

# Übung zur Vorlesung ”Grundlagen Betriebssysteme und Systemsoftware”

(Prof. Dr. J. Schlichter, WS 2008 / 2009)

Übungsleitung: Dr. Georg Groh (grohg@in.tum.de)

Tutoren: Dipl. Inform. Vivian Prinz (prinzv@in.tum.de), Dr. Nils Kammenhuber (kammenhuber@net.in.tum.de), Dipl. Inform. Robert Schmohl (schmohl@in.tum.de), Dipl. Inform. Dipl. Geogr. Jan Herrmann (hermanj@in.tum.de), Dipl. Inform. Robert Eigner, David Brodski (brodski@in.tum.de), Yang Guo (yang.guo@gmx.de), Jan Finis (finis@in.tum.de), Martin Levihn (levihn@in.tum.de)

<http://www11.in.tum.de/Veranstaltungen/GrundlagenBetriebssystemeundSystemsoftware0809>

<http://www11.in.tum.de/Veranstaltungen/GrundlagenBetriebssystemeundSystemsoftware0809/uebung>

## Blatt 8

- Abgabe: bis 15.12.2008 12:00 Uhr per E-Mail an den Tutor der eigenen Gruppe. Die Mail soll einen Zip-Ordner als attachment haben, der für jede Hausaufgabe einen Unterordner enthält, in dem die Lösung als .txt-File(s), als .c-File(s) o.ä. enthalten ist.
- Musterlösungen Hausaufgaben: ab 15.12.2008 12:00 Uhr auf der Übungswebseite zum Download.
- Musterlösungen Tutoraufgaben: ab 19.12.2008 18:00 Uhr auf der Übungswebseite zum Download.

## Stoff

Es sei empfohlen folgende Literatur durchzuarbeiten:

- Skript Kapitel 4
- Tanenbaum Kapitel 10.3 (Processes In Unix), Kapitel 11.4 (Processes and Threads in Windows)

## 1 Hausaufgabe (Parallelisierung mit Java Threads)

### Lernziele

In Kapitel 4 der Vorlesung geht es um Aspekte der Erzeugung von Prozessen und Threads. Die Aufgabe soll motivieren, warum man überhaupt Parallelisierung einsetzt und inwieweit man von Parallelisierung profitieren kann.

## Aufgabe

Die Java-Klassen im auf der Webseite zur Verfügung gestellten File berechnen  $a*b+c*d$  für alle Parameterwerte  $1 \leq a, b, c, d \leq 20$  (160000 Parameterkombinationen). Dabei initialisiert ein Objekt der Klasse `Init` zuerst einen Puffer `startWerte` mit allen möglichen Parameterkombinationen. Anschliessend berechnet ein Objekt der Klasse `Mult` die erste Multiplikation. Dabei liest sie als Eingabe der Reihe nach Einträge aus dem Puffer `startWerte` und schreibt die Ergebnisse in einen Puffer `puffer1`. Ein zweites Objekt der Klasse `Mult` berechnet die zweite Multiplikation und schreibt die Ergebnisse in einen Puffer `puffer2`. Anschliessend nimmt ein Objekt der Klasse `Sum` die beiden Puffer `puffer1` und `puffer2` als Eingabe, summiert deren Einträge und schreibt die Ergebnisse in einen Puffer `ergebnis`. Als Puffer dienen Java `Vector`-Objekte, deren Methoden `add()` und `elementAt()` `synchronized` sind. Die Anzahl der Einträge eines Vectors lässt sich mit `size()` ermitteln.

### 1.1 Teilaufgabe

Die gegebenen Java-Klassen führen obige Berechnung durch (Import-Anweisungen wurden weglassen, Zeilennummern gehören nicht zum Programm). Parallelisieren Sie die Berechnung mit Hilfe von Threads möglichst weitgehend. Jeder Thread kontrolliere, ob in seinem Eingabepuffer neue Daten vorhanden sind. Ist dem nicht so, gibt der Thread den Prozessor frei. Ansonsten führt er die Berechnung für die neuen Werte durch. Ein Thread wird beendet, wenn er alle 160000 Berechnungen durchgeführt hat.

### 1.2 Teilaufgabe

Wieso bringt das Parallelisieren mit Threads für einen Einprozessorrechner nur bedingt Vorteile?

## Abgabe

Der geänderte Code als `.java` Dateien. Die Beantwortung der Frage als `.txt` oder PDF File.

## 2 Hausaufgabe (Dispatcher)

### Lernziele

Wiederholen von Aspekten des Dispatchers.

### Aufgabe

Für die Realisierung von Prozessen auf realen Prozessoren sind Verwaltungsoperationen auszuführen, die die Zustandsübergänge zwischen *real rechnend* und *rechenbereit* der Prozesse durchführen. Die entsprechenden Operationen (Programme), die vom Dispatcher definiert werden, sind das *Binden* eines Prozesses an einen Prozessor und das *Lösen* der Bindung.

