PPaDS MMXX

Matus Jokay, C-503, matus.jokay@stuba.sk
Roderik Ploszek, C-512, roderik.ploszek@stuba.sk

uim.fei.stuba.sk/predmet/i-ppds

konzultacie dohodou

Zakladne synchronizacne vzory

- Mutex
- Multiplex
- Signalizacia
- Rendezvous
- Bariera
- Znovupouzitelna bariera
- Turniket

1. Mutex

- Iba jedno vlakno moze “zamknut” mutex a pokracovat bez cakania dalej
- Ista forma “tokenu”: ten, kto drzi “token”, moze nieco konat, ostatni (ziadajuci o token) musia cakat
- Priklady zo zivota?

- Semafor(1)

Mutex

shared vars:

arr ← [0, …., 0]

ind ← 0

Thread A

```
while True:
        arr[ind] += 1
        ind += 1
```

Thread B

```
while True:
        arr[ind] += 1
        ind += 1
```

Mutex

shared vars:

arr ← [0, …., 0]

ind ← 0

mutex ← Mutex()

Thread A

```
mutex.lock()
while True:
        arr[ind] += 1
        ind += 1
mutex.unlock()
```

Thread B

```
mutex.lock()
while True:
        arr[ind] += 1
        ind += 1
mutex.unlock()
```

Mutex

BAAAD

shared vars:

arr ← [0, …., 0]

ind ← 0

mutex ← Mutex()

Thread A

```
mutex.lock()
while True:
        arr[ind] += 1
        ind += 1
mutex.unlock()
```

Thread B

```
mutex.lock()
while True:
        arr[ind] += 1
        ind += 1
mutex.unlock()
```

Mutex

GOOOD?

shared vars:

arr ← [0, …., 0]

ind ← 0

mutex ← Mutex()

Thread A

```
while True:
    mutex.lock()
    arr[ind] += 1
    ind += 1
    mutex.unlock()
```

Thread B

```
while True:
    mutex.lock()
    arr[ind] += 1
    ind += 1
    mutex.unlock()
```

Mutex

shared vars:

arr ← [0, …., 0]

ind ← 0

mutex ← Mutex()

Thread A

```
while ind < len(arr):
    mutex.lock()
    arr[ind] += 1
    ind += 1
    mutex.unlock()
```

Thread B

```
while ind < len(arr):
    mutex.lock()
    arr[ind] += 1
    ind += 1
    mutex.unlock()
```

Mutex

BAAAD

shared vars:

arr ← [0, …., 0]

ind ← 0

mutex ← Mutex()

Thread A

```
while ind < len(arr):
    mutex.lock()
    arr[ind] += 1
    ind += 1
    mutex.unlock()
```

Thread B

```
while ind < len(arr):
    mutex.lock()
    arr[ind] += 1
    ind += 1
    mutex.unlock()
```

Mutex

shared vars:

arr ← [0, …., 0]

ind ← 0

mutex ← Mutex()

Thread XYZ

```
while True:
    mutex.lock()
    if ind >= len(arr):
        break
    arr[ind] += 1
    ind += 1
    mutex.unlock()
```

Thread XYZ

```
while ind < len(arr):
    mutex.lock()
    arr[ind] += 1
    ind += 1
    mutex.unlock()
```

Mutex

shared vars:

arr ← [0, …., 0]

ind ← 0

mutex ← Mutex()

Thread XYZ

```
while True:
    mutex.lock()
    if ind >= len(arr):
        break
    arr[ind] += 1
    ind += 1
    mutex.unlock()
```

BAAAD

Mutex

shared vars:

arr ← [0, …., 0]

ind ← 0

mutex ← Mutex()

Thread XYZ

```
while True:
    mutex.lock()
    if ind >= len(arr):
        mutex.unlock()
        break
    arr[ind] += 1
    ind += 1
    mutex.unlock()
```

GOOOD?

Mutex

shared vars:

arr ← [0, …., 0]

ind ← 0

mutex ← Mutex()

Thread XYZ

```
while True:
    mutex.lock()
    if ind >= len(arr):
        mutex.unlock()
        break
    arr[ind] += 1
    ind += 1
    mutex.unlock()
```

GOOOD!

Mutex

shared vars:

arr ← [0, …., 0]

ind ← 0

mutex ← Mutex()

Thread XYZ

```
while True:
    mutex.lock()
    arr[ind] += 1
    ind += 1
    if ind >= len(arr):
        mutex.unlock()
        break
    mutex.unlock()
```

GOOOD?

Mutex

shared vars:

arr ← [0, …., 0]

ind ← 0

mutex ← Mutex()

Thread XYZ

```
while True:
    mutex.lock()
    arr[ind] += 1
    ind += 1
    if ind >= len(arr):
        mutex.unlock()
        break
    mutex.unlock()
```

VEERY BAAAD!

Mutex

shared vars:

arr ← [0, …., 0]

ind ← 0

mutex ← Mutex()

Thread XYZ

```
while True:
    mutex.lock()
    arr[ind] += 1
    ind += 1
    if ind >= len(arr):
        mutex.unlock()
        break
    mutex.unlock()
```

Mnozstvo synchronizacneho kodu oproti originalu…

Mutex

shared vars:

arr ← [0, …., 0]

ind ← 0

mutex ← Mutex()

Thread XYZ

```
while True:
    mutex.lock()
    if ind >= len(arr):
        mutex.unlock()
        break
    arr[ind] += 1
    ind += 1
    mutex.unlock()
```

GOOOD!

Da sa to lepsie?

Mutex

Per thread var:

tmp

Thread XYZ

```
while True:
    mutex.lock()
    if ind >= len(arr):
        mutex.unlock()
        break
    arr[ind] += 1
    ind += 1
    mutex.unlock()
```

Thread XYZ

```
while True:
    mutex.lock()
    tmp = ind
    ind += 1
    mutex.unlock()

    if tmp >= len(arr):
        break
    arr[tmp] += 1
```

Mutex

V com je to lepsie?

Per thread var:

tmp

Thread XYZ

```
while True:
    mutex.lock()
    tmp = ind
    ind += 1
    mutex.unlock()

    if tmp >= len(arr):
        break
    arr[tmp] += 1
```

Mutex

V com je to lepsie?

Per thread var:

tmp

Thread XYZ

```
while True:
    mutex.lock()
    tmp = ind
    ind += 1
    mutex.unlock()

    if tmp >= len(arr):
        break
    arr[tmp] += 1
```

Mutex

V tomto naivnom priklade je cas trvania tejto operacie zanedbatelny

Per thread var:

tmp

Thread XYZ

```
while True:
    mutex.lock()
    tmp = ind
    ind += 1
    mutex.unlock()

    if tmp >= len(arr):
        break
    arr[tmp] += 1
```

Mutex

Kolko operacii treba vykonat?

Per thread var:

tmp

Thread XYZ

```
while True:
    mutex.lock()
    tmp = ind
    ind += 1
    mutex.unlock()

    if tmp >= len(arr):
        break
    arr[tmp] += 1
```

Mutex

Kolko operacii treba vykonat?

Tolko, kolko je prvkov pola!

Per thread var:

tmp

Thread XYZ

```
while True:
    mutex.lock()
    tmp = ind
    ind += 1
    mutex.unlock()

    if tmp >= len(arr):
        break
    arr[tmp] += 1
```

Mutex

Tato cast aloritmu je paralelizovatelna.

Naopak, cast algoritmu uzavreta v KO je seriova.

