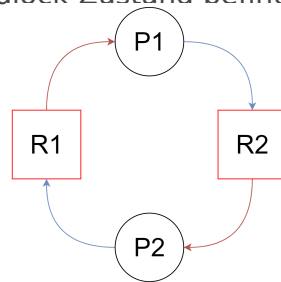


# Deadlocks

Eine Menge von Prozessen befindet sich in einem **Deadlock-Zustand**, wenn jeder Prozess aus der Menge auf ein Ereignis wartet, das nur ein anderer Prozess aus der Menge auslösen kann.

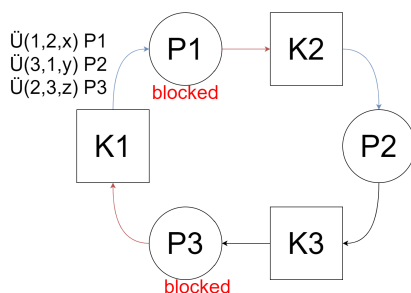
Wenn sich mehrere Prozesse in Deadlock-Zustand befinden, so sagt man auch vereinfachend:



**Es ist ein Deadlock aufgetreten.**

Vier Bedingungen damit ein Deadlock auftritt:

1. **Mutual exclusion condition:** Eine Ressource steht einem Prozess nur exklusiv zur Verfügung
2. **Wait for condition:** Prozesse warten und behalten dabei Kontrolle über bereits Ressourcen
3. **No preemption condition:** Ressourcen können Prozess nicht entrissen werden.
4. **Circular wait condition:** Es gibt eine zyklische Kette von Prozessen



**Prinzip des ersten Kontos**

- P1: Schnappt sich **K1** **Kontextwechsel**
- P2: Schnappt sich **K2** **Kontextwechsel**
- P3: Möchte **K1** haben ist aber belegt: *blockiert*; **Kontextwechsel**
- P1: Möchte **K2** haben ist aber belegt: *blockiert*; **Kontextwechsel**
- P2: schnappt sich **K3**: **Fetch, Decode, Execute**

---

## Deadlocks ignorieren

*"Es gibt keine Deadlocks, weil ich daran glaube, dass es keine Deadlocks gibt!",* sagt das Betriebssystem.

Zum Ignorieren von Deadlocks kommt in Betriebssystemen üblicherweise der **Vogel-Strauß-Algorithmus** zum Einsatz.

## Deadlocks vermeiden

Ein Betriebssystem könnte von vornherein so konstruiert werden, dass Deadlocks gar nicht möglich sind.

## Verhinderung von Deadlocks

Prozess nur dann weitere Ressource zuzusprechen, wenn sichergestellt ist, dass **in der Zukunft** kein Deadlock entstehen kann

Allgemein ist davon auszugehen, dass die Erkennung, Vermeidung oder Verhinderung von Deadlocks sehr aufwendig ist. Eine ausnahmslose Abdeckung aller denkbaren Fälle scheint nahezu unmöglich.

---

Revision #1

Created 24 September 2022 16:24:01 by Merith Holtmann

Updated 2 October 2022 19:22:32 by Merith Holtmann