**Ahmed Ahmed,Wassim Benmansour,Frank Nordemann,Leonid Wodzinski**

**Auswertung der verschiedenen Strategien :**

Die Hauptziele der fünf Strategien sind die Mutual Exclusion der Prozesse 1 und 2, damit sie wenn sie parallel laufen, nicht gleichzeitig in den kritischen Bereich kommen können, und die Vermeidung von Deadlocks und Livelocks, wenn die beiden Prozesse aufeinander unendlich lang warten würden.

Daher versuchen wir, die beste Lösungsstrategie in dieser Aufgabe zu finden und erklären das Prinzip jeder Strategie, die möglichen Probleme und die Ursachen dieser Probleme.

**Strategie 1 :**

Die Variable turn wurde auf 1 deklariert und die beide Prozesse kommunizieren miteinander durch die Variable turn, wobei turn nach dem kritischen Bereich jedes Prozesses zwischen 1 und 2 wechselt und in jedem Lauf den Wert von der Variable turn überprüft, damit beide Prozesse nicht gleichzeitig in den kritischen Bereich kommen können.

Das Problem in dieser Strategie liegt darin, dass wenn ein Prozess außerhalb seines kritischen Bereiches gestoppt wird, der andere Prozess als Folge davon blockiert wird und deswegen ist diese Lösung nicht die gewünschte Lösung.

Protokoll der Konsolenausgabe: EWD123a

2020/04/15 11:43:58.594579 controller.go:67: outside CS: 2 (100 msecs) // Prozess 2 läuft außerhalb des KB.

2020/04/15 11:43:58.634581 controller.go:32: entered CS: 1 // Prozess 1 darf zuerst den KB betreten.

2020/04/15 11:43:58.634581 controller.go:57: inside CS: 1 (50 msecs) // Prozess 1 läuft innerhalb des KB.

2020/04/15 11:43:58.684584 controller.go:47: left CS: 1 // Prozess 1 verlässt den KB.

2020/04/15 11:43:58.684584 controller.go:67: outside CS: 1 (100 msecs) // Prozess 1 läuft außerhalb des KB.

2020/04/15 11:43:58.694585 controller.go:32: entered CS: 2 //Prozess 2 darf danach den KB betreten.

2020/04/15 11:43:58.694585 controller.go:57: inside CS: 2 (50 msecs) // Prozess 2 läuft in den KB.

2020/04/15 11:43:58.744588 controller.go:47: left CS: 2

2020/04/15 11:43:58.744588 controller.go:67: outside CS: 2 (100 msecs)

2020/04/15 11:43:58.784590 controller.go:32: entered CS: 1 // und Prozess 1 wechselt sich jetzt immer wieder ab.

2020/04/15 11:43:58.784590 controller.go:57: inside CS: 1 (50 msecs)

2020/04/15 11:43:58.834593 controller.go:47: left CS: 1

2020/04/15 11:43:59.044605 controller.go:47: left CS: 2 //Prozess 2 verlässt den KB.

2020/04/15 11:43:59.044605 controller.go:67: outside CS: 2 (100 msecs)

2020/04/15 11:43:59.087607 ewd123a.go:48: Process 1 crashed // Prozess 1 wird außerhalb des KBs abgebrochen.

2020/04/15 17:36:41.595017 ewd123c.go:37: Process 2 waiting // ab jetzt wartet Prozess 2, kann aber nie wieder den KB betreten, weil Prozess 1 die turn-Variable nicht mehr zurücksetzen kann.

