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

- Fronta pomocou semaforov
- Synchronizacia vstupu tanecnych parov
- Producent-Konzument
- Vzájomne vylucenie kategorii
- Citatelia-Zapisovatelia
- Vypinac (Lightswitch)

Fronta (Queue)

- Semafor jednoducho implementuje frontu
- Inicializacia na 0
- Ked je fronta prazdna, wait() automaticky blokuje

Tanecne pary

- Vlakna su 2 druhy tanecnikov
  - Muz
  - Zena
- Pred vstupom do tanecnej miestnosti stoja kazdy vo svojom rade (rad muzov a rad zien)
- Kazdy po prichode pozrie do druhej rady; ak ma partnera, mozu vstupit tancovat, inak caka

Tanecne pary

Init():

muzi ← sempahore(0)

zeny ← semaphore(0)

Tanecne pary

Init():
muzi ← sempahore(0)
zeny ← semaphore(0)

Muz():

1) muzi.signal()
2) zeny.wait()
3) tancuj()

Zena():

1) zeny.signal()
2) muzi.wait()
3) tancuj()

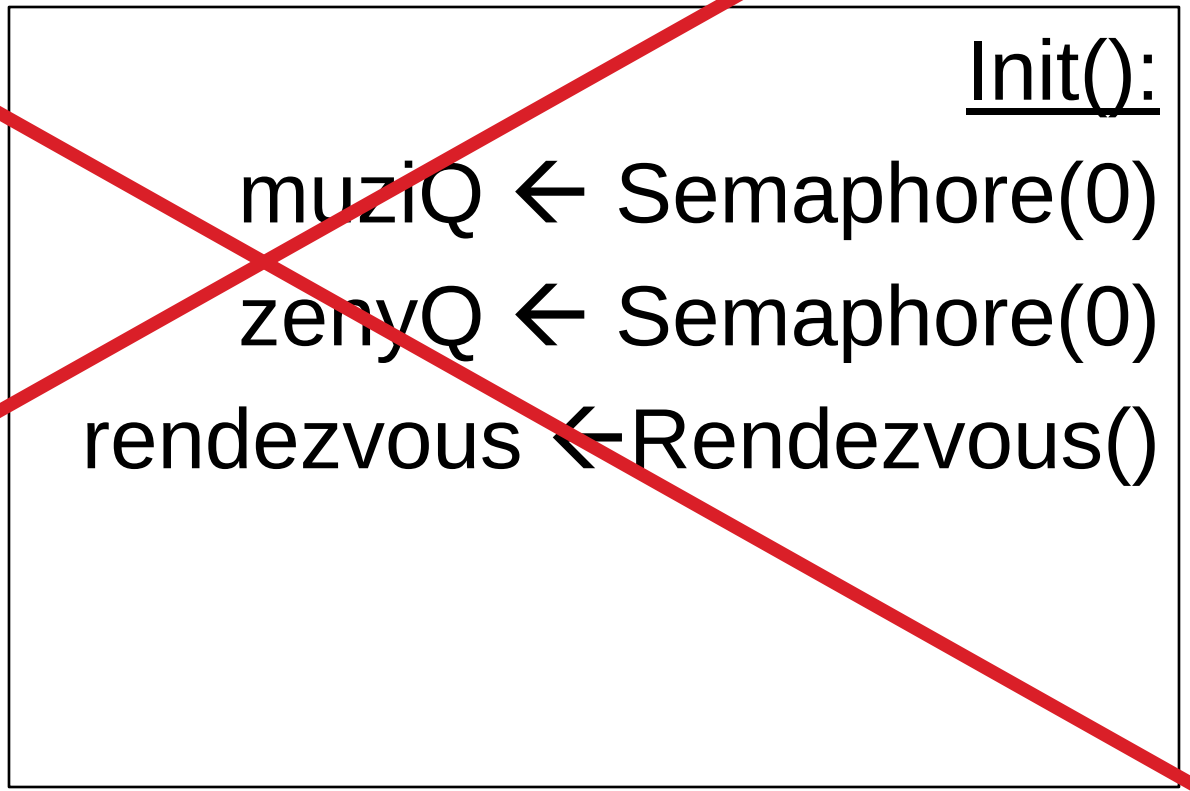
Tanecne pary

- Toto riesenie implementuje aky synchronizacny vzor?

