

Übung zu Betriebssysteme

Gegenseitiger Ausschluss

Wintersemester 2020/21

Bernhard Heinloth & Christian Eichler

Lehrstuhl für Informatik 4
Friedrich-Alexander-Universität Erlangen-Nürnberg



Lehrstuhl für Verteilte Systeme
und Betriebssysteme



FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT

Wie sieht es mit gegenseitigem Ausschluss auf Fadenebene in STuBS aus?

Wir haben doch bereits ein `spinlock` bzw. `ticketlock` implementiert...

Mutex mit aktivem Warten

Beispiel: Drei Threads auf einer CPU

App1

```
mutex.lock()  
// code  
mutex.unlock()
```

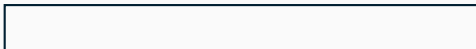
App2

```
mutex.lock()  
// code  
mutex.unlock()
```

App3

```
mutex.lock()  
// code  
mutex.unlock()
```

active



ready list



Mutex mit aktivem Warten

Beispiel: Drei Threads auf einer CPU

App1

```
mutex.lock()
```

```
// code
```

```
mutex.unlock()
```



App2

```
mutex.lock()
```

```
// code
```

```
mutex.unlock()
```

App3

```
mutex.lock()
```

```
// code
```

```
mutex.unlock()
```

active

App1

ready list

App2 App3

CPU-Zeit →



Mutex mit aktivem Warten

Beispiel: Drei Threads auf einer CPU

App1

```
mutex.lock()  
// code  
mutex.unlock()
```

App2

```
mutex.lock() ⚡  
// code  
mutex.unlock()
```

App3

```
mutex.lock()  
// code  
mutex.unlock()
```

active

App2

ready list

App3 App1

CPU-Zeit →



Mutex mit aktivem Warten

Beispiel: Drei Threads auf einer CPU

App1

```
mutex.lock()  
// code  
mutex.unlock()
```

App2

```
mutex.lock()  
// code  
mutex.unlock()
```

App3

```
mutex.lock() ⚡  
// code  
mutex.unlock()
```

active

App3

ready list

App1 App2

CPU-Zeit →



Mutex mit aktivem Warten

Beispiel: Drei Threads auf einer CPU

App1

```
mutex.lock()  
// code  
mutex.unlock()
```

App2

```
mutex.lock()  
// code  
mutex.unlock()
```

App3

```
mutex.lock()  
// code  
mutex.unlock()
```

active

App1

ready list

App2 App3

CPU-Zeit →



Verschwendung von CPU-Zeit

Mutex mit harter Synchronisation

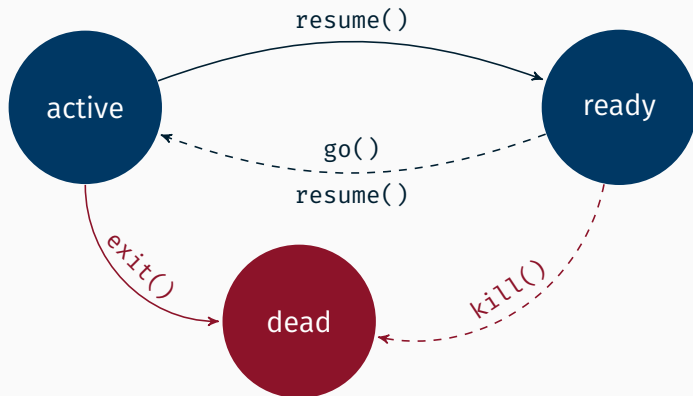
- Analog zur Interruptsperre mit `cli`
- **Ziel:** Kein (präemptives) Scheduling
- Realisierbar durch
 - Multitasking (temporär) deaktivieren
 - Erweiterung des Schedulers
 - Wechsel auf Epilogebe
- **Vorteile:**
 - konsistent
 - (relativ) einfach zu implementieren
- **Nachteile:**
 - Breitbandwirkung
 - Prioritätsverletzung
 - Prophylaktisch

Idee: Passives Warten

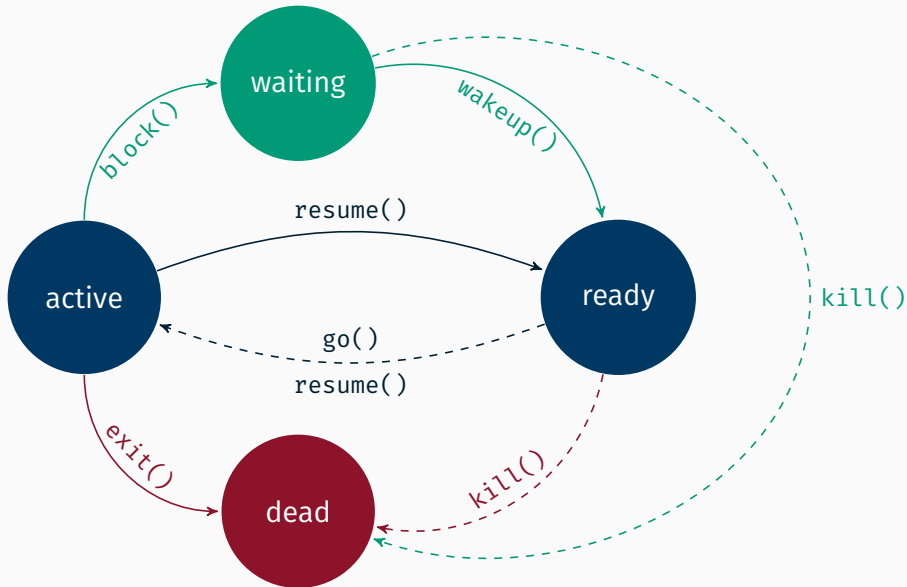
Ansatz: Fäden, die den kritischen Abschnitt nicht betreten können, werden blockiert
(d.h. von der CPU-Zuteilung ausgeschlossen)

