


PPaDS MMXXI

Matúš Jókay, C-503, matus.jokay@stuba.sk

uim.fei.stuba.sk/predmet/i-ppds
konzultácie elektronicky
mail, discord

- 
- Producent-Konzument
 - Vzájomné vylúčenie kategórii
 - Čítatelia-Zapisovatelia
 - Vypínač (Lightswitch)



Producent - konzument

- Veľmi častá schéma v aplikačnom programovaní
 - ▣ Obsluha spojení
 - ▣ Udalosťami riadený systém
- Zdieľaný objekt: sklad (buffer)

Producent - konzument

- Konzistencia skladu (vkladanie/vyberanie položiek)
- Ak je sklad prázdny, konzument musí čakať
- Ak je sklad plný, producent musí čakať

P-K

P():

K():

1) `item = produceItem()`

2) `buffer.add(item)`

1) `item = buffer.get()`

2) `item.process()`

P-K

Init():

mutex \leftarrow Mutex()

items \leftarrow Semaphore(0)

P():

1) item = produceItem()

2) buffer.add(item)

K():

1) item = buffer.get()

2) item.process()

P-K

Init():

`mutex` \leftarrow Mutex()

`items` \leftarrow Semaphore(0)

P():

1) `item = produceItem()`

2) `buffer.add(item)`

K():

1) `items.wait()`

2) `item = buffer.get()`

3) `item.process()`

P-K

Init():

`mutex` ← `Mutex()`

`items` ← `Semaphore(0)`

P():

1) `item = produceItem()`

2) `buffer.add(item)`

3) `items.signal()`

K():

1) `items.wait()`

2) `item = buffer.get()`

3) `item.process()`

P-K

Init():

`mutex` ← `Mutex()`

`items` ← `Semaphore(0)`

P():

- 1) `item = produceItem()`
- 2) `mutex.lock()`
- 3) `buffer.add(item)`
- 4) `items.signal()`

K():

- 1) `items.wait()`
- 2) `mutex.lock()`
- 3) `item = buffer.get()`
- 4) `item.process()`

P-K

Init():

`mutex` ← `Mutex()`

`items` ← `Semaphore(0)`

P():

- 1) `item = produceItem()`
- 2) `mutex.lock()`
- 3) `buffer.add(item)`
- 4) `mutex.unlock()`
- 5) `items.signal()`

K():

- 1) `items.wait()`
- 2) `mutex.lock()`
- 3) `item = buffer.get()`
- 4) `mutex.unlock()`
- 5) `item.process()`

P-K

Init():

```
mutex ← Mutex(), items ← Semaphore(0)  
free ← Semaphore(buffer.size())
```

P():

- 1) item = produceItem()
- 2) mutex.lock()
- 3) buffer.add(item)
- 4) mutex.unlock()
- 5) items.signal()

K():

- 1) items.wait()
- 2) mutex.lock()
- 3) item = buffer.get()
- 4) mutex.unlock()
- 5) item.process()

P-K

Init():

```
mutex ← Mutex(), items ← Semaphore(0)  
free ← Semaphore(buffer.size())
```

P():

- 1) item = produceItem()
- 2) mutex.lock()
- 3) buffer.add(item)
- 4) mutex.unlock()
- 5) items.signal()

- 3 možnosti, kam do P() vložit free.wait():
 - Pred R1
 - Medzi R1-R2
 - Medzi R2-R3
- Po R3 už nemá zmysel riešiť

P-K

Init():

`mutex` \leftarrow Mutex(), `items` \leftarrow Semaphore(0)

`free` \leftarrow Semaphore(buffer.size())

P():

- 1) `item = produceItem()`
- 2) `mutex.lock()`
- 3) `buffer.add(item)`
- 4) `mutex.unlock()`
- 5) `items.signal()`

- Koľko možností je správnych?
- Koľko je optimálnych?
- Ktoré to sú a prečo?

P-K

```
Init():  
mutex ← Mutex(), items ← Semaphore(0)  
free ← Semaphore(buffer.size())
```

P():

- 1) item = produceItem()
 - 2) mutex.lock()
 - 3) buffer.add(item)
 - 4) mutex.unlock()
 - 5) items.signal()
- Koľko možností je správnych?
 - Koľko je optimálnych?
 - Ktoré to sú a prečo?