Per thread var:

tmp

Thread XYZ

```
while True:
    mutex.lock()
    tmp = ind
    ind += 1
    mutex.unlock()

    if tmp >= len(arr):
        break
    arr[tmp] += 1
```

Mutex

Per thread var:

tmp

Thread XYZ

```
while True:
    mutex.lock()
    if ind >= len(arr):
        mutex.unlock()
        break
    arr[ind] += 1
    ind += 1
    mutex.unlock()
```

Thread XYZ

```
while True:
    mutex.lock()
    tmp = ind
    ind += 1
    mutex.unlock()

    if tmp >= len(arr):
        break
    arr[tmp] += 1
```

Mutex

Per thread var:

tmp

Thread XYZ

```
while True
    mutex.lock()
    if ind >= len(arr):
        mutex.unlock()
        break
    arr[ind] += 1
    ind += 1
    mutex.unlock()
```

Thread XYZ

```
while True:
    mutex.lock()
    tmp = ind
    ind += 1
    mutex.unlock()

    if tmp >= len(arr):
        break
    arr[tmp] += 1
```

Spracovanie prvkov pola

- Nech zistenie prvku, ktory sa ma spracovat, trva zanedbatelny cas
- Nech spracovanie prvku trva netrivialny cas

- Priklad: regal s krabicami, ktore treba odniest

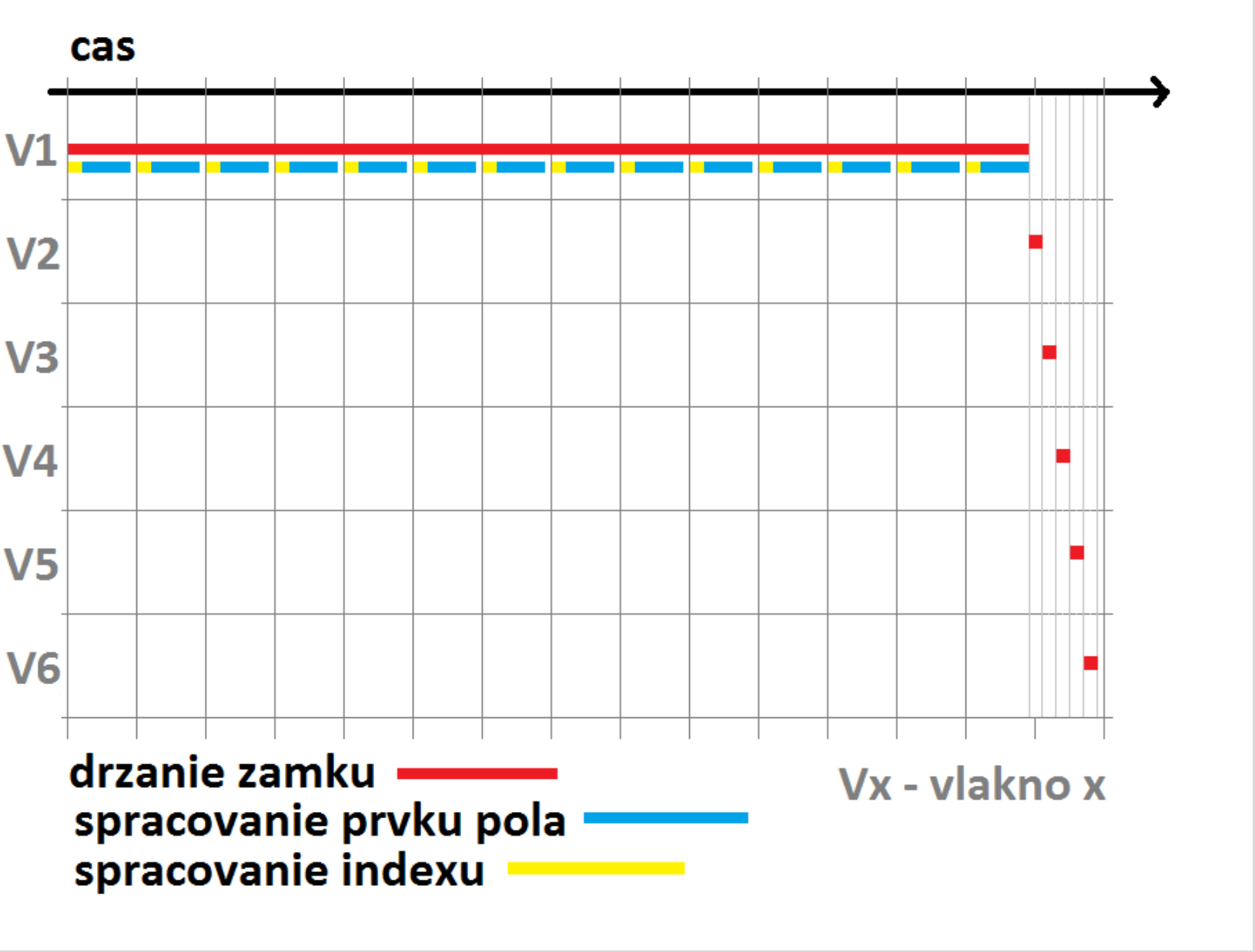
Mutex – Uroven serializacie

1. Serializacia na urovni vlakien (spracovania regalu)

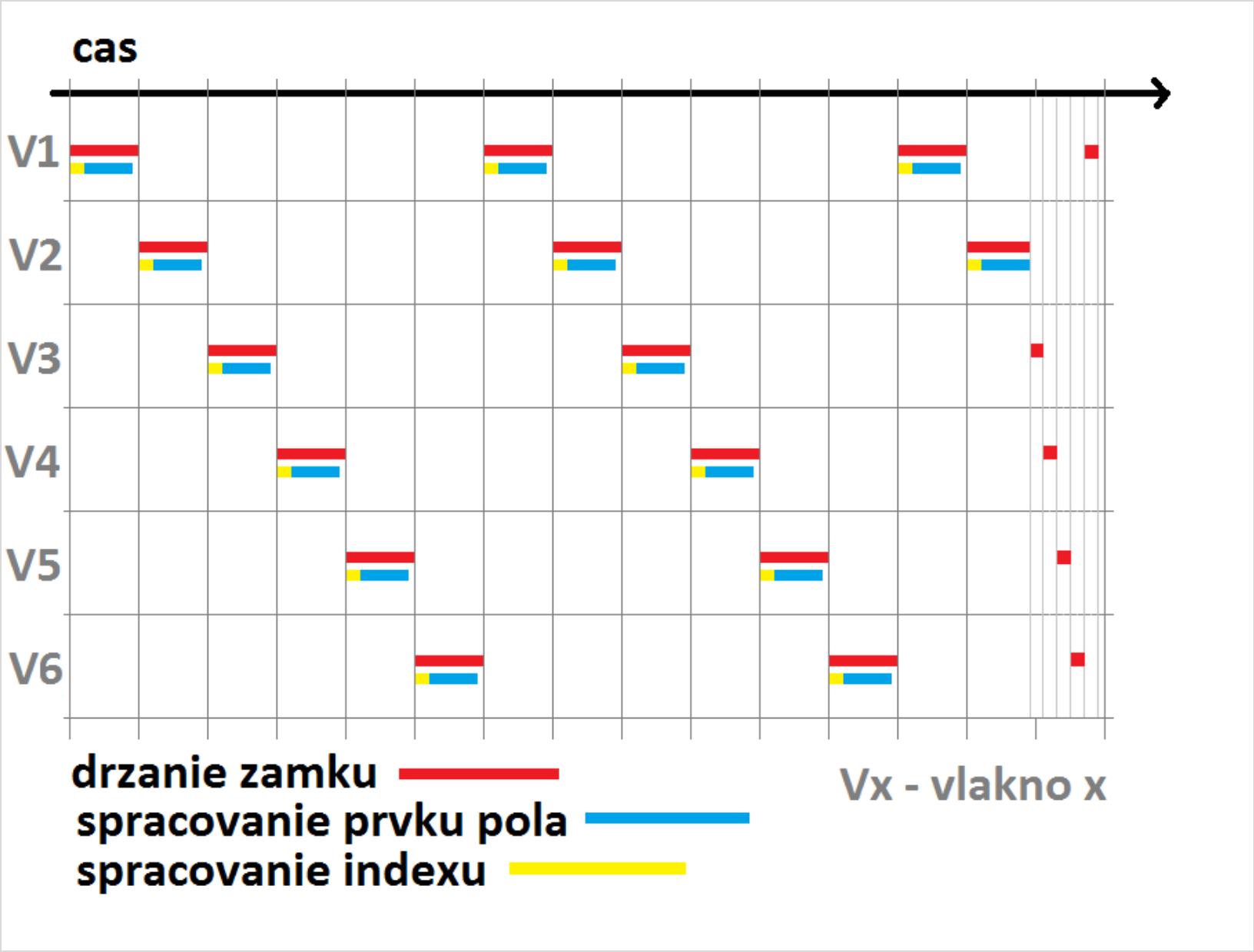
2. Serializacia na urovni pristupu k prvku pola (spracovania krabice)

