VO 182.711 / 182.737 28. Januar 2019

Prüfung Betriebssysteme

KNr. MNr. Zuname, Vorname

Studium: ☐ Informatik (Bakk.) ☐ Elektrotechnik (Master)

Ges.)(100) 1.)(30) 2.)(30) 3.)(40) Zusatzblätter:

Bitte verwenden Sie nur dokumentenechtes Schreibmaterial!

1 Synchronisation mit Semaphoren (30)

Mit Hilfe der Prozesse *Schiff1*, *Schiff2* und *Passagier* soll der Betrieb von zwei Rundfahrtschiffen simuliert werden.

- *Schiff1, Schiff2*: Jeder dieser Prozesse wird genau einmal gestartet und simuliert eines der Schiffe. Die Schiffe fahren abwechselnd. Für jede Rundfahrt lassen sie Passagiere an der Anlegestelle einsteigen, fahren eine Runde und lassen dann die Passagiere wieder aussteigen.
  Der Prozess *SchiffX* ($X \in \{1, 2\}$) simuliert das Öffnen und Schließen des Schiffszugangs zum Ein- und Aussteigen an der Anlegestelle durch Aufrufe der Funktionen *zugang_auf()* bzw. *zugang_zu()*. Die Rundfahrt erfolgt durch Aufruf der Funktion *rundfahrt()*.

- *Passagier*: Von diesem Prozess kann es beliebig viele Instanzen im System geben. Er simuliert einen Passagier bei der Schiffahrt.
  Die Funktionen *einsteigen()* und *aussteigen()* simulieren das Einsteigen in bzw. Aussteigen aus dem gerade verfügbaren Schiff.

Ergänzen Sie die auf der folgenden Seite gegebenen Prozesstemplates ausschließlich mit **Semaphoroperationen** (die Verwendung von globalen Variablen ist nicht erlaubt), um die Prozesse zu synchronisieren. Lokale Variable und Schleifen dürfen Sie verwenden, wenn Sie diese für Ihre Lösung benötigen. Weiters sind folgende Punkte zu beachten:

- Jedes Schiff kann N Fahrgäste aufnehmen. Das Schiff fährt vom Anlegeplatz ab, wenn es voll ist. Das Einsteigen von Passagieren beginnt, sobald alle Passagiere der vorigen Rundfahrt das Schiff verlassen haben und das andere Schiff voll mit Fahrgästen beladen ist.

- Beim Ein- und Aussteigen kann der Zugang zum Schiff immer nur von einem Passagier gleichzeitig betreten werden.

Initialisierungen

```
/** Code Schiff1 **/
for(;;) {

  zugang_auf();

  zugang_zu();

  rundfahrt();

  zugang_auf();

  zugang_zu();

}
```

```
/** Code Schiff2 **/
for(;;) {

  zugang_auf();

  zugang_zu();

  rundfahrt();

  zugang_auf();

  zugang_zu();

}
```

```
/** Code Passagier **/

  einsteigen();

/* geniessen */

  aussteigen();
```

2 Deadlock (30)

Gegeben sind zwei Prozesse, P_1 und P_2, die Ressourcen der Kategorien R_1 (2 Stück), R_2 (ein Stück) und R_3 (ein Stück) unter Mutual Exclusion verwenden.

