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 2018/2019
Übungsblatt 9

Abgabetermin: 07.01.2019, 18:00 Uhr

Besprechung: Besprechung der T-Aufgaben in den Tutorien vom 17. – 21. Dezember 2018
Besprechung der H-Aufgaben in den Tutorien vom 07. – 11. Januar 2019

Aufgabe 40: (T) IPC mit Java Pipes

(– Pkt.)

In der Vorlesung haben Sie das Konzept von (unnamed) Pipes zur Interprozesskommunikation kennengelernt. Pipes finden typischerweise bei Erzeuger/Verbraucher-Problem Anwendung. Folgendes Erzeuger/Verbraucher-Problem sei gegeben:
Ein Erzeuger erzeugt kontinuierlich Zufallszahlen und schreibt diese in eine Pipe. Ein Verbraucher liest diese Zufallszahlen aus der Pipe.

a. Nennen Sie einen Vorteil und einen Nachteil, den die Verwednung von Pipes gegenüber der Verwendung des IPC-Mechanismus Shared Memory hat.

b. Java stellt eine API zur Verfügung, mit der sich das Erzeuger/Verbraucher-Problem mit Hilfe von Pipes implementieren lässt:
Es existieren z.B. die Klassen `PipedInputStream` und `PipedOutputStream` zum Schreiben von Bytes bzw. `PipedReader` und `PipedWriter` zum Schreiben von Unicode Zeichen. Schreiben Sie ein Java-Programm, in dem der Erzeuger Zufallszahlen erzeugt, auf der Konsole ausgibt und anschließend in eine Pipe schreibt. Der Verbraucher soll diese Zahlen aus der Pipe lesen und auf der Konsole ausgeben. Verwenden Sie zur Implementierung des Erzeugers und Verbrauchers zwei Threads. Sorgen Sie zudem dafür, dass sich die beiden Threads zwischen ihren Operationen für eine zufällige Zeitspanne schlafen Legen. Schreiben Sie Ihr Programm so, dass man die Kapazität der Pipe beim Erzeugen dieser mit angeben kann (z.B. 1-, 2- oder 5-elementiger Puffer), um die positiven Effekte dieser zu erkennen.

c. Erweiteren Sie nun das gerade geschriebene Programm durch einen Filter, der die durch den Erzeuger gelieferten Nummern erhält und dem Verbraucher den jeweils aktuellen Durchschnitt aller bereits erzeugten Nummern liefert. Schreiben Sie den Filter wieder so, dass man die Kapazität des Filters beim Erzeugen mit angeben kann.

Aufgabe 41: (T) SJF versus SRPT

(– Pkt.)

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

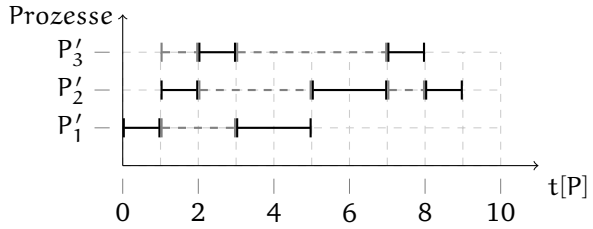
Prozess	Ankunftszeitpunkt	Bedienzeit
P_1	0	6
P_2	2	4
P_3	2	2
P_4	4	1
P_5	8	7
P_6	9	3

- 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:

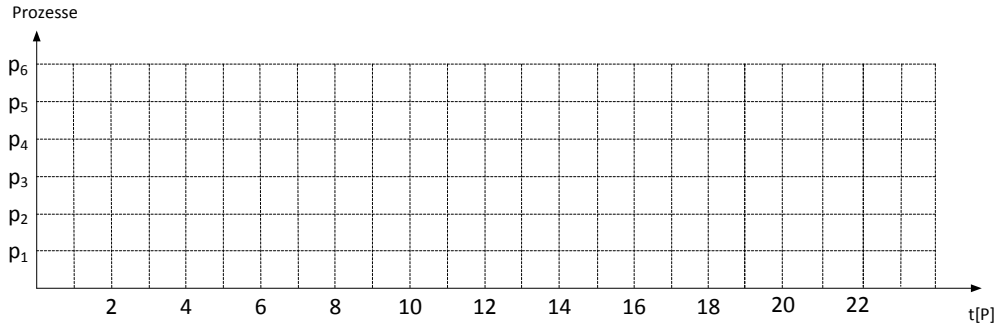
Prozess	Ankunftszeitpunkt	Bedienzeit
P'_1	0	3
P'_2	1	4
P'_3	1	2

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 SJF** und erstellen Sie entsprechend dem vorherigen Beispiel ein Diagramm, das für die Prozesse P_1–P_6 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 **preemptive Strategie SRPT** 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 SJF und SRPT.

d. Wodurch lässt sich der Unterschied zwischen SJF und SRPT in Bezug auf die mittlere Wartezeit begründen?

Aufgabe 42: (H) Petrinetze und Erreichbarkeitsgraphen

(14 Pkt.)