2020/04/15 17:36:41.595017 ewd123c.go:37: Process 2 waiting

2020/04/15 17:36:41.595017 ewd123c.go:37: Process 2 waiting

2020/04/15 17:36:41.596017 ewd123c.go:37: Process 2 waiting

2020/04/15 17:36:41.596017 ewd123c.go:37: Process 2 waiting

2020/04/15 17:36:41.596017 ewd123c.go:37: Process 2 waiting

2020/04/15 17:36:41.596017 ewd123c.go:37: Process 2 waiting

2020/04/15 17:36:41.596017 ewd123c.go:37: Process 2 waiting

2020/04/15 17:36:41.596017 ewd123c.go:37: Process 2 waiting

**Strategie 2 :**

Die Variablen c1,c2 wurden beide auf 1 deklariert und werden immer andersherum im Verlauf der parallelen Prozesse vor dem kritischen Bereich auf 0 und nach dem kritischen Bereich auf 1 initialisiert, damit die beiden Prozesse nicht gleichzeitig in den kritischen Bereich kommen können.

Das Problem in dieser Strategie ist, dass wenn der Wert von c1 = 0 beim ersten Prozess vor dem Kritischen Bereich initialisiert wird und danach direkt der Wert von c2 = 0 beim zweiten Prozess vor dem kritischen Bereich initialisiert wird, kommen die beide Prozesse zusammen in den kritischen Bereich und deswegen ist diese Lösung nicht die gewünschte Lösung.

Protokoll der Konsolenausgabe: EWD123b

2020/04/15 13:29:25.331157 controller.go:32: entered CS: 1 //Prozess 1 darf zuerst den KB betreten.

2020/04/15 13:29:25.331157 ewd123b.go:65: Process 2 waiting // Prozess 2 muss zuerst warten.

2020/04/15 13:29:25.331157 ewd123b.go:65: Process 2 waiting

2020/04/15 13:29:25.331157 controller.go:57: inside CS: 1 (0 msecs) // Prozess 1 läuft innerhalb KB.

2020/04/15 13:29:25.331157 ewd123b.go:65: Process 2 waiting // Prozess 2 wartet weiter.

2020/04/15 13:29:25.331157 ewd123b.go:65: Process 2 waiting // Prozess 2 wartet weiter.

2020/04/15 13:29:25.331157 controller.go:47: left CS: 1 //Prozess 1 verlässt den KB.

2020/04/15 13:29:25.331157 ewd123b.go:65: Process 2 waiting // Prozess 2 wartet weiter.

2020/04/15 13:29:25.331157 controller.go:67: outside CS: 1 (1 msecs) // Prozess 1 läuft außerhalb des KB.

2020/04/15 13:29:25.331157 controller.go:32: entered CS: 2 //Prozess 2 darf danach den KB betreten.

2020/04/15 13:29:25.331157 controller.go:57: inside CS: 2 (0 msecs)

2020/04/15 13:29:25.331157 controller.go:47: left CS: 2 // Prozess 2 verlässt KB.

2020/04/15 13:29:25.331157 controller.go:67: outside CS: 2 (1 msecs) // Prozess 2 läuft außerhalb des KB.

2020/04/15 13:29:25.332157 controller.go:32: entered CS: 2

2020/04/15 13:29:25.332157 ewd123b.go:37: Process 1 waiting // Prozess 1 wartet.

2020/04/15 13:29:25.332157 controller.go:57: inside CS: 2 (0 msecs) // Prozess 2 läuft innerhalb des KB.

2020/04/15 13:29:25.332157 ewd123b.go:37: Process 1 waiting //Prozess 1 wartet weiter.

2020/04/15 13:29:25.332157 ewd123b.go:37: Process 1 waiting //Prozess 1 wartet weiter.

2020/04/15 13:29:25.332157 controller.go:47: left CS: 2 //Prozess 2 verlässt den KB.

2020/04/15 13:29:25.332157 ewd123b.go:37: Process 1 waiting //Prozess 1 wartet weiter.

2020/04/15 13:29:25.332157 controller.go:32: entered CS: 1 //Prozess 1 wächselt sich immer wieder ab.

2020/04/15 13:29:25.332157 controller.go:57: inside CS: 1 (0 msecs) //Prozess 1 läuft innerhalb des KB.

2020/04/15 13:29:25.332157 controller.go:47: left CS: 1 //Prozess 1 verlässt den KB.

