VO 182.711 / 182.737 25. Januar 2022

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 Semphoren (30)

Ein Softwarepaket aus vier Prozessen soll eine Anlage simulieren, die Kartons mit Würfeln und Murmeln befüllt. Zwei gleichartige Prozesse simulieren den An- und Abtransport von Kartons zur bzw. von der Befüllanlage. Je ein Prozess kontrolliert das Einfüllen von Murmeln bzw. Würfeln in einen bereitstehenden Karton.

Ergänzen Sie die gegeben Codestücke mit *Semaphoroperationen*, um die Prozesse entsprechend der angeführten Bedingungen zu synchronisieren. Das Verwenden von globalen Variablen zur Synchronisation oder Kommunikation zwischen Prozessen ist nicht erlaubt.

- Die beiden Prozesse für den An- und Abtransport der Kartons sollen Roboter simulieren. Jeder Roboter hat einen eigenen Vorrat an leeren Kartons und einen eigenen Ablageplatz für volle Kartons und führt wiederholt folgende Aktionen durch:
  - Karton vom Vorrat holen: *karton_aufnehmen()*.
  - Karton zur Befüllanlage transportieren und öffnen: *karton_bereitstellen()*.
  - Karton verschließen und von der Befüllanlage nehmen: *karton_entfernen()*.
  - Karton zum Ablageplatz abtransportieren: *karton_abtransportieren()*.
- Der Prozess Murmelspender ruft wiederholt die Funktion *Murmel_liefern()* auf. Jeder Aufruf der Funktion lässt eine Murmel in den Karton auf der Befüllanlage fallen.
- Der Prozess Würfelspender ruft wiederholt die Funktion *Wuerfel_liefern()* auf. Jeder Aufruf der Funktion lässt einen Würfel in den Karton auf der Befüllanlage fallen.
- Die Prozesse sind so zu synrchonisieren, dass die Parallelität ihrer Aktionen die folgenden Regeln erfüllt, darüberhinaus aber möglichst wenig eingeschränkt wird.
- Auf dem Befüllplatz der Anlage ist Platz für einen Karton. Die Reihenfolge, in der die Roboter die Befüllanlage mit Kartons versorgen, ist nicht einzuschränken.
- Jeder Karton ist mit N Murmeln und $N+K$ Würfeln zu befüllen. Dabei sollen zuerst N Murmeln bzw. Würfel abwechselnd, beginnend mit einem Würfel, und dann die restlichen K Würfel in die Kartons gefüllt werden.

(a) **Initialisierungen** (vor Start der Prozesse)

In der Folge sind Codestücke für die zu synchronisierenden Prozesse gegeben, die bereits alle notwendigen Funktionsaufrufe enthalten. Ergänzen Sie in diesen Codestücken die fehlenden Semaphoroperationen zur Synchronisation.

(b) **Prozesse für die Befüllung**

```
/** Murmelspender **/

do {

    Murmel_liefern();

} while(1);
```

```
/** Wuerfelspender **/

do {

    Wuerfel_liefern();

    Wuerfel_liefern();

} while(1);
```

(c) Prozesse für Roboter

```
/**  Roboter R1  **/

do {

    karton_aufnehmen(R1);

    karton_bereitstellen(R1);

    karton_entfernen(R1);

    karton_abtransportieren(R1);

} while (1)
```

```
/**  Roboter R2  **/

do {

    karton_aufnehmen(R2);

    karton_bereitstellen(R2);

    karton_entfernen(R2);

    karton_abtransportieren(R2);

} while(1);
```

2 Scheduling (30)

Schedulen Sie das nebenstehende Taskset mit den Verfahren *Round Robin* (RR), *Shortest Remaining Time* (SRT), bzw. *Highest Response Ratio Next* (HRRN). Der Overhead für den Taskwechsel ist vernachlässigbar. Bei *RR* beträgt die Zeitscheibenlänge 2 Zeiteinheiten.

Task	Arrival Time	Service Time
P1	0	6
P2	1	4
P3	4	2
P4	8	5
P5	9	3

Fallen beim *RR*-Scheduling das Ende einer Zeitscheibe und die Ankunft eines Tasks zeitlich zusammen, so behandeln Sie den unterbrochenen Task vor dem neuen. Bei *SRT* bzw. *HRRN* ist zu beachten, dass Tasks zu ihrem Ankunftszeitpunkt sofort gescheduled werden können. Gibt es zu einem Zeitpunkt mehrere Tasks mit höchster Priorität, so soll der Task mit der niedrigsten Nummer ausgeführt werden (also beispielsweise $P4$ vor $P5$).

Markieren Sie in den nachstehenden Vorlagen zu jedem Zeitpunkt den jeweils aktiven Task. Kennzeichnen Sie die *Arrival Time* jedes Tasks mit einem Pfeil "→" und den Zeitpunkt, zu dem ein Task seine gesamte *Service Time* konsumiert hat, mit "←". Eine Vorlage dient als Ersatz. Streichen Sie gegebenenfalls die falsch ausgefüllte Vorlage deutlich durch.

Scheduling nach dem **RR**-Verfahren:

P1	→																			
P2																				
P3																				
P4																				
P5																				

0 5 10 15

Scheduling nach dem **SRT**-Verfahren:

P1	→																			
P2																				
P3																				
P4																				
P5																				

0 5 10 15

Scheduling nach dem **HRRN**-Verfahren:

P1	→																			
P2																				
P3																				
P4																				
P5																				

0 5 10 15

Ersatztabelle: Scheduling nach dem _____ -Verfahren:

P1	→																			
P2																				
P3																				
P4																				
P5																				

0 5 10 15

3 Fragen zu Betriebssystemen (40)

Was versteht man unter einem *Microkernel*? Welche zentralen Services muss ein Microkernel zur Verfügung stellen? (4)

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. (5)

Erklären Sie die Begriffe *Process Switch* und *Mode Switch*, sowie die Beziehung, in der diese beiden Konzepte stehen. Zählen Sie weiters die drei Klassen von Ereignissen auf, die einen Mode Switch nach sich ziehen. (4)

Nennen Sie die Bedingungen für das Eintreten eines Deadlocks und erklären Sie diese. (5)

Welche Schritte sind notwendig, um eine Datei, deren absoluter Pfadname gegeben ist, im Filesystem zu finden? (2)

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. (5)

Was versteht man unter dem Begriff *Relocation*? Wofür ist Relocation von Bedeutung? (2)

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? (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)

Nennen Sie die drei Kategorien von *Security Threats* und beschreiben Sie diese. Geben Sie für jede Kategorie an, welches grundlegende Security-Ziel dadurch bedroht wird. (4)