- Garantuje, ze mozu tancovat spolocne
- Negarantuje, ze budu tancovat spolocne!!!
- Negarantuje dvojici, ze bude jedina na parkete…

- Zabezpecme volny parket pre dvojicu

Tanecne pary



Tanecne pary

Init():
muziQ ← Semaphore(0)
zenyQ ← Semaphore(0)
rendezvous ← Rendezvous()

Zena():

1) muziQ.signal()
2) zenyQ.wait()
3) tancuj()
4) rendez.wait()

Muz():

1) zenyQ.signal()
2) muziQ.wait()
3) tancuj()
4) rendez.wait()

Tanecne pary

Init():
muziQ ← Semaphore(0)
zenyQ ← Semaphore(0)
rendezvous ← Rendezvous()

Zena():

1) muziQ.signal()
2) zenyQ.wait()
3) tancuj()
4) rendez.wait()

Muz():

1) zenyQ.signal()
2) muziQ.wait()
3) tancuj()
4) rendez.wait()

Co garantuje ‘rendez’?

Tanecne pary

Init():
muziQ ← Semaphore(0)
zenyQ ← Semaphore(0)
rendezvous ← Rendezvous()

Zena():

1) muziQ.signal()
2) zenyQ.wait()
3) tancuj()
4) rendez.wait()

Muz():

1) zenyQ.signal()
2) muziQ.wait()
3) tancuj()
4) rendez.wait()

Ako zabezpecit parket pre dvojicu?

Tanecne pary

Init():
muziQ ← Semaphore(0)
zenyQ ← Semaphore(0)
rendezvous ← Rendezvous()

Zena():

1) muziQ.signal()
2) zenyQ.wait()
3) tancuj()
4) rendez.wait()

Muz():

1) zenyQ.signal()
2) muziQ.wait()
3) tancuj()
4) rendez.wait()

Pridajme mutex, ktory bude riadit jeden z dvojice

Tanecne pary

Init():
muziQ ← Semaphore(0)
zenyQ ← Semaphore(0)
rendezvous ←Rendezvous()
mutex ← Mutex()

Zena():
1)
2) muziQ.signal()
3) zenyQ.wait()
4) tancuj()
5) rendez.wait()
6)

Muz():
1) mutex.lock()
2) zenyQ.signal()
3) muziQ.wait()
4) tancuj()
5) rendez.wait()
6) mutex.unlock()

Tanecne pary

Init():
muziQ ← Semaphore(0)
zenyQ ← Semaphore(0)
rendezvous ←Rendezvous()
mutex ← Mutex()

Zena():
1)
2) muziQ.signal()
3) zenyQ.wait()
4) tancuj()
5) rendez.wait()
6)

Muz():
1) mutex.lock()
2) zenyQ.signal()
3) muziQ.wait()
4) tancuj()
5) rendez.wait()
6) mutex.unlock()

Prichadzajuci muzi sa nemozu radit do fronty!

Tanecne pary

Init():
muziQ ← Semaphore(0)
zenyQ ← Semaphore(0)
rendezvous ←Rendezvous()
mutex ← Mutex()

Zena():
1)
2) muziQ.signal()
3) zenyQ.wait()
4) tancuj()
5) rendez.wait()
6)

Muz():
1) mutex.lock()
2) zenyQ.signal()
3) muziQ.wait()
4) tancuj()
5) rendez.wait()
6) mutex.unlock()

Prichadzajuci muzi sa nemozu radit do fronty!