P-K

Init():

```
mutex ← Mutex(), items ← Semaphore(0)  
free ← Semaphore(buffer.size())
```

P():

- 1) item = produceItem()
- 2) mutex.lock()
- 3) buffer.add(item)
- 4) mutex.unlock()
- 5) items.signal()

K():

- 1) items.wait()
- 2) mutex.lock()
- 3) item = buffer.get()
- 4) mutex.unlock()
- 5) item.process()

P-K

Init():

```
mutex ← Mutex(), items ← Semaphore(0)  
free ← Semaphore(buffer.size())
```

P():

- 1) item = produceItem()
- 2) free.wait()
- 3) mutex.lock()
- 4) buffer.add(item)
- 5) mutex.unlock()
- 6) items.signal()

K():

- 1) items.wait()
- 2) mutex.lock()
- 3) item = buffer.get()
- 4) mutex.unlock()
- 5) free.signal()
- 6) item.process()



Vylúčenie kategórii procesov

- Napríklad problém spoločného WC
 - ▣ WC s 3 kabínkami
 - ▣ Buď ho používajú muži alebo ženy, ale nie súčasne
- Problém čitateľov a zapisovateľov
 - ▣ Meniť údaje (zapisovať) môže iba 1, vtedy sa nesmie ani čítať
 - ▣ Čítať môžu viacerí, ale sa nemôže zapisovať

Vylúčenie kategórii procesov

- Semafor
 - ▣ Signalizuje obsadenosť oblasti (miestnosti)
 - ▣ Napr. ak je WC voľné, vojde žena a obsadí tak miestnosť (muži nemôžu vojsť); a opačne
- Ako zabezpečiť, aby iba jeden proces signalizoval obsadenie miestnosti?

Vylúčenie kategórii procesov

- Signalizácia obsadenosti
 - ▣ Kto prvý vchádza, signalizuje obsadenosť
 - ▣ Kto posledný vychádza, signalizuje voľnosť
 - ▣ Napr. ceduľka na WC

- Ako vieme, kto je prvý a kto posledný?

Vylúčenie kategórii procesov

- Ako zistíme, kto je prvý a kto je posledný?
 - ▣ Počítadlo!
 - ▣ Ale keď máme počítaadlo, a k nemu prístup viacerých vlákien...
- Stačí nám počítať prítomných v miestnosti, alebo potrebujeme vedieť odlíšiť kategóriu?

Problém čitatelia-zapisovateľa

```
Init():  
mutex ← Mutex()  
readers ← 0  
roomEmpty ← Semaphore(1)
```

R-W

Init():

`mutex` \leftarrow Mutex()

`readers` \leftarrow 0

`roomEmpty` \leftarrow Semaphore(1)

R-W

```
Init():  
  mutex ← Mutex()  
  readers ← 0  
  roomEmpty ← Semaphore(1)
```

R():

W():

- | | |
|-----|-----|
| 1) | 1) |
| 2) | 2) |
| 3) | 3) |
| 4) | 4) |
| 5) | 5) |
| 6) | 6) |
| 7) | 7) |
| 8) | 8) |
| 9) | 9) |
| 10) | 10) |
| 11) | 11) |
| 12) | 12) |
| 13) | 13) |

R-W

```
Init():  
  mutex ← Mutex()  
  readers ← 0  
  roomEmpty ← Semaphore(1)
```

R():

W():

- 1)
- 2)
- 3)
- 4)
- 5)
- 6)
- 7) // kod citatela
- 8)
- 9)
- 10)
- 11)
- 12)
- 13)

- 1)
- 2)
- 3)
- 4)
- 5)
- 6)
- 7) // kod zapisovatela
- 8)
- 9)
- 10)
- 11)
- 12)
- 13)

R-W

```
Init():  
  mutex ← Mutex()  
  readers ← 0  
  roomEmpty ← Semaphore(1)
```

R():

```
1)  
2)  
3)  
4)  
5)  
6)  
7) // kod citatela  
8)  
9)  
10)  
11)  
12)  
13)
```