## 2.1 Teilaufgabe

Was ist ein *Prozesskontrollblock* (PCB)?

## 2.2 Teilaufgabe

Welche weiteren Informationen werden für einen Prozesswechsel benötigt?

## 2.3 Teilaufgabe

Zeigen Sie an Hand des UNIX Befehls *sleep()*, wie für den Dispatcher das Binden und Lösen eines Prozesses durchzuführen ist!

## 2.4 Teilaufgabe

Worin liegt der Unterschied zwischen dem Dispatchen von Prozessen und dem Dispatchen von Threads? Gehen Sie dabei insbesondere auf Benutzer-Level und Kernel Threads ein.

## Abgabe

Antworten als .txt oder PDF Datei.

# 3 Hausaufgabe (Prozess-Verwaltung in Unix)

## Lernziele

Kennenlernen der konkreten Umsetzung von Prozess-Verwaltung in Unix

## Aufgabe

Studieren Sie Kapitel 10.3 (Processes In Unix) und beantworten Sie folgende Fragen!

### 3.1 Teilaufgabe

Sollten Daemons eine höhere oder eine niedrigere Priorität haben als interaktive Prozesse?

### 3.2 Teilaufgabe

Wenn ein neuer Prozess mit *fork* erzeugt wird muss ihm ein eindeutiger Integer-Wert als PID zugewiesen werden? Ist es ausreichend, im Kernel einen Zähler zu haben, der bei jeder Prozess-Erzeugung inkrementiert wird und diesen als neue PID zu nutzen?

### 3.3 Teilaufgabe

In der Prozess Tabelle wird im Eintrag jedes Prozesses die PID des Eltern-Prozesses gespeichert. Warum?

### 3.4 Teilaufgabe

Welche Kombination der sharing\_flag Bits des Linux clone Kommandos korrespondiert zu einem gewöhnlichen fork Kommando in Unix?

### Abgabe

Antworten als .txt oder PDF Datei

## 4 Tutoraufgabe (Prozess-Verwaltung in Windows)

### Lernziele

Kennenlernen der konkreten Umsetzung von Prozess-Verwaltung in Windows

### Aufgabe

Sie haben ja (hoffentlich) bereits Kapitel 11.4 (Processes and Threads in Windows) studiert. Beantworten Sie folgende Fragen!

#### 4.1 Teilaufgabe

Nennen Sie 3 Gründe warum ein Prozess beendet werden kann!

#### 4.2 Teilaufgabe

Nehmen Sie folgende Situation (Abbildung 1) an, in der das System gerade einen Thread schedulen möchte. Unter der Annahme, dass jeder Thread beschränkte Rechenzeit benötigt: Wann wird ein Thread mit Priorität 3 auf Windows 2000 laufen können?

#### 4.3 Teilaufgabe

Nehmen Sie an, dass das Quantum auf 20 ms gesetzt sei und dass der momentane Thread (Priorität 24) gerade ein Quantum begonnen hat. Plötzlich wird eine I/O Operation fertig und ein Thread mit Priorität 28 wird in den Zustand ready versetzt. Wie lange wird es ungefähr dauern bis er dran ist?

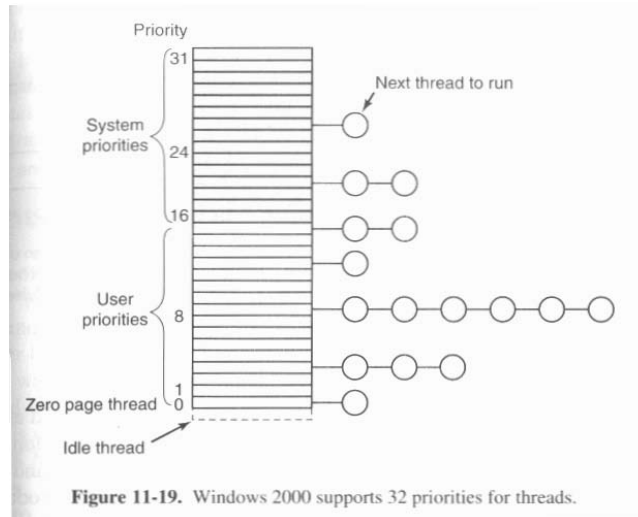


Figure 1: Abbildung aus Tanenbaum

#### 4.4 Teilaufgabe

In Windows 2000 ist die momentane Priorität immer größer oder gleich der Base-Priority. Sind Umstände denkbar, in denen die momentane Priorität als kleiner als die Base-Priority gesetzt werden könnte? Geben Sie ggf ein Beispiel an!

#### Abgabe

Die Aufgabe soll in den Tutorübungen erarbeitet werden und soll nicht abgegeben werden