3. Serializacia na urovni indexu do pola (prevzatia krabice)

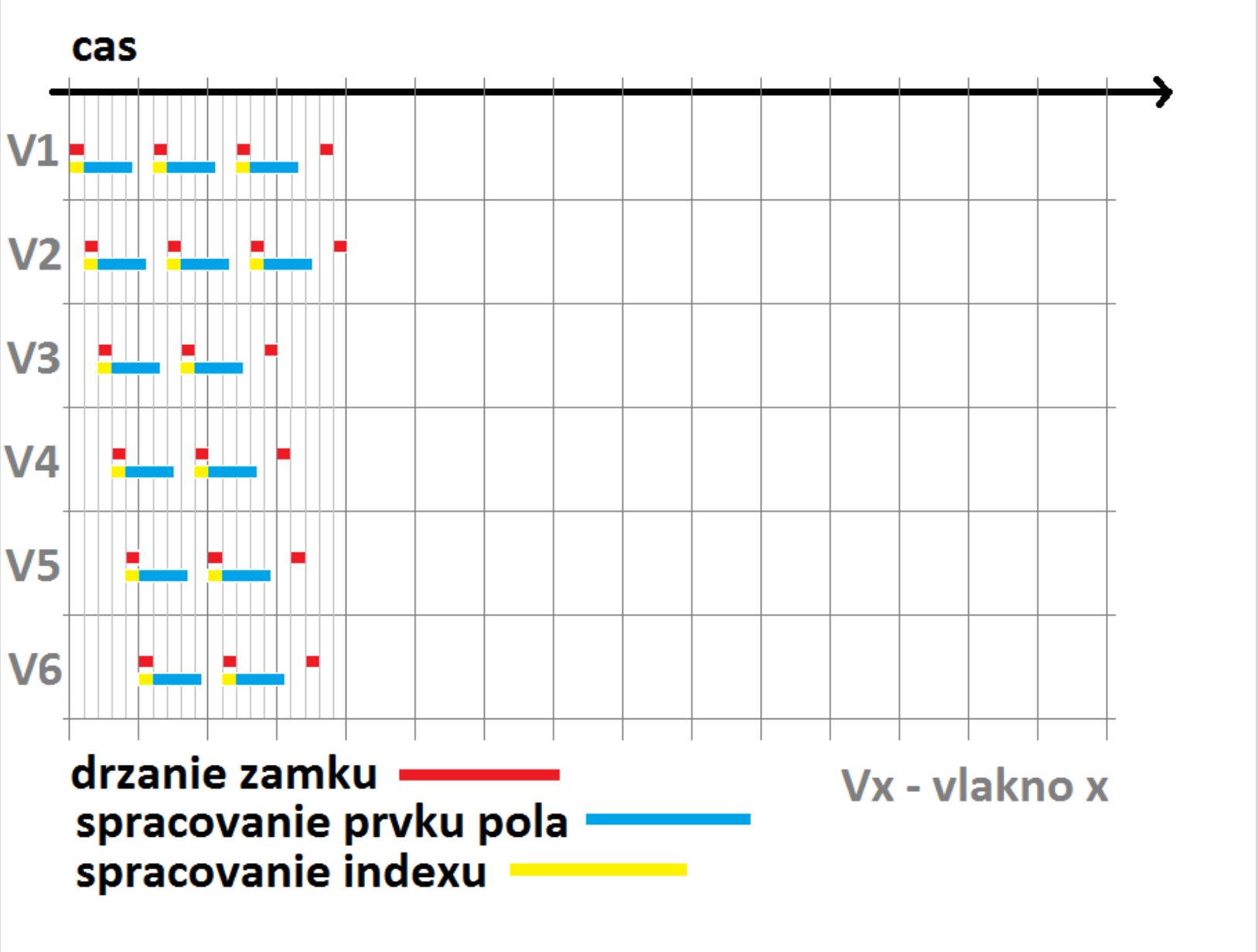
Mutex – Uroven serializacie 1



Mutex – Uroven serializacie 2



Mutex – Uroven serializacie 3



Mutex

- Zachovanie integrity je nutnou podmienkou

- Dosiahnutie konkurencie je zelanym stavom; nutna podmienka nestaci!

Mutex

- Zarucuje, ze v jednom case bude dany programovy kod vykonavat iba jediny tok riadenia

- Dosledok: SERIALIZACIA!
- Jednotlive toky riadenia vykonavaju KO postupne jeden po druhom, za sebou
- Mutex umyselne porusuje konkurentnost!

2. Multiplex

- Zovseobecnenie mutexu
- Nie 1, ale max. N vlakien moze vstupit do KO
- Napriklad: autobus, kino, sauna…
- Implementacia: multiplex = Semaphore(N)

Pouzitie inicializacia

```
mplex := Semaphore(N)
```

Pouzitie vo vlakne

```
mplex.wait()
    # kriticka oblast
mplex.signal()
```

3. Signalizacia

- Pouzitie medzi 2 vlaknami
- Jedno signalizuje druhemu nejaku udalost
- Priklad:
  - T1 nacita riadok
  - T2 ho zobrazi

Pouzitie inicializacia

```
event := Semaphore(0)
```

Pouzitie vlakno A

```
# ... nejaky kod
event.signal()
# ... nejaky kod
```

Pouzitie vlakno B

```
# ... nejaky kod
event.wait()
# ... nejaky kod
```

Signalizacia

Kto vyriesil domacu ulohu z prednasky?

4. Rendezvous

- Obojstranna signalizacia
- A caka na B a B caka na A
- Nemozu pokracovat, kym sa nestretnu

Pouzitie vlakno A	Pouzitie vlakno B
1) # … nejaky kod a1	1) # … nejaky kod b1
2) # … nejaky kod a2	2) # … nejaky kod b2

- Poziadavky tejto ulohy:

$$a1 < b2 \quad \&\& \quad b1 < a2$$

Rendezvous

- Spravne pomenovanie semaforov reprezentujucich udalosti!!!
- Meno musi reprezentovat udalost

Rendezvous

- Spravne pomenovanie semaforov reprezentujucich udalosti!!!
- Meno musi reprezentovat udalost

Inicializacia

aArrived ← Semaphore(0)

bArrived ← Semaphore(0)

Rendezvous

Inicializacia

aArrived ← Semaphore(0)

bArrived ← Semaphore(0)

Pouzitie vlakno A

1) # … nejaky kod a1
2)
3)
4) # … nejaky kod a2

Pouzitie vlakno B

1) # … nejaky kod b1
2)
3)
4) # … nejaky kod b2

??? a1 < b2 && b1 < a2 ???

Rendezvous

Inicializacia

aArrived ← Semaphore(0)

bArrived ← Semaphore(0)

Pouzitie vlakno A

1) # … nejaky kod a1
2) bArrived.wait()
3) aArrived.signal()
4) # … nejaky kod a2

Pouzitie vlakno B

1) # … nejaky kod b1
2) aArrived.wait()
3) bArrived.signal()
4) # … nejaky kod b2

??? a1 < b2 && b1 < a2 ???

Rendezvous

Inicializacia

aArrived ← Semaphore(0)

bArrived ← Semaphore(0)

Pouzitie vlakno A

1) # … nejaky kod a1
2) bArrived.wait()
3) aArrived.signal()
4) # … nejaky kod a2

Pouzitie vlakno B

1) # … nejaky kod b1
2) aArrived.wait()
3) bArrived.signal()
4) # … nejaky kod b2

??? a1 < b2 && b1 < a2 ???

Rendezvous

Inicializacia

aArrived ← Semaphore(0)

bArrived ← Semaphore(0)

Pouzitie vlakno A

1) # … nejaky kod a1
2) bArrived.wait()
3) aArrived.signal()
4) # … nejaky kod a2

Pouzitie vlakno B

1) # … nejaky kod b1
2) aArrived.wait()
3) bArrived.signal()
4) # … nejaky kod b2

UVIAZNUTIE

??? a1 < b2 && b1 < a2 ???

Rendezvous

Inicializacia

aArrived ← Semaphore(0)

bArrived ← Semaphore(0)

Pouzitie vlakno A

1) # … nejaky kod a1
2) bArrived.wait()
3) aArrived.signal()
4) # … nejaky kod a2

Pouzitie vlakno B

1) # … nejaky kod b1
2) aArrived.wait()
3) bArrived.signal()
4) # … nejaky kod b2

??? a1 < b2 && b1 < a2 ???