2020/04/15 13:29:25.332157 controller.go:67: outside CS: 1 (1 msecs) // Prozess 1 läuft außerhalb des KB.

2020/04/15 13:29:25.332157 controller.go:67: outside CS: 2 (1 msecs) // Prozess 2 läuft außerhalb des KB.

2020/04/15 13:29:25.333157 controller.go:32: entered CS: 1 // Prozess 1 darf den KB betreten.

2020/04/15 13:29:25.333157 controller.go:32: entered CS: 2 // Prozess 2 darf auch den KB gleichzeitig mit Prozess 1 betreten ohne es zu warten, weil C1 und C2 auf den Wert 1 initialisiert wurden,danach versuchen sie den KB nacheinander zu betreten.

2020/04/15 13:29:25.333157 controller.go:55: Process 1 tried to work inside the critical section while it has not occupied it // Das Programm wird abgebrochen, weil die beide Prozesse nicht gleichzeitig im KB stehen können.

**Strategie 3 :**

Strategie 3 ist genauso wie die Strategie 2, der einzige Unterschied ist, dass die Initialisierung von c1,c2 = 0 am Anfang des Prozesses geschieht.

Das Problem dieser Strategie ist, dass wenn die Werte von c1 = 0 beim ersten Prozess und c2 = 0 beim zweiten Prozess nacheinander direkt initialisiert werden, kommen die beiden Prozesse zum Livelock(Mutual Blocking), wobei jeder Prozess auf den anderen in einer Endlosschleife wartet.

Protokoll der Konsolenausgabe: EWD123c

2020/04/15 17:32:03.815129 controller.go:32: entered CS: 2 // Prozess 2 darf zuerst in den KB bertreten.

2020/04/15 17:32:03.815129 controller.go:57: inside CS: 2 (1 msecs) //Prozess 2 läuft innerhalb des KB.

2020/04/15 17:32:03.816129 controller.go:47: left CS: 2 //Prozess 2 verlässt den KB.

2020/04/15 17:32:03.816129 controller.go:67: outside CS: 2 (1 msecs) //Prozess 2 läuft außerhalb des KB.

2020/04/15 17:32:03.816129 controller.go:32: entered CS: 1 //Prozess 1 darf danach den KB betreten.

2020/04/15 17:32:03.816129 controller.go:57: inside CS: 1 (1 msecs) //Prozess 1 läuft innerhalb des KB.

2020/04/15 17:32:03.817129 controller.go:47: left CS: 1 //Prozess 1 verlässt den KB.

2020/04/15 17:32:03.817129 controller.go:67: outside CS: 1 (1 msecs) //Prozess 1 läuft außerhalb des KB.

2020/04/15 17:32:03.817129 controller.go:32: entered CS: 2 //Prozess 2 wächselt sich immer wieder ab.

2020/04/15 17:32:03.817129 controller.go:57: inside CS: 2 (1 msecs) //Prozess 2 läuft innerhalb des KB.

2020/04/15 17:32:03.818129 controller.go:47: left CS: 2 //Prozess 2 verlässt den KB.

2020/04/15 17:32:03.818129 controller.go:67: outside CS: 2 (1 msecs) //Prozess 2 läuft außerhalb des KB.

2020/04/15 17:32:03.818129 controller.go:32: entered CS: 1 //Prozess 1 wächselt sich immer wieder ab.

2020/04/15 17:32:03.818129 controller.go:57: inside CS: 1 (1 msecs) //Prozess 1 läuft innerhalb des KB.

2020/04/15 17:32:03.819129 controller.go:47: left CS: 1 //Prozess 1 verlässt den KB.

2020/04/15 17:32:03.819129 controller.go:67: outside CS: 1 (1 msecs) //Prozess 1 läuft außerhalb des KB.

2020/04/15 17:32:03.819129 controller.go:32: entered CS: 2 //Prozess 2 wächselt sich immer wieder ab.

2020/04/15 17:32:03.819129 controller.go:57: inside CS: 2 (1 msecs) //Prozess 2 läuft innerhalb des KB.