W():

```
1)  
2)  
3)  
4) roomEmpty.wait()  
5)  
6)  
7) // kod zapisovatela  
8)  
9)  
10)  
11)  
12)  
13)
```

R-W

```
Init():  
  mutex ← Mutex()  
  readers ← 0  
  roomEmpty ← Semaphore(1)
```

R():

W():

```
1)  
2)  
3)  
4)  
5)  
6)  
7) // kod citatela  
8)  
9)  
10)  
11)  
12)  
13)
```

```
1)  
2)  
3)  
4) roomEmpty.wait()  
5)  
6)  
7) // kod zapisovatela  
8)  
9)  
10)  
11)  
12) roomEmpty.signal()  
13)
```

R-W

```
Init():  
  mutex ← Mutex()  
  readers ← 0  
  roomEmpty ← Semaphore(1)
```

R():

```
1)  
2)  
3)  
4)  roomEmpty.wait()  
5)  
6)  
7)  // kod citatela  
8)  
9)  
10)  
11)  
12)  
13)
```

W():

```
1)  
2)  
3)  
4)  roomEmpty.wait()  
5)  
6)  
7)  // kod zapisovatela  
8)  
9)  
10)  
11)  
12)  roomEmpty.signal()  
13)
```

R-W

```
Init():  
  mutex ← Mutex()  
  readers ← 0  
  roomEmpty ← Semaphore(1)
```

R():

```
1)  
2)  
3)  
4)  roomEmpty.wait()  
5)  
6)  
7)  // kod citatela  
8)  
9)  
10)  
11)  
12) roomEmpty.signal()  
13)
```

W():

```
1)  
2)  
3)  
4)  roomEmpty.wait()  
5)  
6)  
7)  // kod zapisovatela  
8)  
9)  
10)  
11)  
12) roomEmpty.signal()  
13)
```

R-W

```
Init():  
mutex ← Mutex()  
readers ← 0  
roomEmpty ← Semaphore(1)
```

R():

```
1)  
2)  
3) if readers == 1:  
4)   roomEmpty.wait()  
5)  
6)  
7)   // kod citatela  
8)  
9)  
10)  
11)  
12)   roomEmpty.signal()  
13)
```

W():

```
1)  
2)  
3)  
4)   roomEmpty.wait()  
5)  
6)  
7)   // kod zapisovatela  
8)  
9)  
10)  
11)  
12)   roomEmpty.signal()  
13)
```

R-W

```
Init():  
mutex ← Mutex()  
readers ← 0  
roomEmpty ← Semaphore(1)
```

R():

```
1)  
2)  
3) if readers == 1:  
4)     roomEmpty.wait()  
5)  
6)  
7)     // kod citatela  
8)  
9)  
10)  
11) if readers == 0:  
12)     roomEmpty.signal()  
13)
```

W():

```
1)  
2)  
3)  
4)     roomEmpty.wait()  
5)  
6)  
7)     // kod zapisovatela  
8)  
9)  
10)  
11)  
12)     roomEmpty.signal()  
13)
```


R-W

```
Init():  
    mutex ← Mutex()  
    readers ← 0  
    roomEmpty ← Semaphore(1)
```

R():

```
1)  
2)  readers += 1  
3)  if readers == 1:  
4)    roomEmpty.wait()  
5)  
6)  
7)    // kod citatela  
8)  
9)  
10)  
11) if readers == 0:  
12)   roomEmpty.signal()  
13)
```

W():

```
1)  
2)  
3)  
4)  roomEmpty.wait()  
5)  
6)  
7)  // kod zapisovatela  
8)  
9)  
10)  
11)  
12) roomEmpty.signal()  
13)
```

R-W

```
Init():  
    mutex ← Mutex()  
    readers ← 0  
    roomEmpty ← Semaphore(1)
```

R():

```
1)  
2)  readers += 1  
3)  if readers == 1:  
4)    roomEmpty.wait()  
5)  
6)  
7)    // kod citatela  
8)  
9)  
10) readers -= 1  
11) if readers == 0:  
12)    roomEmpty.signal()  
13)
```

W():

```
1)  
2)  
3)  
4)  roomEmpty.wait()  
5)  
6)  
7)  // kod zapisovatela  
8)  
9)  
10)  
11)  
12) roomEmpty.signal()  
13)
```

