Ludwig-Maximilians-Universität München
Institut für Informatik
Lehrstuhl für Mobile und Verteilte Systeme
Prof. Dr. Claudia Linnhoff-Popien

mobile and distributed systems group

Betriebssysteme im Wintersemester 2015/2016
Übungsblatt 7

Abgabetermin: 07.12.2015, 16:00 Uhr

Besprechung: Besprechung der T-Aufgaben in den Tutorien vom 30. November – 04. Dezember
Besprechung der H-Aufgaben in den Tutorien vom 07. – 11. Dezember

Aufgabe 30: (T) User-Level-, Kernel-Level-Threads und Prozesse

(– Pkt.)

Betrachten Sie nun folgendes Fallbeispiel: Gegeben ist eine Anwendung, die bisher single-threaded abläuft, also nur aus einem einzigen Thread besteht. Diese Anwendung bietet eine Eingabe-Schnittstelle, über die der Benutzer mathematische Ausdrücke auswerten lassen kann. Die Anwendung soll nun so modifiziert werden, dass der Benutzer während einer laufenden Auswertung eines Ausdrucks weitere Berechnungen starten oder auch Zwischenergebnisse einer der aktiven Auswertung abfragen kann.

Im Folgenden wird angenommen, dass ein Benutzer die Berechnung der ersten hunderttausend Primzahlen anfordert.

a. Warum muss man bei diesem Szenario unbedingt einen neuen Thread oder Prozess für die Berechnung starten?

b. Es werden folgende Alternativen erwogen:

   - Berechnung der Primzahlen in einem neuen User-Level-Thread starten.
   - Berechnung der Primzahlen in einem neuen Kernel-Level-Thread starten.
   - Berechnung der Primzahlen in einem neuen Prozess starten.

   Bewerten Sie jede der drei Möglichkeiten hinsichtlich der folgenden Gesichtspunkte. Verwenden Sie dazu eine geeignete tabellarische Darstellung.

   (i) Aufwand für die Generierung des neuen Threads/Prozesses

   (ii) Kommunikation/Datenaustausch zwischen der bisherigen Anwendung und dem neuen Thread/Prozess

   (iii) Abwicklung des Schedulings (Wer ist für das Scheduling des neuen Threads/Prozesses verantwortlich?)

   (iv) Ausführung auf einer Multiprozessorumgebung

c. Basierend auf Ihrer Bewertung in der Teilaufgabe b), welche Möglichkeit würden Sie hier wählen? Begründen Sie Ihre Entscheidung.

Aufgabe 31: (T) Threads in Java

(– Pkt.)

a. Betrachten Sie das folgende Java-Programm

```
1  public class SimpleThread extends Thread {
2
3          String msg;
4          int cycles;
5
6          SimpleThread(String m, int c) {
7                  msg = m;
8                  cycles = c;
9          }
10
11         // Overrides run() in Thread class to define object's
12         // behavior.
13         public void run() {
14                 for (int i = 0; i < cycles; i++) {
15                         System.out.println(msg + " cycle " + i);
16                 }
17         }
18
19         // Command-line argument is the number of cycles c
20         // which must be converted from String to int.
21         // Builds and starts two threads of type SimpleThread.
22         // Continues for c cycles.
23
24         public static void main(String[] args) {
25
26                 if (args.length < 1) {
27                         System.out.println("Arguments are:");
28                         System.out.println("  cycles");
29                         System.exit(-1);
30                 }
31
32                 int c = Integer.parseInt(args[0]);
33
34                 SimpleThread t1 = new SimpleThread("Thread 1", c);
35                 SimpleThread t2 = new SimpleThread("Thread 2", c);
36
37                 t1.start();
38                 t2.start();
39         }
40 }
```

Als Eingabeparameter verlangt es eine Integer-Zahl. Wie hängt diese Zahl mit der Ausgabe zusammen? Welche Art der Ausgabe erwarten Sie für verschiedene Integer-Eingaben (zum Beispiel 1, 2, 100 oder 10000)?

b. Basierend auf dem Java-Programm aus Teilaufgabe a:
Geben Sie in Abhängigkeit von `c` eine allgemeine Formel für die Anzahl der möglichen Konsolenausgaben an.

Aufgabe 32: (H) Nicht-preemptives Scheduling

(13 Pkt.)

In dieser Aufgabe sollen zwei Scheduling-Strategien untersucht werden: die nicht-preemptive Strategie FCFS (First Come First Served) und die nicht-preemptive Strategie SJF (Shortest Job First). Dazu seien die folgenden Prozesse mit ihren Ankunftszeitpunkten und Bedienzeiten (in beliebigen Zeiteinheiten) gegeben.