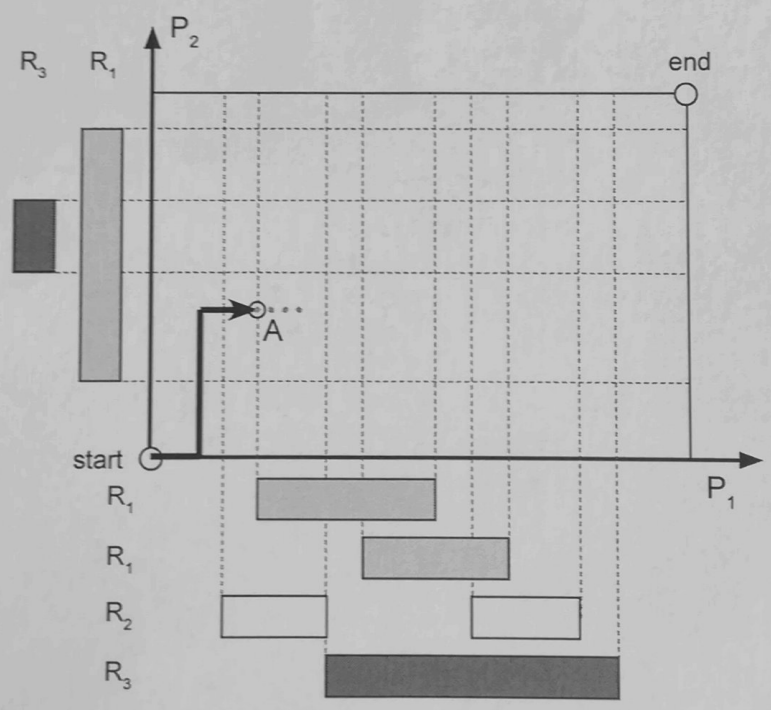
Weiters zeigt ein Prozessfortschrittsdiagramm die Ressourcenbelegung von P_1 und P_2 während ihrer Abarbeitung, sowie den Teil einer (quasi)parallelen Abarbeitung der Prozesse als Kantenzug zwischen den Punkten *start* und A.

Teilaufgabe A (Prozessfortschrittsdiagramm)

1. Umranden und schraffieren Sie im Fortschrittsdiagramm jene Bereiche, durch die der Kantenzug einer (quasi)parallelen Abarbeitung wegen einer Ressourcenüberschreitung nicht möglich ist.

2. Kennzeichnen Sie auf unterschiedliche Weise die Bereiche, die von einem Kantenzug bei einer deadlockfreien Abarbeitung von P_1 und P_2 nicht passiert werden dürfen. Markieren Sie diesen Bereich deutlich mit dem Wort „*Deadlock*".

3. Setzen Sie den gegebenen Kantenzug im Punkt A fort, sodass er eine gültige Abarbeitung von P_1 und P_2 darstellt.



Teilaufgabe B (Deadlock Avoidance)

Stellen Sie mit Hilfe des *Banker's Algorithm* fest, ob die durch den Kantenzug gegebene teilweise Abarbeitung der beiden Prozesse in Punkt A durch Fortführung des Prozesses P_1 (d.h., durch die Zuweisung einer Ressource der Kategorie R_1 an P_1) deadlockfrei fortgesetzt werden kann. Erstellen Sie die notwendigen Vektoren und Matrizen und wenden Sie den Banker's Algorithmus an. Beantworten Sie, ob P_1 am Punkt A fortgesetzt werden kann.

3 Fragen zu Betriebssystemen (40)

Nennen sie zwei Abstraktionen, die ein Betriebssystem dem Benutzer bietet. Welche Aufgaben führt das Betriebssystem durch, um diese Abstraktionen zu realisieren? (4)

Wie unterstützt ein Betriebsysstem die Portabilität von Software? (2)

Geben Sie drei Hardwaremechanismen an, mit denen moderne Mikroprozessorsysteme die Aufgaben des Betriebssystems unterstützen. Charakterisieren sie für jeden genannten Mechanismus kurz, wie diese Unterstützung erfolgt. (3)

Was versteht man unter einem *Process Image*? Erklären Sie, aus welchen Teilen ein Process Image besteht? (4)

Nennen Sie zwei verschiedene Möglichkeiten, wie Sie mittels Nachrichtenkommunikation (ohne andere Synchronisationskonstrukte) in einem Computersystem einen konsistenten Datenaustausch zwischen parallelen Prozessen realisieren können. (4)

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)

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 den Aufbau und die Verwendung einer *Multilevel Page Table* (eventuell mit Skizze). Warum werden Multilevel Page Tables verwendet? (3)

Welches Dateiformat einer regulären Datei hat für ein Betriebssystem besondere Bedeutung? Warum ist dieses Format wichtig? Wie wird es erkannt? (2)

Mit welcher logischen Struktur des I/O Systems versucht man bei der Realisierung von I/O Funktionen sowohl ein einheitliches Programmierinterface, als auch eine möglichst gerätespezifische Ansteuerung zu erreichen? (4)

Was versteht man unter *Synchronous* bzw. *Asynchonous I/O*? Beschreiben Sie die beiden Arten, I/O-Operationen durchzuführen. (3)

Beschreiben Sie das typische Layout einer Disk bzw. eines Filesystems. Erklären Sie kurz die einzelnen Komponenten. (3)