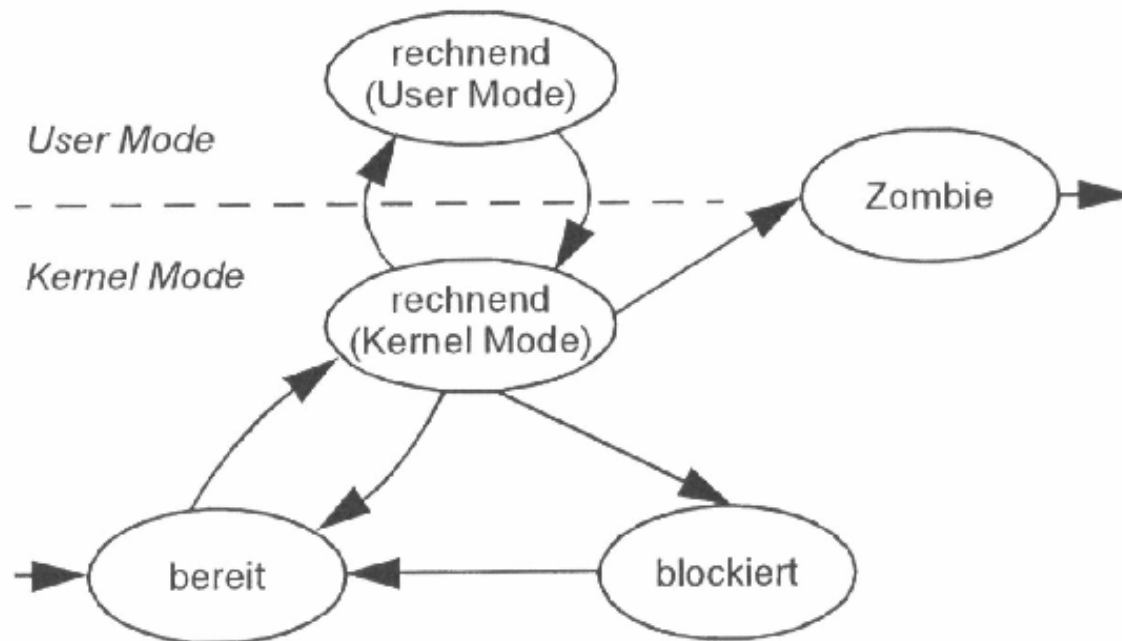
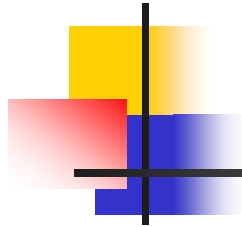


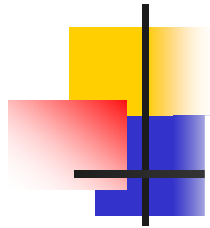
Abb. 4.1: Zustandsdiagramm von Prozessen in UNIX / Linux [Vogt 2001]



3.3 Prozessumschaltung

Ein Prozesswechsel kann erfolgen, wenn eines der folgenden Ereignisse stattfindet:

- Ein neuer Prozess wird erzeugt.
- Prozess beendet.
- Prozess blockiert.
- Die Zeitscheibe ist abgelaufen.



Gliederung

- 1. Einführung
- 2. Einplanungsstrategien im Betriebssystem
 - 2.1 Ziel der Einplanungsstrategien
 - 2.1 Beispiele von Einplanungsstrategien
- 3. Linux Scheduler
 - 3.1 Prozessarten in Linux
 - 3.2 Prozesszustände
 - 3.3 Prozessumschaltung
 - ➔ ▪ **3.4 Praktisches Beispiel: Linux 2.4 Einplanungsstrategie**
 - 3.4.1 Echtzeitprozesse
 - 3.4.2 Normalprozesse
 - 3.4.3 Berechnung der Prozesspriorität
 - 3.4.4 Nachteile
- **4. Zusammenfassung**



3.4 Praktisches Beispiel: Linux Einplanungsstrategie

Hier wird wie in Linux in General, zwischen **Echtzeitprozesse** und **Normalprozesse** unterschiedet und dann eine Einplanungsstrategie gewählt.



3.4.1 Echtzeitprozesse

Jeder Prozess hat eine **statische Priorität** zwischen 1 – 99:

- **SCHED_FIFO**: Wie FCFS. Jedoch wird dieser Prozess verdrängt, wenn es eine höhere Priorität gibt.
- **SCHED_RR**: Wie SCHED_FIFO aber mit Zeitscheiben.