Tanecne pary

```
Init():
    muziQ ← Semaphore(0)
    zenyQ ← Semaphore(0)
    rendezvous ←Rendezvous()
    mutex ← Mutex()
```

Pridajme pocitadlo pre fronty

Tanecne pary

Init():

muziQ ← Semaphore(0)

zenyQ ← Semaphore(0)

rendezvous ←Rendezvous()

mutex ← Mutex()

muziCnt ← 0

zenyCnt ← 0

Pridajme pocitadlo pre fronty

Tanecne pary

Init():
muziQ ← Semaphore(0), zenyQ ← Semaphore(0)
rendezvous ← Rendezvous()
mutex ← Mutex()
muziCnt ← 0, zenyCnt ← 0

Zena():
1)
2) muziQ.signal()
3) zenyQ.wait()
4) tancuj()
5) rendez.wait()
6)

Muz():
1) mutex.lock()
2) zenyQ.signal()
3) muziQ.wait()
4) tancuj()
5) rendez.wait()
6) mutex.unlock()

Tanecne pary

Init():
muziQ ← Semaphore(0), zenyQ ← Semaphore(0)
rendezvous ← Rendezvous()
mutex ← Mutex()
muziCnt ← 0, zenyCnt ← 0

Zena():
1)
2) muziQ.signal()
3) zenyQ.wait()
4) tancuj()
5) rendez.wait()
6)

Muz():
1) mutex.lock()
2) zenyQ.signal()
3) muziQ.wait()
4) tancuj()
5) rendez.wait()
6) mutex.unlock()

Muz po prichode skontroluje pritomnost zien

Tanecne pary

Init():
muziQ ← Semaphore(0), zenyQ ← Semaphore(0)
rendezvous ← Rendezvous()
mutex ← Mutex()
muziCnt ← 0, zenyCnt ← 0

Zena():
1)
2) muziQ.signal()
3) zenyQ.wait()
4) tancuj()
5) rendez.wait()
6)

Muz():
1) mutex.lock()
2) zenyQ.signal()
3) muziQ.wait()
4) tancuj()
5) rendez.wait()
6) mutex.unlock()

Ak tam ziadna nebude, uvolni mutex a caka

Tanecne pary

Init():
muziQ ← Semaphore(0), zenyQ ← Semaphore(0)
rendezvous ← Rendezvous()
mutex ← Mutex()
muziCnt ← 0, zenyCnt ← 0

Zena():
1)
2) muziQ.signal()
3) zenyQ.wait()
4) tancuj()
5) rendez.wait()
6)

Muz():
1) mutex.lock()
2)
3)
4)
5)
6)
7) tancuj()
8) rendez.wait()
9) mutex.unlock()

Tanecne pary

Init():
muziQ ← Semaphore(0), zenyQ ← Semaphore(0)
rendezvous ← Rendezvous()
mutex ← Mutex()
muziCnt ← 0, zenyCnt ← 0

```
Zena():
1)
2)    muziQ.signal()
3)    zenyQ.wait()
4)    tancuj()
5)    rendez.wait()
6)
```

```
Muz():
1)    mutex.lock()
2)    if zenyCnt <= 0:
3)
4)
5)
6)
7)    tancuj()
8)    rendez.wait()
9)    mutex.unlock()
```

Tanecne pary

Init():
muziQ ← Semaphore(0), zenyQ ← Semaphore(0)
rendezvous ← Rendezvous()
mutex ← Mutex()
muziCnt ← 0, zenyCnt ← 0

```
Zena():
1)
2)    muziQ.signal()
3)    zenyQ.wait()
4)    tancuj()
5)    rendez.wait()
6)
```

```
Muz():
1)    mutex.lock()
2)    if zenyCnt <= 0:
3)        muziCnt += 1
4)
5)
6)
7)    tancuj()
8)    rendez.wait()
9)    mutex.unlock()
```

Tanecne pary

Init():
muziQ ← Semaphore(0), zenyQ ← Semaphore(0)
rendezvous ← Rendezvous()
mutex ← Mutex()
muziCnt ← 0, zenyCnt ← 0

```
Zena():
1)
2)    muziQ.signal()
3)    zenyQ.wait()
4)    tancuj()
5)    rendez.wait()
6)
```

```
Muz():
1)    mutex.lock()
2)    if zenyCnt <= 0:
3)        muziCnt += 1
4)        mutex.unlock()
5)
6)
7)    tancuj()
8)    rendez.wait()
9)    mutex.unlock()
```