R-W

```
Init():  
    mutex ← Mutex()  
    readers ← 0  
    roomEmpty ← Semaphore(1)
```

R():

```
1) mutex.lock()  
2) readers += 1  
3) if readers == 1:  
4)     roomEmpty.wait()  
5) mutex.unlock()  
6)  
7)     // kod citatela  
8)  
9)  
10) readers -= 1  
11) if readers == 0:  
12)     roomEmpty.signal()  
13)
```

W():

```
1)  
2)  
3)  
4)     roomEmpty.wait()  
5)  
6)  
7)     // kod zapisovatela  
8)  
9)  
10)  
11)  
12)     roomEmpty.signal()  
13)
```

R-W

```
Init():  
    mutex ← Mutex()  
    readers ← 0  
    roomEmpty ← Semaphore(1)
```

R():

```
1) mutex.lock()  
2) readers += 1  
3) if readers == 1:  
4)     roomEmpty.wait()  
5) mutex.unlock()  
6)  
7)     // kod citatela  
8)  
9) mutex.lock()  
10) readers -= 1  
11) if readers == 0:  
12)     roomEmpty.signal()  
13) mutex.unlock()
```

W():

```
1)  
2)  
3)  
4)     roomEmpty.wait()  
5)  
6)  
7)     // kod zapisovatela  
8)  
9)  
10)  
11)  
12)     roomEmpty.signal()  
13)
```

R-W

W():

- 1) `roomEmpty.wait()`
- 2) `// kod zapisovatela`
- 3) `roomEmpty.signal()`

Init():

```
mutex ← Mutex()
readers ← 0
roomEmpty ← Semaphore(1)
```

R():

- 1) `mutex.lock()`
- 2) `readers += 1`
- 3) `if readers == 1:`
- 4) `roomEmpty.wait()`
- 5) `mutex.unlock()`
- 6)
- 7) `// kod citatela`
- 8)
- 9) `mutex.lock()`
- 10) `readers -= 1`
- 11) `if readers == 0:`
- 12) `roomEmpty.signal()`
- 13) `mutex.unlock()`

R-W

```
W():  
1) roomEmpty.wait()  
2) // kod zapisovatela  
3) roomEmpty.signal()
```

```
Init():  
mutex ← Mutex()  
readers ← 0  
roomEmpty ← Semaphore(1)
```

R():

```
1) mutex.lock()  
2) readers += 1  
3) if readers == 1:  
4)   roomEmpty.wait()  
5) mutex.unlock()  
6)  
7)   // kod citatela  
8)  
9) mutex.lock()  
10) readers -= 1  
11) if readers == 0:  
12)   roomEmpty.signal()  
13) mutex.unlock()
```

□ Vlastnosti riešenia

R-W

```
W():  
1) roomEmpty.wait()  
2) // kod zapisovateľa  
3) roomEmpty.signal()
```

```
Init():  
mutex ← Mutex()  
readers ← 0  
roomEmpty ← Semaphore(1)
```

R():

```
1) mutex.lock()  
2) readers += 1  
3) if readers == 1:  
4)   roomEmpty.wait()  
5) mutex.unlock()  
6)  
7) // kod citateľa  
8)  
9) mutex.lock()  
10) readers -= 1  
11) if readers == 0:  
12)   roomEmpty.signal()  
13) mutex.unlock()
```

□ Vlastnosti riešenia

□ Kto prvý vchádza, rozsvieti

□ Kto posledný odchádza, zhasne

□ Vypínač (lightswitch)

R-W

```
W():  
1) roomEmpty.wait()  
2) // kod zapisovateľa  
3) roomEmpty.signal()
```

```
Init():  
mutex ← Mutex()  
readers ← 0  
roomEmpty ← Semaphore(1)
```

R():

```
1) mutex.lock()  
2) readers += 1  
3) if readers == 1:  
4)   roomEmpty.wait()  
5) mutex.unlock()  
6)  
7) // kod citateľa  
8)  
9) mutex.lock()  
10) readers -= 1  
11) if readers == 0:  
12)   roomEmpty.signal()  
13) mutex.unlock()
```