2020/04/15 17:32:03.821129 controller.go:47: left CS: 2 //Prozess 2 verlässt den KB.

2020/04/15 17:32:03.821129 controller.go:67: outside CS: 2 (1 msecs) //Prozess 2 läuft außerhalb des KB.

2020/04/15 17:32:03.821129 controller.go:32: entered CS: 1 //Prozess 1 wächselt sich immer wieder ab.

2020/04/15 17:32:03.821129 controller.go:57: inside CS: 1 (1 msecs) //Prozess 1 läuft innerhalb des KB.

2020/04/15 17:32:03.824129 controller.go:47: left CS: 1 //Prozess 1 verlässt den KB.

2020/04/15 17:32:03.824129 controller.go:67: outside CS: 1 (1 msecs) //Prozess 1 läuft außerhalb des KB.  
2020/04/15 17:36:41.591017 ewd123c.go:63: Process 2 waiting //Prozess 2 muss zuerst warten.

2020/04/15 17:36:41.591017 ewd123c.go:63: Process 2 waiting //Prozess 2 wartet weiter.

2020/04/15 17:36:41.592017 ewd123c.go:63: Process 2 waiting //Prozess 2 wartet weiter.

2020/04/15 17:36:41.592017 ewd123c.go:63: Process 2 waiting

2020/04/15 17:36:41.592017 ewd123c.go:63: Process 2 waiting

2020/04/15 17:36:41.592017 ewd123c.go:63: Process 2 waiting

2020/04/15 17:36:41.591017 ewd123c.go:37: Process 1 waiting //Prozess 1 wartet Prozess 2 während Prozess 2 wartet auch Prozess 1, dann die beide Prozesse sind in diesem Fall in Endlosschleife (Live Lock),weil sie bis unendlich warten werden.

2020/04/15 17:36:41.592017 ewd123c.go:37: Process 1 waiting

2020/04/15 17:36:41.592017 ewd123c.go:37: Process 1 waiting //Prozess 1 wartet weiter in Live Lock.

2020/04/15 17:36:41.593017 ewd123c.go:37: Process 1 waiting

2020/04/15 17:36:41.595017 ewd123c.go:37: Process 1 waiting

2020/04/15 17:36:41.596017 ewd123c.go:37: Process 1 waiting

2020/04/15 17:36:41.596017 ewd123c.go:37: Process 1 waiting

2020/04/15 17:36:41.596017 ewd123c.go:37: Process 1 waiting

2020/04/15 17:36:41.596017 ewd123c.go:37: Process 1 waiting

2020/04/15 17:36:41.595017 ewd123c.go:63: Process 2 waiting //Prozess 2 wartet weiter in Live Lock.

2020/04/15 17:36:41.596017 ewd123c.go:63: Process 2 waiting

**Strategie 4 :**

Strategie 4 ist genauso wie die Strategie 3, aber diese Strategie führt nicht zum Mutual Blocking, weil der Wert von C wieder auf 1 zurückgesetzt wird, bevor der Prozess wiederholt wird.

Aber auch diese Lösung ist nicht ideal wegen der ineffizienten Geschwindigkeit, weil kein Prozess in den kritischen Bereich hereinkommen kann, solange der andere C Wert auf 0 gesetzt wird, was Wartezeit braucht.

Wenn die Prozesse gleichzeitig in den kritischen Bereich kommen, sollen keine Verschiebungen oder Zeitverlängerungen verursacht werden, deshalb ist diese Lösung auch nicht ganz effizient..

Protokoll der Konsolenausgabe: EWD123d

2020/04/15 17:46:59.795376 controller.go:32: entered CS: 2 // Prozess 2 darf zuerst in den KB bertreten.

2020/04/15 17:46:59.795376 controller.go:57: inside CS: 2 (1 msecs) //Prozess 2 läuft innerhalb des KB.

2020/04/15 17:46:59.796376 ewd123d.go:38: Process 1 waiting //Prozess 1 muss zuerst warten.