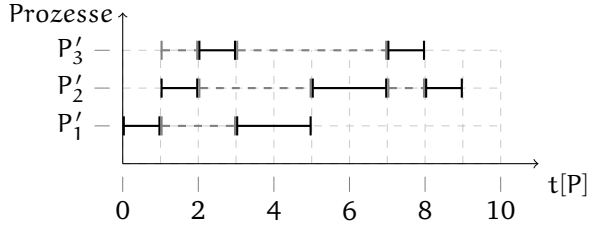
Prozess	Ankunftszeitpunkt	Bedienzeit
P_1	0	2
P_2	2	4
P_3	2	7
P_4	4	3
P_5	5	2

- Trifft ein Prozess zum Zeitpunkt t ein, so wird er direkt zum Zeitpunkt t berücksichtigt.
- Wird ein Prozess zum Zeitpunkt t' unterbrochen, so reiht er sich auch zum Zeitpunkt t' wieder in die Warteschlange ein.
- Sind zwei Prozesse absolut identisch bezüglich ihrer relevanten Werte, so werden die Prozesse nach aufsteigender Prozess-ID in der Warteschlange eingereiht (Prozess P_i vor Prozess P_{i+1}, usw.). Diese Annahme gilt sowohl für neu im System eintreffende Prozesse, als auch für den Prozess, dem der Prozessor u.U. gerade entzogen wird!
- Jeder Prozess nutzt sein Zeitquantum stets vollständig aus d.h. kein Prozess gibt den Prozessor freiwillig frei (Ausnahme: bei Prozessende).

Beispiel: Es seien folgende Ankunfts– und Bedienzeiten für die drei Beispielprozesse P'_1, P'_2 und P'_3 gegeben:

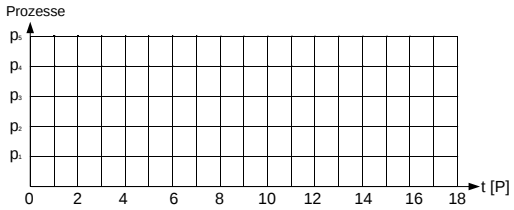
Prozess	Ankunftszeitpunkt	Bedienzeit
P'_1	0	3
P'_2	1	4
P'_3	1	2

Das folgende Diagramm veranschaulicht ein beliebiges Scheduling der drei Prozesse P'_1, P'_2 und P'_3:



Bearbeiten Sie unter den gegebenen Voraussetzungen nun die folgenden Aufgaben:

a. Verwenden Sie nun die **nicht-präemptive Strategie FCFS** und erstellen Sie entsprechend dem vorherigen Beispiel ein Diagramm, das für die Prozesse P_1–P_5 angibt, wann welchem Prozess Rechenzeit zugeteilt wird und wann die Prozesse jeweils terminieren. Kennzeichnen Sie zudem für jeden Prozess seine Ankunftszeit. Erstellen Sie Ihre Lösung basierend auf folgender Vorlage:



b. Verwenden Sie nun die **nicht-präemptive Strategie SJF** und stellen Sie Ihre Lösung, wie in der vorherigen Teilaufgabe a), dar.

c. Berechnen Sie als Dezimalzahl mit einer Nachkommastelle die mittlere Verweil- und Wartezeit für die zwei Verfahren FCFS und SJF.

d. Welchen Nachteil hat SJF in Bezug auf die Verweildauer von Prozessen?

Aufgabe 33: (H) Einfachauswahlaufgabe: Prozesse und Threads

(5 Pkt.)

Für jede der folgenden Fragen ist eine korrekte Antwort auszuwählen („1 aus n“). Eine korrekte Antwort ergibt jeweils einen Punkt. Mehrfache Antworten oder eine falsche Antwort werden mit 0 Punkten bewertet.

<table>
<tr><td colspan="4">a) Welcher Zustand wird im 9-Zustands-Modell im Vergleich zum 7-Zustands-Modell aufgespalten, um zu unterscheiden, ob sich der Prozess im Kernel oder User Mode befindet?</td></tr>
<tr><td>(i) new</td><td>(ii) ready</td><td>(iii) blocked</td><td>(iv) running</td></tr>
<tr><td colspan="4">b) Welcher Fall wird beim Zusammenspiel von Threads mit Prozessen nicht unterschieden?</td></tr>
<tr><td>(i) ein Prozess, kein Thread</td><td>(ii) ein Prozess, ein Thread</td><td>(iii) mehrere Prozesse, ein Thread</td><td>(iv) mehrere Prozesse, mehrere Threads</td></tr>
<tr><td colspan="4">c) Welche Aussage bezüglich des Thread-Konzepts ist falsch?</td></tr>
<tr><td colspan="4">(i) Zur Generierung eines neuen Threads in einem existierenden Prozess ist wesentlich weniger Zeit notwendig, als zur Generierung eines neuen Prozesses.
(ii) Kontextwechsel unter Threads innerhalb eines Prozesses erfordern weniger Zeit als Kontextwechsel von Prozessen.
(iii) Durch den getrennten Adressraum der Threads eines Prozesses sind die Daten einzelner Threads bezüglich anderer Threads sicher.
(iv) Terminiert der übergeordnete Prozess, so terminieren mit ihm alle Threads.</td></tr>
<tr><td colspan="4">d) Welcher der folgenden Zustände wird nicht explizit für die Zustandsmodellierung von Threads verwendet?</td></tr>
<tr><td>(i) ready</td><td>(ii) suspended</td><td>(iii) blocked</td><td>(iv) running</td></tr>
<tr><td colspan="4">e) Was ist keine Funktion, für die eine Thread-Bibliothek für User-Level-Threads Code bereitstellen muss?</td></tr>
<tr><td>(i) Generierung und Terminierung von Threads</td><td>(ii) Nachrichten- und Datenübermittlung zwischen Threads</td><td>(iii) Zuordnung mehrerer Threads eines Prozesses auf verschiedene Prozessoren</td><td>(iv) Sichern und Löschen von Threadkontexten</td></tr>
</table>