□ Vlastnosti riešenia

□ Kto prvý vchádza, rozsvieti

□ Kto posledný odchádza, zhasne

□ Vypínač (lightswitch)

R-W

W():

```
1) roomEmpty.wait()
2) // kod zapisovateľa
3) roomEmpty.signal()
```

Init():

```
mutex ← Mutex()
readers ← 0
roomEmpty ← Semaphore(1)
```

R():

```
1) mutex.lock()
2) readers += 1
3) if readers == 1:
4)     roomEmpty.wait()
5) mutex.unlock()
6)
7)     // kod čitateľa
8)
9) mutex.lock()
10) readers -= 1
11) if readers == 0:
12)     roomEmpty.signal()
13) mutex.unlock()
```

□ Vlastnosti riešenia

- ▣ Iba jeden čitateľ čaká na semafore **roomEmpty**
- ▣ Viacerí zapisovatelia môžu čakať na semafore **roomEmpty**
- ▣ Miestnosť je prázdna, keď čitateľ vyvoláva **roomEmpty.signal()**

R-W

```
W():  
1) roomEmpty.wait()  
2) // kod zapisovatela  
3) roomEmpty.signal()
```

```
Init():  
mutex ← Mutex()  
readers ← 0  
roomEmpty ← Semaphore(1)
```

R():

```
1) mutex.lock()  
2) readers += 1  
3) if readers == 1:  
4) roomEmpty.wait()  
5) mutex.unlock()  
6)  
7) // kod citatela  
8)  
9) mutex.lock()  
10) readers -= 1  
11) if readers == 0:  
12) roomEmpty.signal()  
13) mutex.unlock()
```

□ Vlastnosti riešenia

- ▣ Iba jeden čitateľ čaká na semafore `roomEmpty`
- ▣ Viacerí zapisovatelia môžu čakať na semafore `roomEmpty`
- ▣ Miestnosť je prázdna, keď čitateľ vyvoláva `roomEmpty.signal()`

ADT Lightswitch

- Interný stav
 - ▣ mutex
 - ▣ počítadlo
- Metódy
 - ▣ lock(sem)
 - ▣ unlock(sem)

ADT Lightswitch

- Interný stav
 - ▣ mutex
 - ▣ počítadlo
- Metódy
 - ▣ lock(sem)
 - ▣ unlock(sem)

```
Init():  
mutex ← Mutex()  
counter ← 0
```

ADT Lightswitch

Init():

mutex \leftarrow Mutex()

counter \leftarrow 0

ADT Lightswitch

Init():

mutex \leftarrow Mutex()

counter \leftarrow 0

lock(**sem**):

unlock(**sem**):

ADT Lightswitch

Init():

mutex \leftarrow Mutex()

counter \leftarrow 0

lock(**sem**):

- 1)
- 2)
- 3)
- 4)
- 5)

unlock(**sem**):

- 1)
- 2)
- 3)
- 4)
- 5)

ADT Lightswitch

Init():

mutex \leftarrow Mutex()

counter \leftarrow 0

lock(**sem**):

1)

2)

3)

4) **sem.wait()**

5)

unlock(**sem**):

1)

2)

3)

4) **sem.signal()**

5)

ADT Lightswitch

Init():

`mutex` \leftarrow `Mutex()`

`counter` \leftarrow 0

`lock(sem):`

- 1)
- 2)
- 3) if `counter` == 1:
- 4) `sem.wait()`
- 5)

`unlock(sem):`

- 1)
- 2)
- 3) if `counter` == 0:
- 4) `sem.signal()`
- 5)

ADT Lightswitch

Init():

mutex \leftarrow Mutex()

counter \leftarrow 0

lock(**sem**):

- 1)
- 2) counter += 1
- 3) if counter == 1:
- 4) **sem**.wait()
- 5)

unlock(**sem**):

- 1)
- 2) counter -= 1
- 3) if counter == 0:
- 4) **sem**.signal()
- 5)

ADT Lightswitch

Init():

`mutex` \leftarrow `Mutex()`

`counter` \leftarrow 0

`lock(sem):`

- 1) `mutex.lock()`
- 2) `counter += 1`
- 3) if `counter == 1`:
- 4) `sem.wait()`
- 5) `mutex.unlock()`