2020/04/15 17:46:59.796376 ewd123d.go:38: Process 1 waiting //Prozess 1 muss weiter warten

2020/04/15 17:46:59.796376 ewd123d.go:38: Process 1 waiting

2020/04/15 17:46:59.797376 ewd123d.go:38: Process 1 waiting

2020/04/15 17:46:59.797376 ewd123d.go:38: Process 1 waiting //Prozess 1 muss weiter warten,solange C im Prozess 2 auf 0 gesetzt ist.

2020/04/15 17:46:59.797376 ewd123d.go:38: Process 1 waiting

2020/04/15 17:46:59.797376 ewd123d.go:38: Process 1 waiting

2020/04/15 17:46:59.797376 ewd123d.go:38: Process 1 waiting //Prozess 1 muss weiter warten,solange C im Prozess 2 auf 0 gesetzt ist.

2020/04/15 17:46:59.798376 ewd123d.go:38: Process 1 waiting

2020/04/15 17:46:59.798376 ewd123d.go:38: Process 1 waiting

2020/04/15 17:46:59.798376 ewd123d.go:38: Process 1 waiting

2020/04/15 17:46:59.798376 ewd123d.go:38: Process 1 waiting g //Prozess 1 wartet für lange Zeit, weil C wert im Prozess 2 auf 0 gesetzt ist, und noch nicht auf 1 initialisiert.

2020/04/15 17:46:59.798376 ewd123d.go:38: Process 1 waiting

2020/04/15 17:46:59.799376 ewd123d.go:38: Process 1 waiting //Prozess 1 wartet für lange Zeit, weil C wert im Prozess 2 auf 0 gesetzt ist, und noch nicht auf 1 initialisiert wurde.

2020/04/15 17:46:59.799376 ewd123d.go:38: Process 1 waiting

2020/04/15 17:46:59.799376 ewd123d.go:38: Process 1 waiting

2020/04/15 17:46:59.799376 ewd123d.go:38: Process 1 waiting

2020/04/15 17:46:59.798376 controller.go:47: left CS: 2 //Prozess 2 verlässt den KB.

2020/04/15 17:46:59.800376 controller.go:32: entered CS: 1 //Prozess 1 darf danach den KB betreten.

2020/04/15 17:46:59.801376 controller.go:57: inside CS: 1 (1 msecs) //Prozess 1 läuft innerhalb des KB.

2020/04/15 17:46:59.800376 controller.go:67: outside CS: 2 (1 msecs) //Prozess 2 läuft außerhab des KB.

2020/04/15 17:46:59.802376 ewd123d.go:66: Process 2 waiting //Prozess 2 muss warten

2020/04/15 17:46:59.802376 ewd123d.go:66: Process 2 waiting

2020/04/15 17:46:59.802376 ewd123d.go:66: Process 2 waiting //Prozess 2 muss weiter warten,solange C im Prozess 1 auf 0 gesetzt ist.

2020/04/15 17:46:59.802376 ewd123d.go:66: Process 2 waiting

2020/04/15 17:46:59.803376 ewd123d.go:66: Process 2 waiting

2020/04/15 17:46:59.803376 ewd123d.go:66: Process 2 waiting //Prozess 2 muss weiter warten,solange C im Prozess 1 auf 0 gesetzt ist.

2020/04/15 17:46:59.803376 ewd123d.go:66: Process 2 waiting

2020/04/15 17:46:59.804376 ewd123d.go:66: Process 2 waiting

2020/04/15 17:46:59.802376 controller.go:47: left CS: 1 //Prozess 1 verlässt den KB.

2020/04/15 17:46:59.804376 controller.go:67: outside CS: 1 (1 msecs) //Prozess 2 läuft außerhab des KB.

2020/04/15 17:46:59.804376 ewd123d.go:66: Process 2 waiting

2020/04/15 17:46:59.805377 ewd123d.go:66: Process 2 waiting

2020/04/15 17:46:59.805377 ewd123d.go:66: Process 2 waiting //Prozess 2 wartet für lange Zeit, weil C wert im Prozess 1 auf 0 gesetzt ist, und noch nicht auf 1 initialisiert wurde.

