VO 182.022 6. Mai 2011

Prüfung Betriebssysteme

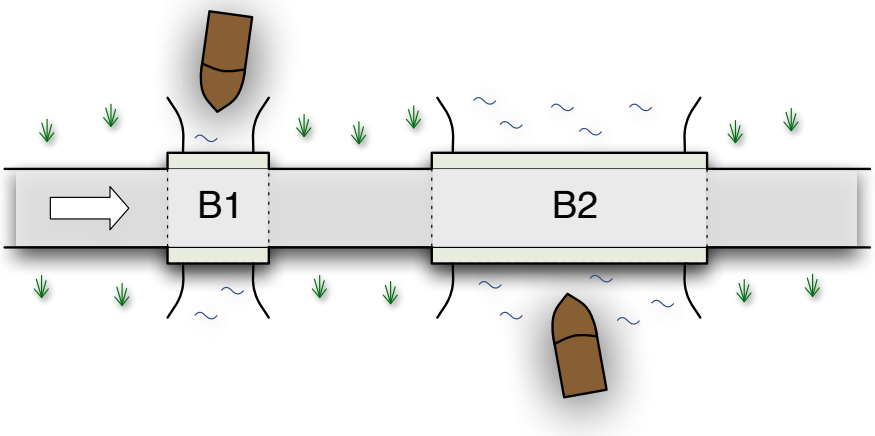
KNr. MNr. Zuname, Vorname

Ges.)(100)	1.)(30)	2.)(25)	3.)(45)	Zusatzblätter:

Bitte verwenden Sie nur dokumentenechtes Schreibmaterial!

1 Synchronisation mit Semphoren (30)

In einem System aus parallelen Prozessen soll die Funktion einer Hebebrückenanlage (siehe Skizze) simuliert werden. Eine Einbahnstraße für Autos kreuzt zwei Schiffahrtswege, wobei die Kreuzungen als Hebebrücken (B1 bzw. B2) ausgeführt sind.



Die unten angegebenen Prozesstemplates sind so mit Semaphoroperationen zu ergänzen, dass die parallele Exekution der Prozesse folgende Eigenschaften erfüllt:

- Im Prozesssystem wird jedes Auto und jedes Schiff durch einen Prozess realisiert. Es gibt drei verschiedene Prozesstemplates: A (für Autos), S1 (für Schiffe, die B1 passieren), und S2 (für Schiffe, die B2 passieren).
- Das System unterstützt beliebig viele aktive Kopien von A, S1 und S2.
- Auf keiner Kreuzung dürfen sich gleichzeitig Autos und Schiffe befinden.
- Für die Kreuzungen B1 und B2 gilt, dass sich zu jedem Zeitpunkt maximal ein Schiff auf der Kreuzung befinden darf.
- Die maximale Anzahl von Autos, die sich gleichzeitig auf den Brücken befinden dürfen, ist 1 für B1 und k für B2.

- Der Schiffsverkehr hat Vorrang gegenüber dem Autoverkehr, d.h., Schiffe dürfen nicht durch Autowarteschlangen, die sich vor den Brücken bilden, verzögert werden.
- Autos, die auf das Passieren von B2 warten, halten bereits vor B1, um B1 nicht zu blockieren (d.h., das Straßenstück zwischen B1 und B2 bleibt leer, wenn B2 von einem Schiff passiert wird).
- Das Öffnen und Schließen der Brücken wird zur Vereinfachung durch die Prozesse A, S1 und S2 gesteuert. Der Code dafür ist bereits in den Codefragmenten der Prozesse enthalten und braucht nicht mehr programmiert zu werden.

Code für Prozesse und Initialisierungen

```
/**  Schiff S1  **/

void S1(void)

{
```



```
    /* Schiff passiert B1 */

    if (!B1_open) open_B1();

    passiere_B1();
```



```
}
```

```
/**  Schiff S2  **/

void S2(void)

{
```



```
    /* Schiff passiert B2 */

    if (!B2_open) open_B2();

    passiere_B2();
```



```
}
```

```
/**  Auto A  **/

void A(void)

{
```



```
    /* B1 passieren */

    if (B1_open) close_B1();

    passiere_B1();
```



```
    /* B2 passieren */

    if (B2_open) close_B2();

    passiere_B2();
```



```
}
```

```
/**  Semaphorinitialisierungen  **/
```

2 Deadlock (25)

Im folgenden Beispiel konkurrieren 5 Prozesse um 4 verschiedene Arten von Ressourcen. Zum betrachteten Zeitpunkt sind 2 Ressourcen vom Typ 1, eine Ressource vom Typ 3 und keine Ressourcen des Typs 2 oder 4 verfügbar.