Tanecne pary

Init():
muziQ ← Semaphore(0), zenyQ ← Semaphore(0)
rendezvous ← Rendezvous()
mutex ← Mutex()
muziCnt ← 0, zenyCnt ← 0

```
Zena():
1)
2)    muziQ.signal()
3)    zenyQ.wait()
4)    tancuj()
5)    rendez.wait()
6)
```

```
Muz():
1)    mutex.lock()
2)    if zenyCnt <= 0:
3)        muziCnt += 1
4)        mutex.unlock()
5)        muziQ.wait()
6)
7)    tancuj()
8)    rendez.wait()
9)    mutex.unlock()
```

Tanecne pary

Init():
muziQ ← Semaphore(0), zenyQ ← Semaphore(0)
rendezvous ← Rendezvous()
mutex ← Mutex()
muziCnt ← 0, zenyCnt ← 0

```
Zena():
1)
2)    muziQ.signal()
3)    zenyQ.wait()
4)    tancuj()
5)    rendez.wait()
6)
```

```
Muz():
1)    mutex.lock()
2)    if zenyCnt <= 0:
3)        muziCnt += 1
4)        mutex.unlock()
5)        muziQ.wait()
6)    zenyQ.signal()
7)    tancuj()
8)    rendez.wait()
9)    mutex.unlock()
```

Tanecne pary

Init():
muziQ ← Semaphore(0), zenyQ ← Semaphore(0)
rendezvous ← Rendezvous()
mutex ← Mutex()
muziCnt ← 0, zenyCnt ← 0

```
Zena():
1)
2)    muziQ.signal()
3)    zenyQ.wait()
4)    tancuj()
5)    rendez.wait()
6)
```

```
Muz():
1)    mutex.lock()
2)    if zenyCnt <= 0:
3)        muziCnt += 1
4)        mutex.unlock()
5)        muziQ.wait()
6)    zenyQ.signal()
7)    tancuj()
8)    rendez.wait()
9)    mutex.unlock()
```

Zvysujeme pocty muzov… ale co zeny?

Tanecne pary

Init():
muziQ ← Semaphore(0), zenyQ ← Semaphore(0)
rendezvous ← Rendezvous()
mutex ← Mutex()
muziCnt ← 0, zenyCnt ← 0

```
Zena():
1)
2)    muziQ.signal()
3)    zenyQ.wait()
4)    tancuj()
5)    rendez.wait()
6)
```

```
Muz():
1)    mutex.lock()
2)    if zenyCnt <= 0:
3)        muziCnt += 1
4)        mutex.unlock()
5)        muziQ.wait()
6)    zenyQ.signal()
7)    tancuj()
8)    rendez.wait()
9)    mutex.unlock()
```

Mali by sme zaviest symetriu riesenia

Tanecne pary

Init():
muziQ ← Semaphore(0), zenyQ ← Semaphore(0)
rendezvous ← Rendezvous()
mutex ← Mutex()
muziCnt ← 0, zenyCnt ← 0

Zena():
```
1)      mutex.lock()
2)
3)
4)
5)
6)
7)      tancuj()
8)      rendez.wait()
9)
```

Muz():
```
1)      mutex.lock()
2)      if zenyCnt <= 0:
3)          muziCnt += 1
4)          mutex.unlock()
5)          muziQ.wait()
6)      zenyQ.signal()
7)      tancuj()
8)      rendez.wait()
9)      mutex.unlock()
```

Tanecne pary

Init():
muziQ ← Semaphore(0), zenyQ ← Semaphore(0)
rendezvous ← Rendezvous()
mutex ← Mutex()
muziCnt ← 0, zenyCnt ← 0

```
Zena():
1)      mutex.lock()
2)      if muziCnt <= 0:
3)
4)
5)
6)
7)      tancuj()
8)      rendez.wait()
9)
```

```
Muz():
1)      mutex.lock()
2)      if zenyCnt <= 0:
3)          muziCnt += 1
4)          mutex.unlock()
5)          muziQ.wait()
6)      zenyQ.signal()
7)      tancuj()
8)      rendez.wait()
9)      mutex.unlock()
```

