Universität Heidelberg
Sommersemester 2019

Lehrstuhl für Parallele und Verteilte Systeme (PVS)
Diego Costa, Artur Andrzejak

# 8. Übung zur Vorlesung „Betriebssysteme und Netzwerke“ (IBN)

Abgabedatum: 18.06.2019, 11:00 Uhr

---

## Aufgabe 1 (3 Punkte)

Die vier Bedingungen für ein Ressourcen-Deadlock sind notwendig aber nicht hinreichend dafür, dass ein Deadlock entsteht.

**a)** Nehmen Sie an, es gibt eine Ressource mit vier Instanzen sowie drei Prozesse. Jeder der Prozesse hat maximal Bedarf an zwei Ressourcen. Zeigen Sie, dass es in diesem Szenario nie zu einem Deadlock kommen kann.

**b)** Geben Sie ein Beispiel an, in dem die notwendigen Bedingungen erfüllt sind, aber dennoch kein Deadlock vorliegt. In Ihrem Beispiel sollen von jeder Ressource höchstens zwei Instanzen vorhanden sein.

## Aufgabe 2 (1 Punkt)

Betrachten Sie das unten abgebildete Szenario eines Verkehrsdeadlocks.

1. Erläutern Sie, warum alle vier Bedingungen für ein Ressourcen-Deadlock erfüllt sind.
2. Geben Sie eine einfache Regel für die Vermeidung von Deadlocks in dieser Situation.


## Aufgabe 3 (3 Punkte)

Ein System habe folgende (ununterbrechbare) Betriebsmittel: zwei Drucker, drei Netzwerkkarten und eine Grafikkarte. Aktuell laufen die drei Prozesse P1, P2 und P3. Prozess P1 belege aktuell

eine Netzwerkkarte und frage einen Drucker an. P2 belege einen Drucker und eine Netzwerkkarte und frage eine weitere Netzwerkkarte sowie die Grafikkarte an. P3 belege den anderen Drucker und eine Netzwerkkarte. Er frage auch die Grafikkarte an.

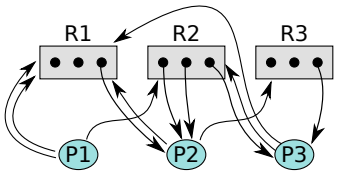
**a)** Zeichnen Sie den Ressourcenbelegungsgraphen.

**b)** Geben Sie Ressourcenvektor, Ressourcenrestvektor, Belegungsmatrix und Anforderungsmatrix an.

**c)** Führen Sie den Bankieralgorithmus aus, um zu entscheiden, ob ein Deadlock vorliegt.

## Aufgabe 4 (3 Punkte)

Der nachfolgende Ressourcenbelegungsgraph zeigt die aktuellen Belegungen und Anforderungen der Prozesse P1, P2 und P3. Die Prozesse können erst dann weiter ausgeführt werden, wenn die eingezeichneten Anforderungen vollständig erfüllt sind.




**a.** Finden Sie mithilfe des Bankieralgorithmus eine sichere Ausführungssequenz der Prozesse.

**b.** Trotz des Ergebnisses in Aufgabenteil a) könnten die obigen Anforderungen zu einem Deadlock führen, falls das Betriebssystem keine Methoden zur Vermeidung dieser implementiert hat. Mit welcher Ausführungsreihenfolge ist dies der Fall? Warum ist dies kein Widerspruch zu dem Ergebnis des Bankieralgorithmus?