Die *Allocation Matrix* A gibt die gegenwärtig von jedem der 5 Prozesse allokierten Ressourcen an, die Matrix Q beschreibt die momentan von den Prozessen angeforderten und noch nicht gewährten Ressourcen (in den Matrizen repräsentiert jede Spalte eine Ressource und jede Zeile einen Prozess).

Kann in dieser Situation ein Deadlock vorliegen? Führen Sie den *Deadlock Detection* Algorithmus durch, um diese Frage zu beantworten.

$$A = \begin{pmatrix} 2 & 0 & 1 & 1 \\ 0 & 1 & 3 & 0 \\ 1 & 1 & 0 & 3 \\ 0 & 0 & 0 & 0 \\ 1 & 2 & 1 & 1 \end{pmatrix} \quad Q = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 2 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 3 & 3 & 0 & 4 \\ 1 & 0 & 3 & 0 \end{pmatrix}$$

Ein Deadlock kann durch das schrittweise Terminieren einzelner Prozesse aufgehoben werden. Illustrieren Sie diese Vorgangsweise anhand des vorangehenden Beispiels und erklären Sie, warum nach den vorgenommenen Terminierungen kein Deadlock vorliegt.

Welche anderen Strategien zur Aufhebung eines Deadlocks kennen Sie?

Jeder Prozess benötig für seine Abarbeitung CPU und Arbeitsspeicher. Welche Strategien werden angewandt, um das Auftreten eines Deadlocks durch den Wettbewerb um diese Ressourcen auszuschließen? Erklären Sie, wie diese Verfahren auf einem Computersystem konkret realisiert werden.

3 Fragen zu Betriebssystemen (45)

Was versteht man unter einem *Microkernel*? Welche Services stellt ein Microkernel zur Verfügung? Welche Vor- bzw. Nachteile ergeben sich bei der Verwendung eines Microkernel-Betriebssystems? (5)

Welcher Vorteil ergibt sich aus der Einführung von *Threads* für den Benutzer? Wodurch kommt es zu diesem Vorteil? Worauf muss der Benutzer bei der Programmierung von Threads achten? (4)

Was versteht man unter *Long-Term Scheduling*, *Mid-Term Scheduling* und *Short-Term Scheduling*? Erklären Sie jeden der Begriffe. (3)

Bei welchen der folgenden *Scheduling*-Strategien kann es zur *Starvation* kommen: (a) FCFS, (b) Shortest Job First, (c) Round Robin, (d) Priority Scheduling? Begründen Sie jeweils Ihre Antwort. (4)

Was ist *Thrashing*, wodurch kommt es dazu? Wie erkennt das Betriebssystem Thrashing? Wie kann dieses Problem beseitigt werden? (4)

Wir betrachten ein System mit *Virtual Memory Management*. Diskutieren Sie, welche der folgenden Situationen beim Referenzieren einer virtuellen Adresse auftreten bzw. nicht auftreten kann: (a) TLB Miss ohne Page Fault, (b) TLB Miss mit Page Fault, (c) TLB Hit ohne Page Fault, (d) TLB Hit mit Page Fault. (4)

Beschreiben Sie, auf welche Arten in einem Paging-System Speicherschutz realisiert werden kann. (4)

Was versteht man unter *Buffering*? Welche Vorteile bietet es, wo liegen seine Grenzen und worauf hat man bei der Verwendung von Puffern bei der Betriebssystemimplementierung zu achten? (5)

Was versteht man unter einer *File Allocation Table*? Wie ist diese organisiert? (2)

Eine Festplatte hat 5000 Zylinder, die von 0 bis 4999 nummeriert sind. Es wird gerade auf Zylinder 195 zugegriffen, der vorhergehende Zugriff erfolgte auf Zylinder 164. Requests auf folgende Zylinder stehen zur Behandlung an (in FIFO Reihenfolge angegeben): 92, 1470, 917, 1841, 967, 1518, 1154, 1620, 173. Über wieviele Zylinder muss sich der Lese/Schreibkopf von der aktuellen Position in Summe bewegen, um die Requests nach folgenden Strategien abzuarbeiten: (a) FCFS, (b) Elevator Algorithm (SCAN) und (c) C-SCAN? (6)

Nennen Sie Design Prinzipien für die Konstruktion von sicheren Systemen. Geben Sie für jede Regel ein Beispiel an. (4)