`unlock(sem):`

- 1) `mutex.lock()`
- 2) `counter -= 1`
- 3) if `counter == 0`:
- 4) `sem.signal()`
- 5) `mutex.unlock()`

R-W

```
W():  
1) roomEmpty.wait()  
2) // kod zapisovatela  
3) roomEmpty.signal()
```

```
Init():  
mutex ← Mutex()  
readers ← 0  
roomEmpty ← Semaphore(1)
```

R():

```
1) mutex.lock()  
2) readers += 1  
3) if readers == 1:  
4)   roomEmpty.wait()  
5) mutex.unlock()  
6)  
7)   // kod citatela  
8)  
9) mutex.lock()  
10) readers -= 1  
11) if readers == 0:  
12)   roomEmpty.signal()  
13) mutex.unlock()
```

□ Prepis pomocou ADT Lightswitch

R-W

```
W():  
1) roomEmpty.wait()  
2) // kod zapisovatela  
3) roomEmpty.signal()
```

```
Init():  
mutex ← Mutex()  
readers ← 0  
roomEmpty ← Semaphore(1)
```

R():

```
1) mutex.lock()  
2) readers += 1  
3) if readers == 1:  
4)   roomEmpty.wait()  
5) mutex.unlock()  
6)  
7)   // kod citatela  
8)  
9) mutex.lock()  
10) readers -= 1  
11) if readers == 0:  
12)   roomEmpty.signal()  
13) mutex.unlock()
```

lock(roomEmpty)

unlock(roomEmpty)

R-W

```
W():  
1) roomEmpty.wait()  
2) // kod zapisovatela  
3) roomEmpty.signal()
```

```
Init():  
mutex ← Mutex()  
readers ← 0  
roomEmpty ← Semaphore(1)
```

R():

```
1) mutex.lock()  
2) readers += 1  
3) if readers == 1:  
4)   roomEmpty.wait()  
5) mutex.unlock()
```



lock(roomEmpty)

```
6)  
7) // kod citatela
```

```
8)  
9) mutex.lock()  
10) readers -= 1  
11) if readers == 0:  
12)   roomEmpty.signal()  
13) mutex.unlock()
```



unlock(roomEmpty)

R-W pomocou LS

Init():

readLS \leftarrow LS()

roomE \leftarrow Sem(1)

R-W pomocou LS

Init():
readLS \leftarrow LS()
roomE \leftarrow Sem(1)

R():

1)

2)

3)

W():

1)

2)

3)

R-W pomocou LS

Init():
readLS \leftarrow LS()
roomE \leftarrow Sem(1)

R():

1)

2) // kod citat.

3)

W():

1)

2) // kod zapis.

3)

R-W pomocou LS

```
Init():  
  readLS ← LS()  
  roomE ← Sem(1)
```

R():

- 1) readLS.lock(roomE)
- 2) // kod citat.
- 3) readLS.unlock(roomE)

W():

- 1) roomE.wait()
- 2) // kod zapis.
- 3) roomE.signal()

Vyhľadovenie (Starvation)



Vyhladovenie (Starvation)

- Nie je to uviaznutie (deadlock)
- Problém, keď sa nejaký proces nikdy “nedostane k slovu”
 - Napr. WC s 3 misami v krčme
 - 2 ženy, každá vypije 1 kávu za 2 hodiny
 - 37 chlapov, každý vypije 5-13 piv za 10 min ;)

Vyhladovenie (Starvation)

- Veľmi často nastáva pri riešení synchronizačných problémov, kde sa využíva LS
- Napríklad aj v R-W môže byť problém, v závislosti od konfigurácie modelu
 - ▣ Rádovy nepomer vlákien
 - ▣ Rôzna doba čítania a zápisu vzhľadom na počty vlákien

Vyhľadovanie

Init():
readLS \leftarrow LS()
roomE \leftarrow Sem(1)

R():

- 1) readLS.lock(roomE)
- 2) // stale je tu nejaky cit.
- 3) readLS.unlock(roomE)

W():

- 1) roomE.wait()
- 2) // kod zapis.
- 3) roomE.signal()

Vyhladovenie (Starvation)