Rendezvous

Inicializacia

aArrived ← Semaphore(0)

bArrived ← Semaphore(0)

Pouzitie vlakno A

1) # … nejaky kod a1
2) bArrived.wait()
3) aArrived.signal()
4) # … nejaky kod a2

Pouzitie vlakno B

1) # … nejaky kod b1
2) bArrived.signal()
3) aArrived.wait()
4) # … nejaky kod b2

??? a1 < b2 && b1 < a2 ???

Rendezvous

Inicializacia

aArrived ← Semaphore(0)

bArrived ← Semaphore(0)

Pouzitie vlakno A

1) # … nejaky kod a1
2) bArrived.wait()
3) aArrived.signal()
4) # … nejaky kod a2

Pouzitie vlakno B

1) # … nejaky kod b1
2) bArrived.signal()
3) aArrived.wait()
4) # … nejaky kod b2

EFEKTIVITA

??? a1 < b2 && b1 < a2 ???

Rendezvous

Inicializacia

aArrived ← Semaphore(0)

bArrived ← Semaphore(0)

Pouzitie vlakno A

1) # … nejaky kod a1
2) bArrived.wait()
3) aArrived.signal()
4) # … nejaky kod a2

Pouzitie vlakno B

1) # … nejaky kod b1
2) bArrived.signal()
3) aArrived.wait()
4) # … nejaky kod b2

??? a1 < b2 && b1 < a2 ???

Rendezvous

Inicializacia

aArrived ← Semaphore(0)

bArrived ← Semaphore(0)

Pouzitie vlakno A

1) # … nejaky kod a1
2) aArrived.signal()
3) bArrived.wait()
4) # … nejaky kod a2

Pouzitie vlakno B

1) # … nejaky kod b1
2) bArrived.signal()
3) aArrived.wait()
4) # … nejaky kod b2

??? a1 < b2 && b1 < a2 ???

Rendezvous

Inicializacia

aArrived ← Semaphore(0)

bArrived ← Semaphore(0)

Pouzitie vlakno A

1) # … nejaky kod a1
2) aArrived.signal()
3) bArrived.wait()
4) # … nejaky kod a2

Pouzitie vlakno B

1) # … nejaky kod b1
2) bArrived.signal()
3) aArrived.wait()
4) # … nejaky kod b2

VERY GOOD

??? a1 < b2 && b1 < a2 ???

5. Bariera

- Zovseobecnenie stretnutia z 2 na N vlakien
- Ziadne z vlakien nesmie zacat vykonavat KO, pokym vsetky vlakna neskoncili vykonavanie rendezvous
- Konstanta N je dostupna vsetkym vlaknam
- Prvych N-1 vlakien sa musi zablokovat, kym nepride N-te vlakno; az potom mozu pokracovat

Vlakno (kod bariery):

1) Rendezvous
2) KO

Bariera

- N je pocet vlakien (pre 2 je to klasicka uloha Rendezvous)

- Doteraz sme si vystacili s Mutexom / Semaforom
- Pri bariere budeme potrebovat pocitadlo
  - Pritup k pocitadlu maju vsetky vlakna
  - Preto jeho integritu musime chranit serializaciou pomocou mutexu

Bariera

- N je pocet vlakien (pre 2 je to klasicka uloha Rendezvous)

- Co budeme potrebovat:
  - Pocitadlo vlakien, ktore prichadzaju k bariere
  - Mutex, ktorym budeme chranit integritu pocitadla
  - Semafor, ktory bude sluzit ako bariera (posledne vlakno, ktore pride k bariere, tento semafor uvolni, aby mohli vlakna pokracovat)

Bariera

Inicializacia:

N – vopred dane a dostupne

count ← 0

mutex ← Semaphore(1)

barrier ← Semaphore(0)

Bariera

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Semaphore(1)

barrier ← Semaphore(0)

Vlakno XYZ:

1) rendezvous

2) KO

Bariera

Vlakno XYZ:

1) rendezvous

2) KO

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Semaphore(1)

barrier ← Semaphore(0)

- Pridajme blokovanie vlakien pred vstupom do ‘KO’

Bariera

Vlakno XYZ:

1) rendezvous

2) barrier.wait()

3) KO

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Semaphore(1)

barrier ← Semaphore(0)

- Pridajme blokovanie vlakien pred vstupom do ‘KO’

Bariera

Vlakno XYZ:

1) rendezvous

2) barrier.wait()

3) KO

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Semaphore(1)

barrier ← Semaphore(0)

- Velmi sme si nepomohli, vsetky vlakna ostavaju blokovane...

Bariera

Vlakno XYZ:

1) rendezvous

2) barrier.wait()

3) KO

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Semaphore(1)

barrier ← Semaphore(0)

- Pridajme pocitadlo vlakien, ktore prechadzaju z ‘rendezvous’ do ‘KO’

Bariera

Vlakno XYZ:

1) rendezvous

2) count += 1

3) barrier.wait()

4) KO

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Semaphore(1)

barrier ← Semaphore(0)

Bariera

Vlakno XYZ:

1) rendezvous

2) count += 1

3) barrier.wait()

4) KO

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Semaphore(1)

barrier ← Semaphore(0)

- Pocitadlo ale NIE je chranene proti sucasnemu pristupu vlakien!!!

Bariera

Vlakno XYZ:

1) rendezvous

2) count += 1

3) barrier.wait()

4) KO

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Semaphore(1)

barrier ← Semaphore(0)

- Pridajme teda mutex, ktory nam zaruci atomicitu operacie

Bariera

Vlakno XYZ:

1) rendezvous
2) mutex.wait()
3) count += 1
4) mutex.signal()

5) barrier.wait()
6) KO

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Semaphore(1)

barrier ← Semaphore(0)

- Pocitadlo je chranene proti sucasnemu pristupu vlakien
- Co s tym? Stale mame zablokovane vlakna...

Bariera

Vlakno XYZ:

1) rendezvous
2) mutex.wait()
3) count += 1
4) mutex.signal()
5) barrier.wait()
6) KO

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Semaphore(1)

barrier ← Semaphore(0)

- Posledne (N-te) vlakno uvolni barieru!

Bariera

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Semaphore(1)

barrier ← Semaphore(0)

Vlakno XYZ:

1) rendezvous
2) mutex.wait()
3) count += 1
4) mutex.signal()
5) if count == N: barrier.signal()
6) barrier.wait()
7) KO

Bariera

Vlakno XYZ:

1) rendezvous
2) mutex.wait()
3) count += 1
4) mutex.signal()
5) if count == N: barrie
6) barrier.wait()
7) KO

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Semaphore(1)

barrier ← Semaphore(0)

- Testovanie premennej mimo uzamknutia
- Problem?

Bariera

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Semaphore(1)

barrier ← Semaphore(0)

Vlakno XYZ:

1) rendezvous
2) mutex.wait()
3) count += 1
4) mutex.signal()
5) if count == N: barrier.signal()
6) barrier.wait()
7) KO

NIE, ALE...

Bariera

Vlakno XYZ:

1) rendezvous
2) mutex.wait()
3) count += 1
4) mutex.signal()
5) if count == N: barrier.signal()
6) barrier.wait()
7) KO

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Semaphore(1)

barrier ← Semaphore(0)

PRECO???