3.4.2 Normalprozesse

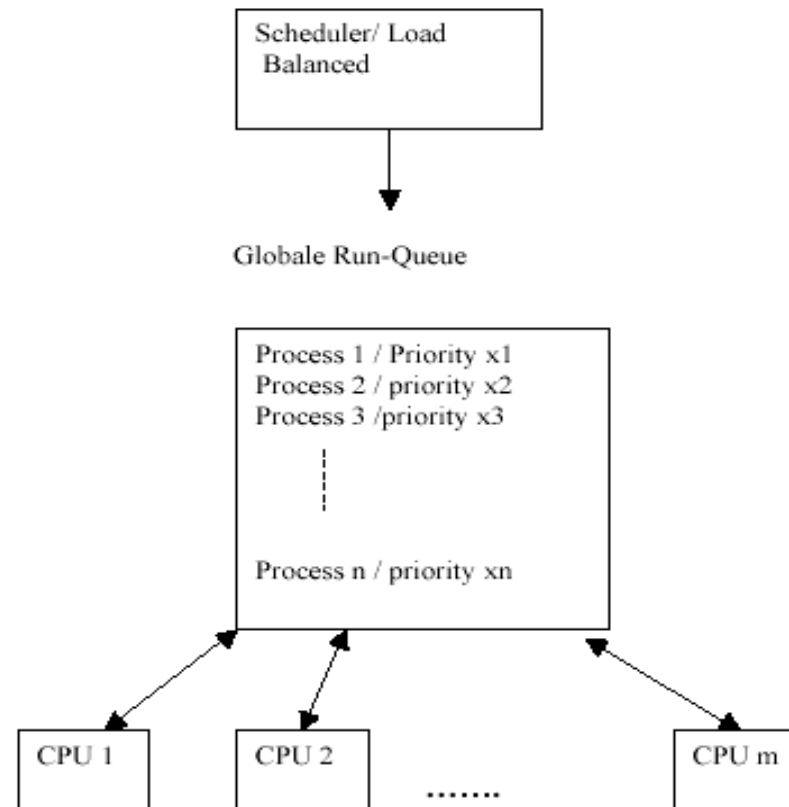
Hier geht es um **Nicht**-Echtzeitprozesse.

Nur eine Einplanungsstrategie:

SCHED_OTHER:

- Dynamische Priorität.
Priorität zwischen $[-20, 19]$ kann durch *nice ()* beeinflusst werden.
- Wird nur ausgeführt, wenn es keine Echtzeitprozesse mehr gibt.

3.4.3 Prozessauswahl



Kernel 2.4



3.4.3 Prozessauswahl

- Der CPU benutzt den „**Goodness**“-Wert (G) um den auszuführenden Prozess auszuwählen. Diese Werte werden dank des Prioritätswerts intern berechnet.

Scheduling-Strategie	verbliebenes Zeitquantum	Goodness
SCHED_FIFO, SCHED_RR	-	1000 + Priorität
SCHED_OTHER	> 0	Quantum + Priorität (+1)
SCHED-OTHER	0	0

„Goodness“- tabelle iX 2/2005



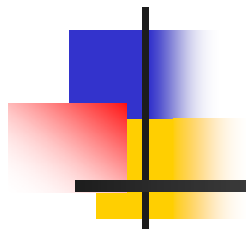
3.4.4 Nachteile von Linux 2.4

- **Großer Aufwand:** Bei neuer Auswahl müssen alle Prozesse in der „Run-Queue“ nach den besten „Goodness“-Werten ausgesucht werden.
- **Verklemmungsgefahr:** Es kann passieren, dass ein CPU wartet bis die Liste freigegeben ist, bevor ein neuer Prozess ausgeführt wird.



4. Zusammenfassung

- **Scheduling** dient dazu die Reihenfolge, in der die Prozesse ausgeführt werden sollen, festzulegen.
- **In Linux** wird zuerst zwischen Echtzeitprozessen und Normalprozessen unterschieden; dann wird eine Einplanungsstrategie gewählt: SCHED_FIFO, SCHED_RR, SCHED_OTHER. Der Scheduler berechnet zum Auswahlverfahren den sogenannten „Goodness“-Wert.



Noch Fragen??