Idee: Passives Warten

Ansatz: Fäden, die den kritischen Abschnitt nicht betreten können, werden blockiert
(d.h. von der CPU-Zuteilung ausgeschlossen)



Idee: Passives Warten



Idee: Passives Warten



Einführung eines **waiting rooms**
(Liste mit wartenden Threads)

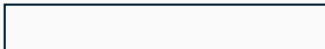
Mutex mit passivem Warten

Beispiel: Drei Threads auf einer CPU

App1

```
mutex.lock()  
// code  
mutex.unlock()  
// mehr code
```

active



App2

```
mutex.lock()  
// code  
mutex.unlock()  
// mehr code
```

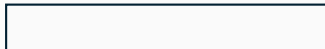
ready list



App3

```
mutex.lock()  
// code  
mutex.unlock()  
// mehr code
```

waiting room



Mutex mit passivem Warten

Beispiel: Drei Threads auf einer CPU

App1

```
mutex.lock()
```

```
// code
```

```
mutex.unlock()
```

```
// mehr code
```

App2

```
mutex.lock()
```

```
// code
```

```
mutex.unlock()
```

```
// mehr code
```

App3

```
mutex.lock()
```

```
// code
```

```
mutex.unlock()
```

```
// mehr code
```

active

App1

ready list

App2 App3

waiting room

CPU-Zeit →



Mutex mit passivem Warten

Beispiel: Drei Threads auf einer CPU

App1

```
mutex.lock()  
// code  
mutex.unlock()  
// mehr code
```

App2

```
mutex.lock()  
// code  
mutex.unlock()  
// mehr code
```

App3

```
mutex.lock()  
// code  
mutex.unlock()  
// mehr code
```

active

App2

ready list

App3 App1

waiting room

CPU-Zeit →



Mutex mit passivem Warten

Beispiel: Drei Threads auf einer CPU

App1

```
mutex.lock()  
// code  
mutex.unlock()  
// mehr code
```

active

App3

App2

```
mutex.lock()  
// code  
mutex.unlock()  
// mehr code
```

ready list

App1

App3

```
mutex.lock()  
// code  
mutex.unlock()  
// mehr code
```

waiting room

App2

CPU-Zeit →



Mutex mit passivem Warten

Beispiel: Drei Threads auf einer CPU

App1

```
mutex.lock()  
// code  
mutex.unlock()  
// mehr code
```

active

App1

App2

```
mutex.lock()  
// code  
mutex.unlock()  
// mehr code
```

ready list

App3

```
mutex.lock()  
// code  
mutex.unlock()  
// mehr code
```

waiting room

App2 App3

CPU-Zeit →



Mutex mit passivem Warten

Beispiel: Drei Threads auf einer CPU

App1

```
mutex.lock()  
// code  
mutex.unlock()  
// mehr code
```



App2

```
mutex.lock()  
// code  
mutex.unlock()  
// mehr code
```

App3

```
mutex.lock()  
// code  
mutex.unlock()  
// mehr code
```

active

App1

ready list

App2

waiting room

App3

CPU-Zeit →



Mutex mit passivem Warten

Beispiel: Drei Threads auf einer CPU

App1

```
mutex.lock()  
// code  
mutex.unlock()  
// mehr code
```

active

App2

App2

```
mutex.lock()  
// code  
mutex.unlock()  
// mehr code
```

ready list

App1

App3

```
mutex.lock()  
// code  
mutex.unlock()  
// mehr code
```

waiting room

App3

CPU-Zeit →



Mutex mit passivem Warten

Beispiel: Drei Threads auf einer CPU

App1

```
mutex.lock()  
// code  
mutex.unlock()  
// mehr code
```

App2

```
mutex.lock()  
// code  
mutex.unlock()  
// mehr code
```

App3

```
mutex.lock()  
// code  
mutex.unlock()  
// mehr code
```

active

App2

ready list

App1 App3

waiting room

CPU-Zeit →



Semaphore

sem·a·phore

1. any apparatus for signaling, as by an arrangement of lights, flags, and mechanical arms on railroads
2. a system for signaling by the use of two flags, one held in each hand: the letters of the alphabet are represented by the various positions of the arms
3. any system of signaling by semaphore

nach V. E. Neufeld, Webster's New World Dictionary, Simon & Schuster Inc., third college edition, 1988

Semaphore(*i*) Zähler *c* mit positivem Wert *i* initialisieren

P() von *Prolaag* (versuchen zu verringern)

bzw. *Passeering*[†] (passieren)

$c > 0$ dekrementieren

$c = 0$ warten

V() von *Verhoog* (erhöhen) bzw. *Vrijgave*[†] (freigeben)

- nächsten wartenden Faden aufwecken oder
- Zähler *c* erhöhen

[†] nach Edsger W. Dijkstra, Over de sequentialiteit van procesbeschrijvingen, ca. 1962

```
Semaphore mutex(1);
```

```
void func() {
```

```
    mutex.p();  // lock
```

```
    // critical section
```

```
    mutex.v();  // unlock
```

```
}
```

```
Semaphore empty(size);
```

```
Semaphore full(0);
```

```
void producer(){
```

```
    empty.p();
```

```
    // produce
```

```
    full.v();
```

```
}
```

```
void consumer(){
```

```
    full.p();
```

```
    // consume
```

```
    empty.v();
```

```
}
```