2020/04/15 17:46:59.805377 ewd123d.go:66: Process 2 waiting

2020/04/15 17:46:59.805377 ewd123d.go:66: Process 2 waiting

2020/04/15 17:46:59.806377 ewd123d.go:66: Process 2 waiting

2020/04/15 17:46:59.805377 controller.go:32: entered CS: 1 //Prozess 1 wächselt sich immer wieder ab.

2020/04/15 17:46:59.806377 ewd123d.go:66: Process 2 waiting

2020/04/15 17:46:59.806377 ewd123d.go:66: Process 2 waiting //Prozess 2 wartet für lange Zeit, weil C wert im Prozess 1 auf 0 gesetzt ist, und noch nicht auf 1 initialisiert wurde.

2020/04/15 17:46:59.807377 ewd123d.go:66: Process 2 waiting

2020/04/15 17:46:59.806377 controller.go:57: inside CS: 1 (1 msecs) //Prozess 1 läuft innerhalb des KB.

2020/04/15 17:46:59.807377 ewd123d.go:66: Process 2 waiting

2020/04/15 17:46:59.807377 ewd123d.go:66: Process 2 waiting

2020/04/15 17:46:59.807377 ewd123d.go:66: Process 2 waiting

2020/04/15 17:46:59.808377 ewd123d.go:66: Process 2 waiting //Prozess 2 wartet für lange Zeit, weil C wert im Prozess 1 auf 0 gesetzt ist, und noch nicht auf 1 initialisiert wurde.

2020/04/15 17:46:59.808377 ewd123d.go:66: Process 2 waiting

2020/04/15 17:46:59.808377 ewd123d.go:66: Process 2 waiting

2020/04/15 17:46:59.808377 ewd123d.go:66: Process 2 waiting //Prozess 2 wartet für lange Zeit, weil C wert im Prozess 1 auf 0 gesetzt ist, und noch nicht auf 1 initialisiert wurde.

2020/04/15 17:46:59.809377 ewd123d.go:66: Process 2 waiting

2020/04/15 17:46:59.809377 ewd123d.go:66: Process 2 waiting

2020/04/15 17:46:59.808377 controller.go:47: left CS: 1 //Prozess 1 verlässt den KB.

2020/04/15 17:46:59.811377 controller.go:67: outside CS: 1 (1 msecs) //Prozess 1 läuft außerhalb des KB.

2020/04/15 17:46:59.810377 ewd123d.go:66: Process 2 waiting

2020/04/15 17:46:59.811377 controller.go:32: entered CS: 2 //Prozess 1 wächselt sich immer wieder ab.

**Strategie 5 :**

Strategie 5 ist die bestmögliche Lösungsstrategie aller fünf angegebenen Strategien. Sie hat das selbe Arbeitsprinzip wie Strategie 4, aber mit einer Lösung für ihr Problem, bei dem das Eintreten des kritischen Bereiches von der Initialisierung der Variable C abhängt, was eine Wartezeit für jeden Prozess kostet um in den kritischen Bereich reinkommen zu können, deswegen nutzt diese Strategie noch die Variable turn um die Initialisierung von c = 1 zu halten bis der andere Prozess den kritischen Bereich verlässt, sodass keine Zeit dazwischen verloren wird.

Daher können wir daraus erschließen, dass diese Strategie zur Vermeidung von Deadlocks, Livelocks, Endlosschleifen, gleichzeitiger Zugriffe der Prozesse auf den kritischen Bereich und auch von Wartezeit beim Wechsel der Prozesse, dies bedeutet das diese Straegie die gewünschte Lösung ist.

Protokoll der Konsolenausgabe: EWD123dekker

2020/04/15 17:55:28.263459 ewd123.go:33: \*\*\* Start

2020/04/15 17:55:28.265459 controller.go:32: entered CS: 2 //Prozess 2 darf zuerst in den KB betreten.

