Betriebssysteme, Kapitel 4.2

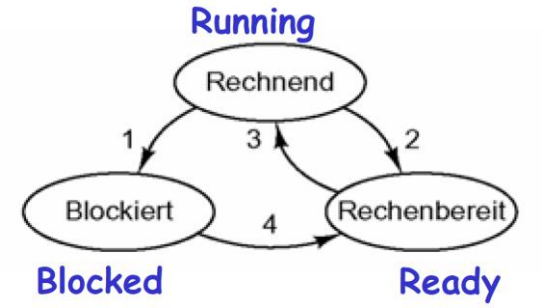
A) Fragen

1. Wie ist es möglich, dass auf einer einzigen CPU mehrere Prozesse parallel ablaufen?
   Scheduler von Betriebssystem: Die Prozesse werden quasi parallel ausgeführt.

2. Wie werden, technisch gesehen, Prozesse erzeugt?
   Via Sysemaufruf: Systemcall

3. Nennen Sie die drei Zustände im Leben eines Prozesses
   Rechnend, Blockiert, Rechenbereit

4. Zeichnen Sie - möglichst auswendig – die Zustandsübergänge eines Prozesses.



5. Kann mehr als ein Prozess gleichzeitig rechenbereit sein?
   Ja

6. Kann mehr als ein Prozess gleichzeitig rechnend sein?
   Bei einem Single Prozessor: Nein
   Multicore CPU: Ja

7. Beschreiben Sie die Semantik eines Semaphors.
   - *Down-Operation, **P(s)***
     - *falls Semaphor = 0, dann sleep*
     - *erniedrige Semaphor falls > 0*
   - *Up-Operation, **V(s)***
     - *erhöhe den Semaphor und wecke einen auf das Semaphor wartenden Prozess wieder auf*

B) Theoriefragen

Bezeichnen Sie bei den folgenden Fragen alle richtigen Antworten mit einem Kreuz.

a) Welche Aufgaben erfüllen Prozesse in einem Programmsystem?

- ☐ Sie können wie Prozeduren oder Methoden gerufen werden.
- ☐ Sie stellen über eine Interfaceliste Routinen und Variablen der Umgebung zur Verfügung.
- ☑ Sie sind die Träger der Aktivitäten im Programmsystem
- ☑ Sie dienen der dynamischen Strukturierung der zu lösenden Aufgabe.

b) Was versteht man unter dem Begriff MUTEX?

- ☐ Eine Programmiersprache für Parallele Prozesse.
- ☑ Den wechselseitigen Ausschluss beim parallelen Zugriff auf gemeinsame Betriebsmittel.
- ☐ Eine virtuelle Maschine zur Ausführung Paralleler Prozesse.
- ☐ Eine Maschineninstruktion zur Erzeugung nebenläufiger Prozesse.

c) Welche Aussagen treffen zu:

- ☐ Damit Prozesse parallel ablaufen können, dürfen sie überhaupt nicht miteinander verkoppelt sein.
- ☑ Prozesse stellen fast unabhängige, lose gekoppelte Recheneinheiten dar.
- ☑ Parallele Prozesse können auch auf virtuellen Maschinen laufen.
- ☑ Parallele Programme lassen sich in kritische Abschnitte und unkritische Abschnitte aufteilen.

d) Was passiert bei einem V(Semaphor) wenn kein Prozess darauf wartet?

- ☐ Das Signalisieren hat keine Wirkung, es verpufft ins Leere.
- ☑ Der dem Semaphor zugeordnete Wert wird verändert. Es wird kein Prozess aufgeweckt. (da ja kein Prozess wartet)
- ☐ Der dem Semaphor zugeordnete Wert wird verändert. Es wird ein beliebiger schlafender Prozess aufgeweckt.
- ☐ Es entsteht ein Laufzeitfehler, der vom System abgefangen wird.

e) Auf wie viele verschiedene Varianten können n=5 Prozesse eine einzige Instruktion ausführen?