Gegeben sei ein Rechner mit einer CPU und zwei Prozessen P_1 und P_2, die mit preemptivem Multitasking auf der CPU ausgeführt werden. Wird zwischen den Prozessen gewechselt, so wird der aktive Prozess supendiert und das Betriebssystem *BS* aktiv. Dieses führt mittels Scheduler und Dispatcher den Wechsel zwischen den Prozessen durch. Der ausgewählte Prozess wird der CPU durch das Dispatchen zum Rechnen zugewiesen. Gehen Sie zunächst von folgenden Bedingungen aus:

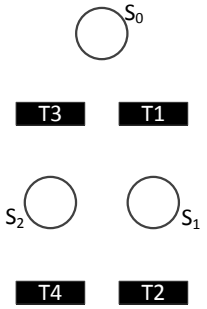
– Auf der CPU kann zu jedem Zeitpunkt immer nur genau ein Prozess gleichzeitig rechnen.

– Die Betriebssystemfunktionen sind ebenfalls als eigener Prozess zu beachten.

– Zu Beginn ist das Betriebssystem aktiv.

Bearbeiten Sie dazu die folgenden Teilaufgaben:

a. Im Folgenden soll das beschriebene System mit einem Petrinetz modelliert werden. Die Markierung der Stellen kann dabei beschreiben, welcher der Prozesse aktuell der CPU zugeordnet ist. Ergänzen Sie nun das nachfolgend angedeutete Petrinetz um eine minimale Anzahl an weiteren Marken, Stellen, Transitionen und Kanten (insofern erforderlich), so dass die oben genannten Bedinungen erfüllt sind. Erklären Sie für alle Stellen und Transitionen jeweils kurz deren Bedeutung.



b. Erweitern Sie nun das in Aufgabenteil a) angedeutete Petrinetz um eine minimale Anzahl an weiteren Marken, Stellen, Transitionen und Kanten (insofern erforderlich), so dass das Verhalten von Round-Robin-Scheduling modelliert wird. Hierbei soll P_1 der erste Prozess sein, welcher der CPU vom Betriebssystem zum Rechnen zugewiesen wird. Geben Sie allen neu eingefügten Transitionen oder Stellen einen eindeutigen Bezeichner und erklären Sie kurz deren Bedeutung.

c. Geben Sie den Erreichbarkeitsgraphen zu dem Petrinetz aus Teilaufgabe b) an. Geben Sie hierbei auch an, wie in den Markierungen des Erreichbarkeitsgraphen die Stellen angeordnet sind. Beschriften Sie alle Übergänge zwischen Markierungen mit der Bezeichnung der Transition, die hierfür feuern muss.

d. Kann einer der beiden Prozesse P_1 oder P_2 verhungern? Begründen Sie ihre Aussage mithilfe des Erreichbarkeitsgraphen zu dem Petrinetz aus Teilaufgabe b).

e. Begründen Sie mithilfe des Erreichbarkeitsgraphen zu dem Petrinetz aus Teilaufgabe b), ob das System frei von Verklemmungen bzw. teilweisen Verklemmungen ist.

Aufgabe 43: (H) Einfachauswahlaufgabe: Prozesskoordination

(5 Pkt.)

Für jede der folgenden Fragen ist eine korrekte Antwort auszuwählen („1 aus n“). Nennen Sie dazu in Ihrer Abgabe explizit die jeweils ausgewählte Antwortnummer ((i), (ii), (iii) oder (iv)). Eine korrekte Antwort ergibt jeweils einen Punkt. Mehrfache Antworten oder eine falsche Antwort werden mit 0 Punkten bewertet.

<table>
<tr><td colspan="4">a) Welche Art der Interprozesskommunikation kann ausschließlich zwischen Prozessen erfolgen, die eine Eltern- Kind-Beziehung bzw. einen gemeinsamen Vorfahren in der Prozesshierarchie besitzen?</td></tr>
<tr><td>(i) (Unnamed) Pipes</td><td>(ii) Semaphore</td><td>(iii) Shared Memory</td><td>(iv) Message-Queues</td></tr>
<tr><td colspan="4">b) Wie bezeichnet man einen Speicher, der in Form eines eindimensionalen Arrays unter Verwendung der Modulo-Funktion realisiert wird?</td></tr>
<tr><td>(i) Sparpuffer</td><td>(ii) Linearpuffer</td><td>(iii) Wechselpuffer</td><td>(iv) Ringpuffer</td></tr>
<tr><td colspan="4">c) Wie bezeichnet man die Phase, in der sich zu einem Zeitpunkt nur ein Prozess befinden darf, da sich sonst z.B. inkonsistente Zustände bei gemeinsam genutzten Datenstrukturen ergeben?</td></tr>
<tr><td>(i) einfacher Bereich</td><td>(ii) schwieriger Bereich</td><td>(iii) kritischer Bereich</td><td>(iv) unkritischer Bereich</td></tr>
<tr><td colspan="4">d) Wie berechnet sich allgemein die Verweildauer?</td></tr>
<tr><td>(i) Verweildauer = Bedienzeit + Wartezeit</td><td>(ii) Verweildauer = Beendigungszeit + Wartezeit</td><td>(iii) Verweildauer = Startzeit + Wartezeit</td><td>(iv) Verweildauer = Ankunftszeit + Wartezeit</td></tr>
<tr><td colspan="4">e) Wie bezeichnet man den preemptiven Scheduling-Algorithmus, bei welchem Prozesse in periodischen Intervallen in ihrer Abarbeitung unterbrochen werden, bis sie wieder ein gewisses Zeitquantum für die Abarbeitung erhalten?</td></tr>
<tr><td>(i) First Come First Served</td><td>(ii) Round Robin</td><td>(iii) Shortest Remaining Processing Time</td><td>(iv) Shortest Job First</td></tr>
</table>