Bariera

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Semaphore(1)

barrier ← Semaphore(0)

Vlakno XYZ:

1) rendezvous
2) mutex.wait()
3) count += 1
4) mutex.signal()
5) if count == N: barrier.signal()
6) barrier.wait()
7) KO

- Uvolni sa nam aspoň 1 vlakno, ale nie N!
- Aspoň 1x sa vola signal() nad barierou

Bariera

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Semaphore(1)

barrier ← Semaphore(0)

Vlakno XYZ:

1) rendezvous
2) mutex.wait()
3) count += 1
4) mutex.signal()
5) if count == N: barrier.signal()
6) barrier.wait()
7) KO

- Uvolni sa nam aspoň 1 vlakno, ale nie N!
- Aspoň 1x sa vola signal() nad barierou

Bariera

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Semaphore(1)

barrier ← Semaphore(0)

Vlakno XYZ:

1) rendezvous
2) mutex.wait()
3) count += 1
4) mutex.signal()
5) if count == N: barrier.signal()
6) barrier.wait()
7) KO

- Nech kazde vlakno po prechode barierou uvolni barieru dalsiemu vlaknu

Bariera

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Semaphore(1)

barrier ← Semaphore(0)

Vlakno XYZ:

1) rendezvous
2) mutex.wait()
3) count += 1
4) mutex.signal()
5) if count == N: barrier.signal()
6) barrier.wait()
7) barrier.signal()
8) KO

- Nech kazde vlakno po prechode barierou uvolni barieru dalsiemu vlaknu

Bariera

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Semaphore(1)

barrier ← Semaphore(0)

Vlakno XYZ:

1) rendezvous
2) mutex.wait()
3) count += 1
4) mutex.signal()
5) if count == N: barrier.signal()
6) barrier.wait()
7) barrier.signal()
8) KO

- Nech kazde vlakno po prechode barierou uvolni barieru dalsiemu vlaknu

Bariera

Vlakno XYZ:

1) rendezvous
2) mutex.wait()
3) count += 1
4) mutex.signal()
5) if count == N: barrie
6) barrier.wait()
7) barrier.signal()
8) KO

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Semaphore(1)

barrier ← Semaphore(0)

- Synchronizacny vzor
- Turniket (turnstile)
- 1 vlakno v 1 case

Bariera

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Semaphore(1)

barrier ← Semaphore(0)

Vlakno XYZ:

1) rendezvous
2) mutex.wait()
3) count += 1
4) mutex.signal()
5) if count == N: barrier.signal()
6) barrier.wait()
7) barrier.signal()
8) KO

- Bude nasledovat ukazka velmi castej chyby

Bariera

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Semaphore(1)

barrier ← Semaphore(0)

Vlakno XYZ:

1) rendezvous
2) mutex.wait()
3) count += 1
4) if count == N: barrier.signal()
5) barrier.wait()
6) barrier.signal()
7) mutex.signal()
8) KO

- O aku chybu sa jedna?

Bariera

Vlakno XYZ:

1) rendezvous
2) mutex.wait()
3) count += 1
4) if count == N: barrier.signal()
5) barrier.wait()
6) barrier.signal()
7) mutex.signal()
8) KO

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Semaphore(1)

barrier ← Semaphore(0)

- Casta chyba!!!
- Blokovanie na semafore pocas drzania zamku!

Bariera

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Semaphore(1)

barrier ← Semaphore(0)

Vlakno XYZ:

1) rendezvous
2) mutex.wait()
3) count += 1
4) mutex.signal()
5) if count == N: barrier.signal()
6) barrier.wait()
7) barrier.signal()
8) KO

- Takze zatial finalna verzia jeden krat pouzitelnej bariery

Bariera

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Semaphore(1)

barrier ← Semaphore(0)

Vlakno XYZ:

1) rendezvous
2) mutex.wait()
3) count += 1
4) mutex.signal()
5) if count == N: barrier.signal()
6) barrier.wait()
7) barrier.signal()
8) KO

- Kolko krat sa vyvola signal()?
- Minimalne N-krat!

6. Znovupouzitelna bariera

- Kazde vlakno vykonava nejaky algoritmus v cykle
- Chceme, aby sa napriklad vsetky vlakna pockali vzdy PRED kazdou iteraciou

Vlakno XYZ:

1) while True:
2) rendezvous
3) KO

Vlakno XYZ:

```
1) while True:
2)     rendezvous
3)     mutex.lock()
4)         count += 1
5)     mutex.unlock()
6)     if count == N: turnstile.signal()
7)     turnstile.wait()
8)     turnstile.signal()
9)     KO
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile ← Semaphore(0)

- Rozsirme teda nase riesenie bariery
- Dajme ho do cyklu

Vlakno XYZ:

```
1) while True:
2)     rendezvous
3)     mutex.lock()
4)         count += 1
5)     mutex.unlock()
6)     if count == N: turnstile.signal()
7)     turnstile.wait()
8)     turnstile.signal()
9)     KO
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile ← Semaphore(0)

GOOOD?

Vlakno XYZ:

```
1) while True:
2)     rendezvous
3)     mutex.lock()
4)         count += 1
5)     mutex.unlock()
6)     if count == N: turnstile.signal()
7)     turnstile.wait()
8)     turnstile.signal()
9)     KO
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile ← Semaphore(0)

BAAAD!!!

Vlakno XYZ:

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)      mutex.unlock()
6)      if count == N: turnstile.signal()
7)      turnstile.wait()
8)      turnstile.signal()
9)      KO
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile ← Semaphore(0)

BAAAD!!!

Vlakno XYZ:

```
1) while True:
2)     rendezvous
3)     mutex.lock()
4)         count += 1
5)     mutex.unlock()
6)     if count == N: turnstile.signal()
7)     turnstile.wait()
8)     turnstile.signal()
9)     KO
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile ← Semaphore(0)

- Nevieme, aka je hodnota turnstile na konci cyklu
- Moze byt <1; N>
- Skusme chybu opravit

Vlakno XYZ:

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N: turnstile.signal()
6)      mutex.unlock()
7)      turnstile.wait()
8)      turnstile.signal()
9)      KO
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile ← Semaphore(0)

- Teraz vieme isto, ze hodnota turnstile bude po prechode vsetkych vlakien na konci cyklu 1

Vlakno XYZ:

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N: turnstile.signal()
6)      mutex.unlock()
7)      turnstile.wait()
8)      turnstile.signal()
9)      KO
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile ← Semaphore(0)

GOOOD!

Vlakno XYZ:

```
1) while True:
2)     rendezvous
3)     mutex.lock()
4)         count += 1
5)         if count == N: turnstile.signal()
6)     mutex.unlock()
7)     turnstile.wait()
8)     turnstile.signal()
9)     KO
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile ← Semaphore(0)

- Ako sa spravaju vlakna v cykle?

Vlakno XYZ:

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N: turnstile.signal()
6)      mutex.unlock()
7)      turnstile.wait()
8)      turnstile.signal()
9)      KO
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile ← Semaphore(0)

- Mame N volani wait()
- Mame N+1 volani signal()

Vlakno XYZ:

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N: turnstile.signal()
6)      mutex.unlock()
7)      turnstile.wait()
8)      turnstile.signal()
9)      KO
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile ← Semaphore(0)

- Ako to opravime?

Vlakno XYZ:

```
1) while True:
2)     rendezvous
3)     mutex.lock()
4)         count += 1
5)         if count == N: turnstile.signal()
6)     mutex.unlock()
7)     turnstile.wait()
8)     turnstile.signal()
9)     KO
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile ← Semaphore(0)

- Musi nam sediet pocet volani signal() a wait()
- Jedno vlakno nech robi wait() navyse
- Podobne ako jedno robi signal() navyse

Vlakno XYZ:

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N: turnstile.signal()
6)      mutex.unlock()
7)      turnstile.wait()
8)      turnstile.signal()
9)      KO
10)     if count == N: turnstile.wait()
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile ← Semaphore(0)

Vlakno XYZ:

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N: turnstile.signal()
6)      mutex.unlock()
7)      turnstile.wait()
8)      turnstile.signal()
9)      KO
10)     if count == N: turnstile.wait()
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile ← Semaphore(0)

- Aspon jedno vlakno robi wait()
- Ale potrebujeme prave jedno vlakno!

Vlakno XYZ:

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N: turnstile.signal()
6)      mutex.unlock()
7)      turnstile.wait()
8)      turnstile.signal()
9)      KO
10)     if count == N: turnstile.wait()
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile ← Semaphore(0)

- Ako to opravime?

Vlakno XYZ:

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N: turnstile.signal()
6)      mutex.unlock()
7)      turnstile.wait()
8)      turnstile.signal()
9)      KO
10)     mutex.lock()
11)         if count == N: turnstile.wait()
12)     mutex.unlock()
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile ← Semaphore(0)

Vlakno XYZ:

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N: turnstile.signal()
6)      mutex.unlock()
7)      turnstile.wait()
8)      turnstile.signal()
9)      KO
10)     mutex.lock()
11)         if count == N: turnstile.wait()
12)     mutex.unlock()
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile ← Semaphore(0)

PARADA!

