VO 182.711 7. November 2014

Prüfung Betriebssysteme

KNr. MNr. Zuname, Vorname

Ges.)(100)	1.)(30)	2.)(20)	3.)(50)	Zusatzblätter:

Bitte verwenden Sie nur dokumentenechtes Schreibmaterial!

1 Synchronisation mit Semaphoren (30)

Gegeben sind die Templates von drei Prozessen, A, B und C, die jeweils eine Folge von Funktionen ($a1()$, $a2()$, ..., $b1()$, $b2()$, ..., $c1()$, $c2()$, ...) aufrufen. Fügen Sie in die Codefragmente für die drei Prozesse *Semaphoreoperationen* ein, sodass die Abarbeitung der Prozesse die folgenden Anforderungen erfüllt.

- Die Funktionen $a1()$, $b1()$ und $c1()$ sind in genau dieser Reihenfolge auszuführen.
- Die Funktionen $a2()$, $b2()$ und $c2()$ sollen parallel ausführbar sein, d.h. die Synchronisationskonstrukte dürfen die Parallelität der entsprechenden Programmteile nicht einschränken.
- Für die Ausführung von $a3()$, $b3()$ und $c3()$ muss gelten, dass maximale Parallelität möglich ist, wobei sichergestellt werden muss, dass jede der drei Funktionen erst dann ausgeführt wird, wenn die Ausführung der drei Funktionen $a2()$, $b2()$ und $c2()$ bereits abgeschlossen ist.
- Die Ausführung von $a4()$ darf sich nicht mit der Ausführung von $b4()$ oder $c4()$ überlappen, während die Parallelausführung von $b4()$ und $c4()$ möglich sein muss.

Vervollständigen Sie die Initialisierungen und ergänzen Sie die fehlenden Semaphoroperationen in die folgenden Programmfragmenten. Gehen Sie davon aus, dass jeder der drei Prozesse nur ein Mal gestartet wird.

Initialisierungen

/** Code Prozess A **/	/** Code Prozess B **/	/** Code Prozess C **/
a1();	b1();	c1();
a2();	b2();	c2();
a3();	b3();	c3();
a4();	b4();	c4();

2 Deadlock (20)

Gegeben sind zwei Prozesse, P_1 und P_2, die die beiden Ressourcen R_1 und R_2 unter Mutual Exclusion verwenden.

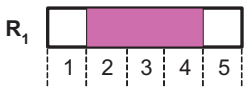
Der Fortschritt von P_1 und P_2 bei der (quasi)parallelen Abarbeitung kann im Prozessfortschrittsdiagramm als Kantenzug zwischen den Punkten *start* und *end* eingetragen werden. Die Achsenbeschriftung entspricht dabei der Zeilennummer des gerade auszuführenden Befehls.

Unterhalb bzw. links der Diagrammachsen sind Balken abgedruckt, in denen die Reservierugen von Ressourcen für P_1 bzw. P_2 eingetragen werden.

Teilaufgabe A

1. Tragen Sie die Reservierungen von Ressourcen für P_1 bzw. P_2 in die entsprechenden Balken ein. Dabei gilt eine Ressource ab Start der Anweisung `get()` als belegt und nach Beendigung der Anweisung `free()` als wieder freigegeben:

```
2: get(R1)
3: ...
4: free(R1)
```



2. Umranden und schraffieren Sie in der Grafik jene Bereiche, durch die der Kantenzug einer (quasi)parallelen Abarbeitung wegen Ressourcenkonflikten nicht möglich ist.

3. 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.

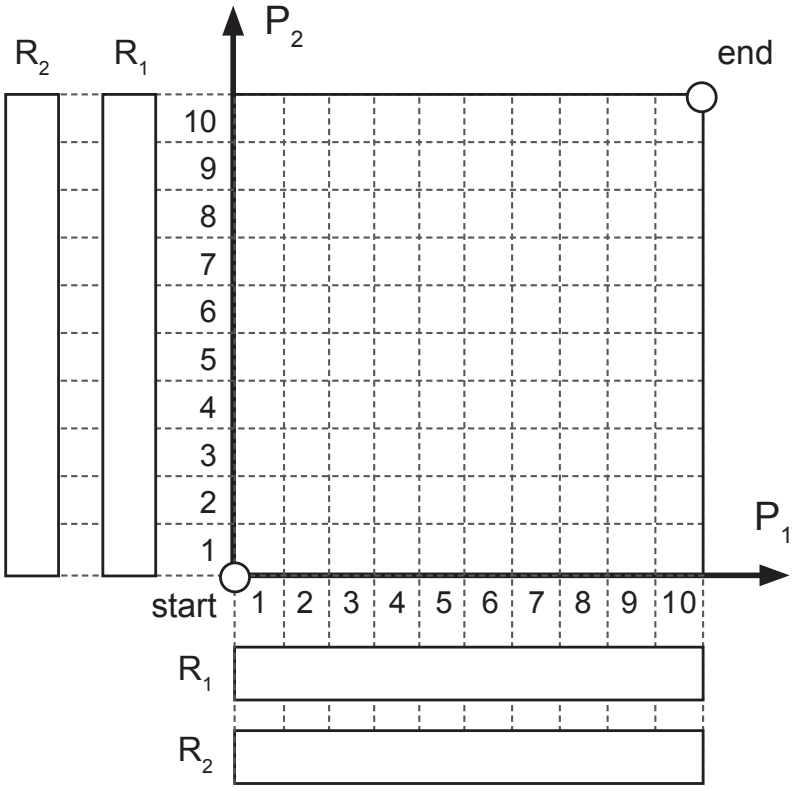
4. Zeichnen Sie einen Kantenzug für eine gültige, deadlockfreie Abarbeitung von P_1 und P_2 in der Grafik ein.