- Riešenie turniketom
- Napríklad R-W
 - ▣ Čitatelia chodia cez turniket ako na kolotoči
 - ▣ Zapisovateľ má možnosť zablokovať turniket, takže už neprejdú cez turniket ďalší čitatelia

Vyhľadovanie

```
Init():  
  readLS ← LS()  
  roomE ← Sem(1)  
  turn ← Sem(1)
```


Vyhľadovanie

Init():
readLS ← LS()
roomE ← Sem(1)
turn ← Sem(1)

R():

- 1)
- 2)
- 3) readLS.lock(roomE)
- 4) // kod citatelov
- 5) readLS.unlock(roomE)

W():

- 1)
- 2) roomE.wait()
- 3) // kod zapis.
- 4) roomE.signal()
- 5)

Vyhľadovanie

```
Init():  
readLS ← LS()  
roomE ← Sem(1)  
turn ← Sem(1)
```

R():

- 1) `turn.wait()`
- 2) `turn.signal()`
- 3) `readLS.lock(roomE)`
- 4) `// kod citatelov`
- 5) `readLS.unlock(roomE)`

W():

- 1)
- 2) `roomE.wait()`
- 3) `// kod zapis.`
- 4) `roomE.signal()`
- 5)

Vyhľadovanie

```
Init():  
  readLS ← LS()  
  roomE ← Sem(1)  
  turn ← Sem(1)
```

R():

- 1) `turn.wait()`
- 2) `turn.signal()`
- 3) `readLS.lock(roomE)`
- 4) `// kod citatelov`
- 5) `readLS.unlock(roomE)`

W():

- 1) `turn.wait()`
- 2) `roomE.wait()`
- 3) `// kod zapis.`
- 4) `roomE.signal()`
- 5) `turn.signal()`

Vyhľadovanie

```
Init():  
readLS ← LS()  
roomE ← Sem(1)  
turn ← Sem(1)
```

R():

- 1) `turn.wait()`
- 2) `turn.signal()`
- 3) `readLS.lock(roomE)`
- 4) `// kod citatelov`
- 5) `readLS.unlock(roomE)`

W():

- 1) `turn.wait()`
- 2) `roomE.wait()`
- 3) `// kod zapis.`
- 4) `roomE.signal()`
- 5) `turn.signal()`

Vyhľadovanie

```
Init():  
  readLS ← LS()  
  roomE ← Sem(1)  
  turn ← Sem(1)
```

W():

- 1) `turn.wait()`
- 2) `roomE.wait()`
- 3) `// kod zapis.`
- 4) `roomE.signal()`
- 5) `turn.signal()`

Vyhľadovanie

Init():
readLS \leftarrow LS()
roomE \leftarrow Sem(1)
turn \leftarrow Sem(1)

W():

- 1) turn.wait()
- 2) roomE.wait()
- 3) // kod zapis.
- 4)
- 5)

W():

- 1) turn.wait()
- 2) roomE.wait()
- 3) // kod zapis.
- 4) roomE.signal()
- 5) turn.signal()

Vyhľadovanie

Init():
readLS \leftarrow LS()
roomE \leftarrow Sem(1)
turn \leftarrow Sem(1)

W():

- 1) turn.wait()
- 2) roomE.wait()
- 3) // kod zapis.
- 4) turn.signal()
- 5) roomE.signal()

W():

- 1) turn.wait()
- 2) roomE.wait()
- 3) // kod zapis.
- 4) roomE.signal()
- 5) turn.signal()

Vyhľadovanie

```
Init():  
  readLS ← LS()  
  roomE ← Sem(1)  
  turn ← Sem(1)
```

W():

- 1) `turn.wait()`
- 2) `roomE.wait()`
- 3) // kod zapis.
- 4) `turn.signal()`
- 5) `roomE.signal()`



W():

- 1) `turn.wait()`
- 2) `roomE.wait()`
- 3) // kod zapis.
- 4) `roomE.signal()`
- 5) `turn.signal()`

Vyhľadovanie

- Na cvičení implementujte obe verzie riešenia problému vyhľadovania zapisovateľov
- Experimentujte s nastaveniami systému
- Je rozdiel medzi oboma riešeniami? Aký? Ktoré je lepšie? Za akých podmienok?