Vlakno XYZ:

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N: turnstile.signal()
6)      mutex.unlock()
7)      turnstile.wait()
8)      turnstile.signal()
9)      KO
10)     mutex.lock()
11)         if count == N: turnstile.wait()
12)     mutex.unlock()
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile ← Semaphore(0)

Vlakno XYZ:

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N: turnstile.signal()
6)      mutex.unlock()
7)      turnstile.wait()
8)      turnstile.signal()
9)      KO
10)     mutex.lock()
11)         if count == N: turnstile.wait()
12)     mutex.unlock()
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile ← Semaphore(0)

- Funguje nam bariera?

Vlakno XYZ:

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N: turnstile.signal()
6)      mutex.unlock()
7)      turnstile.wait()
8)      turnstile.signal()
9)      KO
10)     mutex.lock()
11)         if count == N: turnstile.wait()
12)     mutex.unlock()
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile ← Semaphore(0)

- Kolko krat?

Vlakno XYZ:

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N: turnstile.signal()
6)      mutex.unlock()
7)      turnstile.wait()
8)      turnstile.signal()
9)      KO
10)     mutex.lock()
11)         if count == N: turnstile.wait()
12)     mutex.unlock()
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile ← Semaphore(0)

OPRAVTE

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N: turnstile.signal()
6)      mutex.unlock()
7)      turnstile.wait()
8)      turnstile.signal()
9)      KO
10)     mutex.lock()
11)         if count == N: turnstile.wait()


12)     mutex.unlock()
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile ← Semaphore(0)

- Opravme pocitadlo

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N: turnstile.signal()
6)      mutex.unlock()
7)      turnstile.wait()
8)      turnstile.signal()
9)      KO
10)     mutex.lock()
11)         if count == N: turnstile.wait()
12)         count -= 1
13)     mutex.unlock()
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile ← Semaphore(0)

- Opravme pocitadlo

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N: turnstile.signal()
6)      mutex.unlock()
7)      turnstile.wait()
8)      turnstile.signal()
9)      KO
10)     mutex.lock()
11)         if count == N: turnstile.wait()
12)         count -= 1
13)     mutex.unlock()
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile ← Semaphore(0)

STALE ZLE!

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N: turnstile.signal()
6)      mutex.unlock()
7)      turnstile.wait()
8)      turnstile.signal()
9)      KO
10)     mutex.lock()
11)         if count == N: turnstile.wait()
12)         count -= 1
13)     mutex.unlock()
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile ← Semaphore(0)

- Kod bezi v cykle!
- Po vykonani posledneho riadku kazde vlakno ide znovu na rande!

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N: turnstile.signal()
6)      mutex.unlock()
7)      turnstile.wait()
8)      turnstile.signal()
9)      KO
10)     mutex.lock()
11)         if count == N: turnstile.wait()
12)         count -= 1
13)     mutex.unlock()
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile ← Semaphore(0)

- Zatial mame 1 turniket
  - Potrebujeme ale 2!
- Prvy mame medzi R a KO
  - Druhy potrebujeme medzi KO a R

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N: turnstile1.signal()
6)      mutex.unlock()
7)      turnstile1.wait()
8)      turnstile1.signal()
9)      KO
10)     mutex.lock()
11)         if count == N: turnstile1.wait()
12)         count -= 1
13)     mutex.unlock()
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile1 ← Semaphore(0)

- Premenujeme prvy turniket

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N: turnstile1.signal()
6)      mutex.unlock()
7)      turnstile1.wait()
8)      turnstile1.signal()
9)      KO
10)     mutex.lock()
11)         if count == N: turnstile1.wait()
12)         count -= 1
13)     mutex.unlock()
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile1 ← Semaphore(0)

turnstile2 ← Semaphore(0)

- Pridame druhy turniket
  - turnstile2

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N: turnstile1.signal()
6)      mutex.unlock()
7)      turnstile1.wait()
8)      turnstile1.signal()
9)      KO
10)     mutex.lock()
11)         if count == N: turnstile1.wait()
12)         count -= 1
13)     mutex.unlock()
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile1 ← Semaphore(0)

turnstile2 ← Semaphore(0)

- Ako medzi R a KO

1. Vsetky vlakna cakaju za KO
2. Posledne vlakno spusta turniket

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N: turnstile1.signal()
6)      mutex.unlock()
7)      turnstile1.wait()
8)      turnstile1.signal()
9)      KO
10)     mutex.lock()
11)         if count == N: turnstile1.wait()
12)         count -= 1
13)     mutex.unlock()
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile1 ← Semaphore(0)

turnstile2 ← Semaphore(0)

- Vsetky vlakna cakaju za KO

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N: turnstile1.signal()
6)      mutex.unlock()
7)      turnstile1.wait()
8)      turnstile1.signal()
9)      KO
10)     mutex.lock()
11)         if count == N: turnstile1.wait()
12)         count -= 1
13)     mutex.unlock()
14)     turnstile2.wait()
15)     turnstile2.signal()
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile1 ← Semaphore(0)

turnstile2 ← Semaphore(0)

- Vsetky vlakna cakaju za KO

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N: turnstile1.signal()
6)      mutex.unlock()
7)      turnstile1.wait()
8)      turnstile1.signal()
9)      KO
10)     mutex.lock()
11)         if count == N: turnstile1.wait()
12)         count -= 1

13)     mutex.unlock()
14)     turnstile2.wait()
15)     turnstile2.signal()
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile1 ← Semaphore(0)

turnstile2 ← Semaphore(0)

- Posledne vlakno ho otvara

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N: turnstile1.signal()
6)      mutex.unlock()
7)      turnstile1.wait()
8)      turnstile1.signal()
9)      KO
10)     mutex.lock()
11)         if count == N: turnstile1.wait()
12)         count -= 1
13)         if count == 0: turnstile2.signal()
14)     mutex.unlock()
15)     turnstile2.wait()
16)     turnstile2.signal()
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile1 ← Semaphore(0)

turnstile2 ← Semaphore(0)

- Posledne vlakno ho otvara

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N: turnstile1.signal()
6)      mutex.unlock()
7)      turnstile1.wait()
8)      turnstile1.signal()
9)      KO
10)     mutex.lock()
11)       if count == N: turnstile1.wait()
12)         count -= 1
13)       if count == 0: turnstile2.signal()
14)     mutex.unlock()
15)     turnstile2.wait()
16)     turnstile2.signal()
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile1 ← Semaphore(0)

turnstile2 ← Semaphore(0)

- Musime mat dva rozne testy?

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N: turnstile1.signal()
6)      mutex.unlock()
7)      turnstile1.wait()
8)      turnstile1.signal()
9)      KO
10)     mutex.lock()
11)         if count == N: turnstile1.wait()
12)         count -= 1
13)         if count == 0: turnstile2.signal()
14)     mutex.unlock()
15)     turnstile2.wait()
16)     turnstile2.signal()
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile1 ← Semaphore(0)

turnstile2 ← Semaphore(0)

- Teraz robi wait() prve vlakno, ktore sa sem dostane
- Moze ho vykonat aj posledne vlakno?

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N: turnstile1.signal()
6)      mutex.unlock()
7)      turnstile1.wait()
8)      turnstile1.signal()
9)      KO
10)     mutex.lock()
11)         count -= 1
12)         if count == 0:
13)             turnstile1.wait()
14)             turnstile2.signal()
15)     mutex.unlock()
16)     turnstile2.wait()
17)     turnstile2.signal()
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile1 ← Semaphore(0)

turnstile2 ← Semaphore(0)

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N: turnstile1.signal()
6)      mutex.unlock()
7)      turnstile1.wait()
8)      turnstile1.signal()
9)      KO
10)     mutex.lock()
11)         count -= 1
12)         if count == 0:
13)             turnstile1.wait()
14)             turnstile2.signal()
15)     mutex.unlock()
16)     turnstile2.wait()
17)     turnstile2.signal()
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile1 ← Semaphore(0)

turnstile2 ← Semaphore(0)

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N: turnstile1.signal()
6)      mutex.unlock()
7)      turnstile1.wait()
8)      turnstile1.signal()
9)      KO
10)     mutex.lock()
11)         count -= 1
12)         if count == 0:
13)             turnstile1.wait()
14)             turnstile2.signal()
15)     mutex.unlock()
16)     turnstile2.wait()
17)     turnstile2.signal()
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile1 ← Semaphore(0)

turnstile2 ← Semaphore(0)

- Zalezi na poradi wait() a signal() roznych turniketov?