- ☐ Auf 1 Art
- ☐ Auf 5 Arten
- ☑ Auf 120 Arten
- ☐ Auf 240 Arten

Zuerst stehen 5 Prozesse zur Auswahl, dann 4, 3, 2, 1:
*5 * 4 * 3 * 2 * 1 => 5! = 120*

f) Auf wie viele verschiedene Arten kann ein Prozess ein Programm mit n = 5 Instruktionen ausführen?

- ☐ Auf n ! (n Fakultät) = 5*4*3*2*1 = 120 Arten
- ☐ Das kann nicht generell beantwortet werden, es kommt darauf an, in wie viele atomare Instruktionen die n=5 Instruktionen zerlegt werden.
- ☑ Auf genau eine Art: sequentiell.
- ☐ Die Anzahl hängt vom Initialwert des Semaphors ab.

Da es nur ein einziger Prozess ist, kann er nur auf eine Art, sequentiell ausgeführt werden.

C) Aufgaben

1. Parallele Programmausführung *

Nennen Sie alle möglichen Schlussresultate der Variablen „x“. Erklären und dokumentieren Sie Ihren Lösungsweg!

"//" bedeutet, dass die drei Prozesse quasi-parallel in Ausführung sind.

Randbedingung: Jeder Prozess läuft nur 1 mal.

```
var s2: semaphore(0); s1: semaphore(1); X:=3;
concurrent_begin
      Prozess1:: P(s2); P(s1); x:=x*2; V(s1)
   // Prozess2:: P(s1); x:=x*x; V(s1)
   // Prozess3:: P(s1); x:=x+5;V(s2); V(s1)
concurrent_end
```

Prozess1 kann nicht als erster starten weil er die Semaphore P(s2) benötigt, welche auf 0 ist.

P2 oder P3 kann als erstes Starten:

P2 -> P3 -> P1 *x = 3 -> 9 -> 14 -> 28*

P3 -> P1 -> P2 *x = 3 -> 8 -> 16 -> 256*

P3 -> P2 -> P1 *x = 3 -> 8 -> 64 -> 128*

2. Addiere Parallel (Fehleranalyse)

Gegeben ist folgender Programmausschnitt, der in einer gemeinsamen Variablen, die Anzahl der BesucherInnen zählt, die durch zwei Drehkreuze (Prozesse) ein Museum betreten:

```
class Counter {
   int value=0;
   ……
   void increment() {
      int temp = value;          // Lesen
      Simulate.HWinterrupt();    // Scheduling Erzwingen
      value = temp + 1;          // Addieren, Zurückspeichern
   }
}
……
Counter people = new Counter();
```

Durch jedes der beiden Drehkreuze betreten 20 BesucherInnen das Museum, d.h. zwei Prozesse rufen parallel (gleichzeitig) `people.increment();`

Was gibt das Resultat? Erklären Sie! Ist das Programm korrekt? Falls das Programm fehlerhaft ist, geben Sie ein präzises Beispiel (ein Szenario) einer fehlerhaften Programmausführung an und korrigieren Sie das Programm!

Bsp: value = 17

Beide Prozesse laufen parallel und speichern in temp den Wert 17, danach warten beide Prozesse auf den Interrupt. Der erste Prozess bekommt den Interrupt und speichert den Wert 18 zurück in value. Danach wird auch der Prozess 2 „freigegeben“, dieser speichert dann ebenfalls 18 in value zurück. Eigentlich müsste es jetzt 19 sein.

Der kritische Bereich ist die gemeinsame Zahl value.

Möglicher Lösungsvorschlag:

```
   void increment() {
      Simulate.HWinterrupt();    // Scheduling Erzwingen
      P(s1);                     // Semaphore, Bereich schützen
      value = value + 1;         // 1 Addieren, Zurückspeichern
      V(s1);                     // Semaphore, Bereich freigeben
   }
```

Mögliche Probleme: Falls der Interrupt zu lange dauert wird ggf. die Methode nicht abgearbeitet bis zum nächsten Interrupt.