Program P_1:

```
 1: read(buf);
 2: get(R1);
 3: copy(R1, buf);
 4: get(R2);
 5: use(R1, R2);
 6: copy(buf2, R2);
 7: use2(R2, R1);
 8: free(R2);
 9: free(R1);
10: return;
```

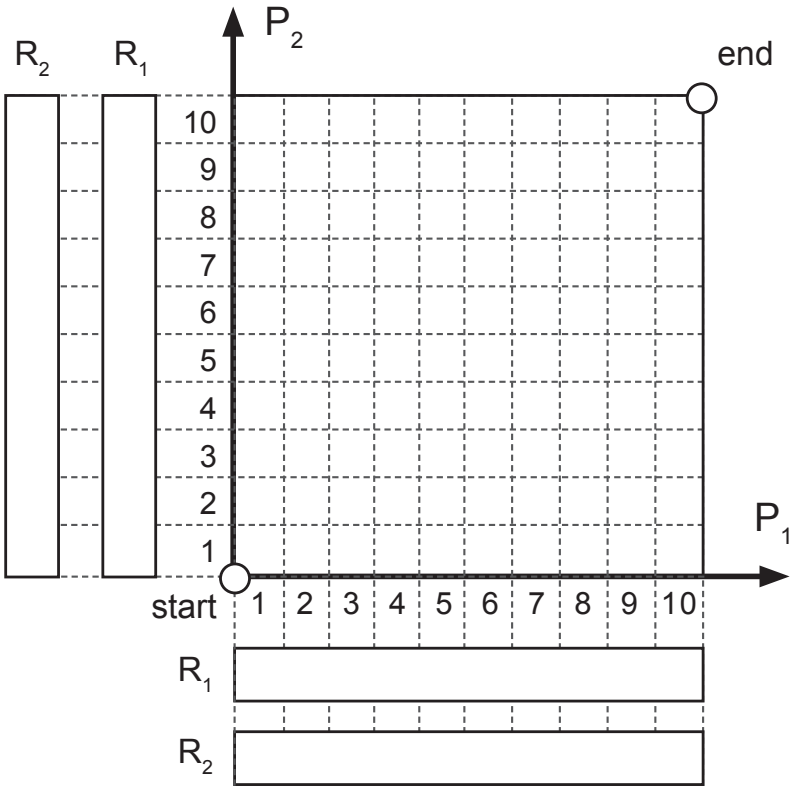
Program P_2:

```
 1: read(buf3);
 2: get(R2);
 3: copy(R2, buf3);
 4: clear(buf4);
 5: get(R1);
 6: copy(buf4, R1);
 7: use(R1, R2);
 8: free(R2);
 9: free(R1);
10: return;
```



Teilaufgabe B (Deadlock Prevention)

Nehmen Sie nun ein System an, das *Hold and Wait* unterbindet. Tragen Sie die modifizierten Lösungen für die Punkte 1 bis 4 aus Teilaufgabe A im folgenden Diagramm ein.



3 Fragen zu Betriebssystemen (50)

Was versteht man unter einem *Process Control Block*? Beschreiben Sie, aus welchen Teilen der PCB besteht und welche Informationen in diesen Teilen jeweils verwaltet werden. (6)

Wodurch unterscheiden sich die Prozesszustände *Ready* und *Blocked*? (2)

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)

Wie unterscheidet sich das Blockierverhalten von *Kernel Level Threads* und *User Level Threads*? (2)

Für die Lösung des Problems des geregelten Eintritts in einen kritischen Abschnitt werden drei Eigenschaften gefordert. (a) Nennen Sie diese drei Eigenschaften und erklären Sie deren Bedeutung. (b) Wodurch werden die drei Eigenschaften gewährleistet, wenn Semaphore zum Schutz eines kritischen Abschnitts verwendet werden? (5)

Gegeben sei ein Computersystem, in dem Ihnen zur Synchronisation bzw. Kommunikation von Prozessen nur Nachrichten zur Verfügung stehen (d.h., es gibt keine Semaphore oder andere Synchronisationskonstrukte). Nennen Sie zwei verschiedene Möglichkeiten, wie Sie in diesem Computersystem einen konsistenten Datenaustausch zwischen parallelen Prozessen realisieren können. (4)

Bei der Realisierung von Dateisystemen gibt es verschiedene Möglichkeiten, um die zu einer Datei gehörenden Datenblöcke zu organisieren bzw. auffindbar zu machen (Block-Allokierung). Nennen Sie vier verschiedene Strategien zur Block-Allokierung von Dateien und beschreiben Sie diese mit ihren Vor- und Nachteilen. (6)

Wozu wird die *Clock Policy* verwendet? Beschreiben Sie deren Funktionsweise. (5)

Was versteht man unter der *Working-Set Strategie*? Beschreiben Sie deren Funktionsweise und erklären Sie, wie diese Strategie zur Optimierung eines Paging-Systems eingesetzt werden kann. (5)

Beschreiben Sie Aufgabe und Funktion eines *Translation Lookaside Buffers*? Worauf hat man bei der Betriebssystemimplementierung bei einem Process Switch zu achten, wenn man einen Translation Lookaside Buffer verwendet? (4)

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

Beschreiben Sie drei *Naming Strategien*, die in *Verteilten Dateisystemen (DFS)* verwendet werden. (4)