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N: turnstile1.signal()
6)      mutex.unlock()
7)      turnstile1.wait()
8)      turnstile1.signal()
9)      KO
10)     mutex.lock()
11)         count -= 1
12)         if count == 0:
13)             turnstile2.signal()
14)         turnstile1.wait()
15)     mutex.unlock()
16)     turnstile2.wait()
17)     turnstile2.signal()
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile1 ← Semaphore(0)

turnstile2 ← Semaphore(0)

- Zalezi na poradi wait() a signal() roznych turniketov?
- Co keby sme to urobili takto?

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N: turnstile1.signal()
6)      mutex.unlock()
7)      turnstile1.wait()
8)      turnstile1.signal()
9)      KO
10)     mutex.lock()
11)         count -= 1
12)         if count == 0:
13)             turnstile2.signal()
14)             turnstile1.wait()
15)     mutex.unlock()
16)     turnstile2.wait()
17)     turnstile2.signal()
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile1 ← Semaphore(0)

turnstile2 ← Semaphore(0)

- Co keby sme nepouzivali iba jeden mutex, ale dva?
- Pre kazdy turniket zvlast?

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N: turnstile1.signal()
6)      mutex.unlock()
7)      turnstile1.wait()
8)      turnstile1.signal()
9)      KO
10)     mutex.lock()
11)         count -= 1
12)         if count == 0:
13)             turnstile1.wait()
14)             turnstile2.signal()
15)     mutex.unlock()
16)     turnstile2.wait()
17)     turnstile2.signal()
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile1 ← Semaphore(0)

turnstile2 ← Semaphore(0)

- A co takto?
- Je to v pohode aj pri pouziti vlastneho mutexu pre kazdy turniket?

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N: turnstile1.signal()
6)      mutex.unlock()
7)      turnstile1.wait()
8)      turnstile1.signal()
9)      KO
10)     mutex.lock()
11)         count -= 1
12)         if count == 0:
13)             turnstile1.wait()
14)             turnstile2.signal()
15)     mutex.unlock()
16)     turnstile2.wait()
17)     turnstile2.signal()
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile1 ← Semaphore(0)

turnstile2 ← Semaphore(0)

- A co takto?
- Je to v pohode aj pri pouziti vlastneho mutexu pre kazdy turniket?
- Co pocitadlo? Stale pohodicka?

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N: turnstile1.signal()
6)      mutex.unlock()
7)      turnstile1.wait()
8)      turnstile1.signal()
9)      KO
10)     mutex.lock()
11)         count -= 1
12)         if count == 0:
13)             turnstile1.wait()
14)             turnstile2.signal()
15)     mutex.unlock()
16)     turnstile2.wait()
17)     turnstile2.signal()
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile1 ← Semaphore(0)

turnstile2 ← Semaphore(0)

- Riesenie vyzaduje symetriu!

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N: turnstile1.signal()
6)      mutex.unlock()
7)      turnstile1.wait()
8)      turnstile1.signal()
9)      KO
10)     mutex.lock()
11)         count -= 1
12)         if count == 0:
13)             turnstile1.wait()
14)             turnstile2.signal()
15)     mutex.unlock()
16)     turnstile2.wait()
17)     turnstile2.signal()
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile1 ← Semaphore(0)

turnstile2 ← Semaphore(0)

- Riesenie vyzaduje symetriu!
- Znovu mame problem, ze #signal() a #wait() pri turnstile2 sa nezhoduju!

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N:

6)              turnstile1.signal()
7)      mutex.unlock()
8)      turnstile1.wait()
9)      turnstile1.signal()
10)     KO
11)     mutex.lock()
12)         count -= 1
13)         if count == 0:
14)             turnstile1.wait()
15)             turnstile2.signal()
16)     mutex.unlock()
17)     turnstile2.wait()
18)     turnstile2.signal()
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile1 ← Semaphore(0)

turnstile2 ← Semaphore(0)

- Riesenie vyzaduje symetriu!
- Znovu mame problem, ze #signal() a #wait() pri turnstile2 sa nezhoduju!

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N:
6)              turnstile2.wait()
7)              turnstile1.signal()
8)      mutex.unlock()
9)      turnstile1.wait()
10)     turnstile1.signal()
11)     KO
12)     mutex.lock()
13)         count -= 1
14)         if count == 0:
15)             turnstile1.wait()
16)             turnstile2.signal()
17)     mutex.unlock()
18)     turnstile2.wait()
19)     turnstile2.signal()
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile1 ← Semaphore(0)

turnstile2 ← Semaphore(0)

- Pridanim wait() v jednom vlakne sa pocet volani signal() vyrovna

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N:
6)              turnstile2.wait()
7)              turnstile1.signal()
8)      mutex.unlock()
9)      turnstile1.wait()
10)     turnstile1.signal()
11)     KO
12)     mutex.lock()
13)         count -= 1
14)         if count == 0:
15)             turnstile1.wait()
16)             turnstile2.signal()
17)     mutex.unlock()
18)     turnstile2.wait()
19)     turnstile2.signal()
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile1 ← Semaphore(0)

turnstile2 ← Semaphore(0)

- Tesili sme sa, ze uz to mame...
- Blizime sa ku cifre 2^7 obrazkov prezentacie...

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N:
6)              turnstile2.wait()
7)              turnstile1.signal()
8)      mutex.unlock()
9)      turnstile1.wait()
10)     turnstile1.signal()
11)     KO
12)     mutex.lock()
13)         count -= 1
14)         if count == 0:
15)             turnstile1.wait()
16)             turnstile2.signal()
17)     mutex.unlock()
18)     turnstile2.wait()
19)     turnstile2.signal()
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile1 ← Semaphore(0)

turnstile2 ← Semaphore(0)

- Tesili sme sa, ze uz to mame...
- Blizime sa ku cifre 2^7 obrazkov prezentacie...

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N:
6)              turnstile2.wait()
7)              turnstile1.signal()
8)      mutex.unlock()
9)      turnstile1.wait()
10)     turnstile1.signal()
11)     KO
12)     mutex.lock()
13)         count -= 1
14)         if count == 0:
15)             turnstile1.wait()
16)             turnstile2.signal()
17)     mutex.unlock()
18)     turnstile2.wait()
19)     turnstile2.signal()
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile1 ← Semaphore(0)

turnstile2 ← Semaphore(0)

- Pridali sme 1 wait() do riesenia...

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N:
6)              turnstile2.wait()
7)              turnstile1.signal()
8)      mutex.unlock()
9)      turnstile1.wait()
10)     turnstile1.signal()
11)     KO
12)     mutex.lock()
13)         count -= 1
14)         if count == 0:
15)             turnstile1.wait()
16)             turnstile2.signal()
17)     mutex.unlock()
18)     turnstile2.wait()
19)     turnstile1.signal()
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile1 ← Semaphore(0)

turnstile2 ← Semaphore(0)

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N:
6)              turnstile2.wait()
7)              turnstile1.signal()
8)      mutex.unlock()
9)      turnstile1.wait()
10)     turnstile1.signal()
11)     KO
12)     mutex.lock()
13)         count -= 1
14)         if count == 0:
15)             turnstile1.wait()
16)             turnstile2.signal()
17)     mutex.unlock()
18)     turnstile2.wait()
19)     turnstile2.signal()
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile1 ← Semaphore(0)

turnstile2 ← Semaphore(0)

- Ako je inicializovany turnstile2?

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N:
6)              turnstile2.wait()
7)              turnstile1.signal()
8)      mutex.unlock()
9)      turnstile1.wait()
10)     turnstile1.signal()
11)     KO
12)     mutex.lock()
13)         count -= 1
14)         if count == 0:
15)             turnstile1.wait()
16)             turnstile2.signal()
17)     mutex.unlock()
18)     turnstile2.wait()
19)     turnstile2.signal()
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile1 ← Semaphore(0)

turnstile2 ← Semaphore(0)

- Ako je inicializovany turnstile2?