## Aufgabe 5 (6 Punkte)

Schreiben Sie ein Programm, um den Deadlock-Erkennungsalgorithmus mit mehreren Ressourcen von jedem Typ zu implementieren. Ihr Programm soll aus einer Datei die folgenden Eingaben einlesen: die Anzahl der Prozesse, die Anzahl der Ressourcenarten, die Anzahl der verfügbaren Ressourcen jedes Typs (Vektor E), die aktuelle Belegungsmatrix C (erste Zeile, zweite Zeile usw.) und die Anforderungsmatrix R (analog wie bei C). Das Programm soll nach jedem Markieren eines Prozesses den Index des Prozesses und den aktualisierten Vektor A ausgeben. Falls ein Deadlock festgestellt wird, sollen die Indizes aller beteiligten Prozesse ausgegeben werden. Testen Sie Ihr Programm an den Beispielen aus Vorlesung 15, sowie an einem eigenen Beispiel, bei dem es zu einem Deadlock kommt. Geben Sie als Teil der Lösung (neben dem Quelltext) die Eingabe und Ausgabe Ihres Programms für die beiden Testfälle an. Als Implementationssprache können Sie C, C++, oder Java benutzen. Fragen Sie bitte bei Ihrem/r Tutor/in nach, falls Sie eine andere Progammiersprache benutzen wollen.

**Aufgabe 6** **(Bonus, 4 Punkte)**

Im Juni 2003 startete NASA die Raumsonde Spirit, um den Mars zu erkunden. 17 (Erden-)Tage nach seiner Ankunft, am 21. Januar 2004, ergab sich eine Anomalie, die den Rover veranlasste, ständig neuzustarten. Das machte die Experimente, die der Rover durchführen sollte, unmöglich. Es erschwerte zudem die Kommunikation mit der Erde und verhinderte außerdem, dass das solarbetriebene Gefährt zur Marsnacht herunterfuhr, um Strom zu sparen.

Die Ursache war eine Kette von mehreren Softwarefehlern.

**a)** Entnehmen Sie im Bericht *The Mars Rover Spirit FLASH Anomaly*[1] dem Absatz *DOS Library Design Flaw* den Anfang dieser Kette und beschreiben Sie diese Ursache mit eigenen Worten.

**b)** Beschreiben Sie außerdem, wie ein Deadlock zur Propagierung des Fehlers beitrug (Absatz *Unfortunate Side Effects*).

Vor zwei Jahren startete eine Testmission von LightSail. Das Projekt erforscht die Technik, Raumsonden durch Sonnensegel anzutreiben. Nur zwei Tage nach Start kam es zu einem unerwarteten Softwarefehler.[2] Als die Log-Datei der *Beacons*, die die Sonde regelmäßig senden sollte, die Größe von 32 MB überschritt, stürzte das Linux-basierte Betriebssystem ab. Mangels Radio-Kommunikation konnte man das System auch nicht neustarten.

**c)** Lesen Sie die Wikipediaseite über das Konzept eines *Watchdogs*[3], das Probleme wie bei Spirit und LightSail lösen kann. In der Elektrotechnik-Masterarbeit *Fault Tolerant and Flexible CubeSat Software Architecture*[4] wird eine Softwarearchtiktur für die Familie der kompakten CubeSat-Raumsonden, die der von LightSail ähnelt, beschrieben. Lesen Sie die Absätze 5.2.4 und 5.2.5 und beschreiben Sie die Arbeitsweise des Software-Watchdogs und die Statusverifikation am Beispiel des Prozesses, der für das Beacon-Signal verantwortlich ist.

Es gibt verschiedene Spekulationen, warum kein Watchdog-Mechanismus bei LightSail gegriffen hat. Der glückliche Einschlag von kosmischer Strahlung hat jedenfalls nach 8 Tagen dazu geführt, dass LightSail wieder neustartete.[5] Auch Spirit hat nach seinen Startschwierigkeiten hervorragend funktioniert. Statt der im Projekt veranschlagten Dauer von 3 Monaten auf dem Mars hielt der Rover mehr als 5 Jahre. Sein Zwilling Opportunity, der fast zeitgleich auf der anderen Seite des Mars landete, ist heute noch aktiv.

[1] http://ieeexplore.ieee.org/abstract/document/1559723/
[2] http://www.planetary.org/blogs/jason-davis/2015/20150526-software-glitch-pauses-ls-test.html
[3] http://de.wikipedia.org/wiki/Watchdog
[4] http://digitalcommons.calpoly.edu/theses/550/
[5] http://www.planetary.org/blogs/jason-davis/2015/20150530-lightsail-phones-home.html