2020/04/15 17:55:28.265459 controller.go:57: inside CS: 2 (50 msecs) // Prozess 2 läuft innerhalb des KB.

2020/04/15 17:55:28.331463 controller.go:47: left CS: 2 // Prozess 2 verlässt den KB.

2020/04/15 17:55:28.331463 controller.go:67: outside CS: 2 (100 msecs) // Prozess 2 läuft außerhalb des KB.

2020/04/15 17:55:28.331463 controller.go:32: entered CS: 1 //Prozess 1 darf danach den KB betreten.

2020/04/15 17:55:28.332463 controller.go:57: inside CS: 1 (50 msecs) // Prozess 1 läuft innerhalb des KB.

2020/04/15 17:55:28.382465 controller.go:47: left CS: 1 // Prozess 1 verlässt den KB.

2020/04/15 17:55:28.382465 controller.go:67: outside CS: 1 (100 msecs) // Prozess 1 läuft außerhalb des KB.

2020/04/15 17:55:28.431468 controller.go:32: entered CS: 2 2 //Prozess 2 wächselt sich immer wieder ab.

2020/04/15 17:55:28.431468 controller.go:57: inside CS: 2 (50 msecs) // Prozess 2 läuft innerhalb des KB.

2020/04/15 17:55:28.481471 controller.go:47: left CS: 2 // Prozess 2 verlässt den KB.

2020/04/15 17:55:28.481471 controller.go:67: outside CS: 2 (100 msecs) // Prozess 2 läuft außerhalb des KB.

2020/04/15 17:55:28.482471 controller.go:32: entered CS: 1 //Prozess 2 wächselt sich immer wieder ab.

2020/04/15 17:55:28.482471 controller.go:57: inside CS: 1 (50 msecs)

2020/04/15 17:55:28.532474 controller.go:47: left CS: 1

2020/04/15 17:55:28.532474 controller.go:67: outside CS: 1 (100 msecs)

2020/04/15 17:55:28.581477 controller.go:32: entered CS: 2 //Prozess 2 wächselt sich immer wieder ab.

2020/04/15 17:55:28.581477 controller.go:57: inside CS: 2 (50 msecs)

2020/04/15 17:55:28.631480 controller.go:47: left CS: 2

2020/04/15 17:55:28.631480 controller.go:67: outside CS: 2 (100 msecs)

2020/04/15 17:55:28.632480 controller.go:32: entered CS: 1 //Prozess 1 wächselt sich immer wieder ab.

2020/04/15 17:55:28.632480 controller.go:57: inside CS: 1 (50 msecs)

2020/04/15 17:55:28.682483 controller.go:47: left CS: 1

2020/04/15 17:55:28.682483 controller.go:67: outside CS: 1 (100 msecs)

2020/04/15 17:55:28.738486 controller.go:32: entered CS: 2 //Prozess 2 wächselt sich immer wieder ab.

2020/04/15 17:55:28.739486 controller.go:57: inside CS: 2 (50 msecs)

2020/04/15 17:55:28.798489 controller.go:47: left CS: 2

2020/04/15 17:55:28.798489 controller.go:67: outside CS: 2 (100 msecs)

2020/04/15 17:55:28.798489 ewd123dekker.go:48: Process 1 waiting //Prozess 1 muss zuerst warten.

//Man merkt hier,dass Prozess 1 nur einmal wartet, und nicht für lange Zeit warten wie Strategie 4 , weil das Eintreten des KB nicht von der Initialisierung der Variable C im anderen Prozess abhängt, sondern turn Variable hält die Initialisierung von der Variable C = 1 bis der andere Prozess den KB verlässt.

2020/04/15 17:55:28.798489 controller.go:32: entered CS: 1 //Prozess 1 wächselt sich immer wieder ab.

2020/04/15 17:55:28.799489 controller.go:57: inside CS: 1 (50 msecs)

2020/04/15 17:55:28.849492 controller.go:47: left CS: 1