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N:
6)              turnstile2.wait()
7)              turnstile1.signal()
8)      mutex.unlock()
9)      turnstile1.wait()
10)     turnstile1.signal()
11)     KO
12)     mutex.lock()
13)         count -= 1
14)         if count == 0:
15)             turnstile1.wait()
16)             turnstile2.signal()
17)     mutex.unlock()
18)     turnstile2.wait()
19)     turnstile2.signal()
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile1 ← Semaphore(0)

turnstile2 ← Semaphore(1)

- Oprava...
- Moze to tak byt?

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N:
6)              turnstile2.wait()
7)              turnstile1.signal()
8)      mutex.unlock()
9)      turnstile1.wait()
10)     turnstile1.signal()
11)     KO
12)     mutex.lock()
13)         count -= 1
14)         if count == 0:
15)             turnstile1.wait()
16)             turnstile2.signal()
17)     mutex.unlock()
18)     turnstile2.wait()
19)     turnstile2.signal()
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile1 ← Semaphore(0)

turnstile2 ← Semaphore(1)

Dvojfazova bariera

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N:
6)              turnstile2.wait()
7)              turnstile1.signal()
8)      mutex.unlock()
9)      turnstile1.wait()
10)     turnstile1.signal()
11)     KO
12)     mutex.lock()
13)         count -= 1
14)         if count == 0:
15)             turnstile1.wait()
16)             turnstile2.signal()
17)     mutex.unlock()
18)     turnstile2.wait()
19)     turnstile2.signal()
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile1 ← Semaphore(0)

turnstile2 ← Semaphore(1)

- Iba n-te vlakno moze zamykat/odomykat turnikety
- Pred odomknutim vlakno vzdy zamkne druhy turniket, takze nejake vlakno moze predbehnut ine iba o 1 turniket!

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N:
6)              turnstile2.wait()
7)              turnstile1.signal()
8)      mutex.unlock()
9)      turnstile1.wait()
10)     turnstile1.signal()
11)     KO
12)     mutex.lock()
13)         count -= 1
14)         if count == 0:
15)             turnstile1.wait()
16)             turnstile2.signal()
17)     mutex.unlock()
18)     turnstile2.wait()
19)     turnstile2.signal()
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile1 ← Semaphore(0)

turnstile2 ← Semaphore(1)

- Kod bariery sa da zjednodusit
- Ak mame moznost zvysit hodnotu semaforu nie iba o 1, ale o N

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N:

6)              turnstile1.signal(N)
7)      mutex.unlock()
8)      turnstile1.wait()

9)      KO
10)     mutex.lock()
11)         count -= 1
12)         if count == 0:

13)             turnstile2.signal(N)
14)     mutex.unlock()
15)     turnstile2.wait()
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile1 ← Semaphore(0)

turnstile2 ← Semaphore(1)

- Kod bariery sa da zjednodusit
- Ak mame moznost zvysit hodnotu semaforu nie iba o 1, ale o N

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N:

6)              turnstile1.signal(N)
7)      mutex.unlock()
8)      turnstile1.wait()

9)      KO
10)     mutex.lock()
11)         count -= 1
12)         if count == 0:

13)             turnstile2.signal(N)
14)     mutex.unlock()
15)     turnstile2.wait()
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile1 ← Semaphore(0)

turnstile2 ← Semaphore(1)

- Vsimnime si riadky 3-8 a 10-15
- Vzhladom na turniket su uplne nezavisle!
- Tuto skutocnost o chvilku vyuzijeme...

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N:
6)              turnstile1.signal(N)
7)      mutex.unlock()
8)      turnstile1.wait()
9)      KO
10)     mutex.lock()
11)         count -= 1
12)         if count == 0:
13)             turnstile2.signal(N)
14)     mutex.unlock()
15)     turnstile2.wait()
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile1 ← Semaphore(0)

turnstile2 ← Semaphore(1)

- Zvysenim hodnoty semaforu o N miesto o 1 usetrime 4 riadky kodu

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N:
6)              turnstile1.signal(N)
7)      mutex.unlock()
8)      turnstile1.wait()
9)      KO
10)     mutex.lock()
11)         count += 1
12)         if count == 0:
13)             turnstile2.signal(N)
14)     mutex.unlock()
15)     turnstile2.wait()
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile1 ← Semaphore(0)

turnstile2 ← Semaphore(1)

SETRIME ZLE!

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N:
6)              turnstile1.signal(N)
7)      mutex.unlock()
8)      turnstile1.wait()
9)      KO
10)     mutex.lock()
11)         count -= 1
12)         if count == 0:
13)             turnstile2.signal(N)
14)     mutex.unlock()
15)     turnstile2.wait()
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile1 ← Semaphore(0)

turnstile2 ← Semaphore(1)

- Kazda zmena kodu vyzaduje MAXIMALNE sustredenie a pozornost!
- Po zmene kodu kontrola inicializacie a opacne!

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N:
6)              turnstile1.signal(N)
7)      mutex.unlock()
8)      turnstile1.wait()
9)      KO
10)     mutex.lock()
11)         count -= 1
12)         if count == 0:
13)             turnstile2.signal(N)
14)     mutex.unlock()
15)     turnstile2.wait()
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile1 ← Semaphore(0)

turnstile2 ← Semaphore(0)

- Kazda zmena kodu vyzaduje MAXIMALNE sustredenie a pozornost!
- Po zmene kodu kontrola inicializacie a opacne!

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N:
6)              turnstile1.signal(N)
7)      mutex.unlock()
8)      turnstile1.wait()
9)      KO
10)     mutex.lock()
11)         count -= 1
12)         if count == 0:
13)             turnstile2.signal(N)
14)     mutex.unlock()
15)     turnstile2.wait()
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile1 ← Semaphore(0)

turnstile2 ← Semaphore(0)

```
1)  while True:
2)      rendezvous
3)      mutex.lock()
4)          count += 1
5)          if count == N:
6)              turnstile1.signal(N)
7)      mutex.unlock()
8)      turnstile1.wait()
9)      KO
10)     mutex.lock()
11)         count -= 1
12)         if count == 0:
13)             turnstile2.signal(N)
14)     mutex.unlock()
15)     turnstile2.wait()
```

Inicializacia:

N – dane a dostupne

count ← 0

mutex ← Mutex()

turnstile1 ← Semaphore(0)

turnstile2 ← Semaphore(0)

‘Nabitie’ turniketu

7. ADT SimpleBarrier

- Na zaciatku turniket zablokovany
- N-te vlakno turniket odblokuje
- N vlakien moze prejst turniketom

- Stav turniketu:
  - pocet prechadzajucich N, dane pri vytvarani ADT
  - mutex M, pocitadlo C
  - Semafor turniketu T

Init(N):

C ← 0

M ← Mutex()

T ← Semaphore(0)

```
1)  M.lock()
2)      C += 1
3)      if C == N:
4)          C = 0
5)          T.signal(N)
6)  M.unlock()
7)  T.wait()
```

Init(N):

C ← 0

M ← Mutex()

T ← Semaphore(0)

```
1)  M.lock()
2)      C += 1
3)      if C == N:
4)          C = 0
5)          T.signal(N)
6)  M.unlock()
7)  T.wait()
```

Init(N):

C ← 0

M ← Mutex()

T ← Semaphore(0)

```
1)  M.lock()
2)      C += 1
3)      if C == N:
4)          C = 0
5)          T.signal(N)
6)  M.unlock()
7)  T.wait()
```

- Ak by sme nevynulovali pocitadlo, znovu upadneme do hroznej biedY!

Znovu pouzitelna bariera s ADT SimpleBarrier

1) while True:
2) rendezvous
3) barrier1()
4) KO
5) barrier2()

Init(N):

barrier1 ← SimpleBarrier(N)

barrier2 ← SimpleBarrier(N)

8. Cvicenie

- Definicia triedy v jazyku Python
- Implementacia roznych veci:
  - Jednoducha bariera
  - Znovu pouzitelna bariera
- Vyuzitie nasledovnych synchronizacnych vzorov v prikladoch:
  - Signalizacia, Rendezvous, (opakovana) Bariera

? Nejake otazockY ?

Dakujem za pozornost!