Tanecne pary

Init():
muziQ ← Semaphore(0), zenyQ ← Semaphore(0)
rendezvous ← Rendezvous()
mutex ← Mutex()
muziCnt ← 0, zenyCnt ← 0

```
Zena():
1)      mutex.lock()
2)      if muziCnt <= 0:
3)          zenyCnt += 1
4)
5)
6)
7)      tancuj()
8)      rendez.wait()
9)
```

```
Muz():
1)      mutex.lock()
2)      if zenyCnt <= 0:
3)          muziCnt += 1
4)          mutex.unlock()
5)          muziQ.wait()
6)      zenyQ.signal()
7)      tancuj()
8)      rendez.wait()
9)      mutex.unlock()
```

Tanecne pary

Init():
muziQ ← Semaphore(0), zenyQ ← Semaphore(0)
rendezvous ← Rendezvous()
mutex ← Mutex()
muziCnt ← 0, zenyCnt ← 0

```
Zena():
1)      mutex.lock()
2)      if muziCnt <= 0:
3)          zenyCnt += 1
4)          mutex.unlock()
5)
6)
7)      tancuj()
8)      rendez.wait()
9)
```

```
Muz():
1)      mutex.lock()
2)      if zenyCnt <= 0:
3)          muziCnt += 1
4)          mutex.unlock()
5)          muziQ.wait()
6)      zenyQ.signal()
7)      tancuj()
8)      rendez.wait()
9)      mutex.unlock()
```

Tanecne pary

Init():
muziQ ← Semaphore(0), zenyQ ← Semaphore(0)
rendezvous ← Rendezvous()
mutex ← Mutex()
muziCnt ← 0, zenyCnt ← 0

```
Zena():
1)      mutex.lock()
2)      if muziCnt <= 0:
3)          zenyCnt += 1
4)          mutex.unlock()
5)          zenyQ.wait()
6)
7)      tancuj()
8)      rendez.wait()
9)
```

```
Muz():
1)      mutex.lock()
2)      if zenyCnt <= 0:
3)          muziCnt += 1
4)          mutex.unlock()
5)          muziQ.wait()
6)      zenyQ.signal()
7)      tancuj()
8)      rendez.wait()
9)      mutex.unlock()
```

Tanecne pary

Init():
muziQ ← Semaphore(0), zenyQ ← Semaphore(0)
rendezvous ← Rendezvous()
mutex ← Mutex()
muziCnt ← 0, zenyCnt ← 0

```
Zena():
1)      mutex.lock()
2)      if muziCnt <= 0:
3)          zenyCnt += 1
4)          mutex.unlock()
5)          zenyQ.wait()
6)      muziQ.signal()
7)      tancuj()
8)      rendez.wait()
9)
```

```
Muz():
1)      mutex.lock()
2)      if zenyCnt <= 0:
3)          muziCnt += 1
4)          mutex.unlock()
5)          muziQ.wait()
6)      zenyQ.signal()
7)      tancuj()
8)      rendez.wait()
9)      mutex.unlock()
```

Tanecne pary

Init():
muziQ ← Semaphore(0), zenyQ ← Semaphore(0)
rendezvous ← Rendezvous()
mutex ← Mutex()
muziCnt ← 0, zenyCnt ← 0

```
Zena():
1)      mutex.lock()
2)      if muziCnt <= 0:
3)          zenyCnt += 1
4)          mutex.unlock()
5)          zenyQ.wait()
6)      muziQ.signal()
7)      tancuj()
8)      rendez.wait()
9)
```

```
Muz():
1)      mutex.lock()
2)      if zenyCnt <= 0:
3)          muziCnt += 1
4)          mutex.unlock()
5)          muziQ.wait()
6)      zenyQ.signal()
7)      tancuj()
8)      rendez.wait()
9)      mutex.unlock()
```

Nikde neznizujeme pocty vo frontach!!!

Tanecne pary

Init():
muziQ ← Semaphore(0), zenyQ ← Semaphore(0)
rendezvous ← Rendezvous()
mutex ← Mutex()
muziCnt ← 0, zenyCnt ← 0

```
Zena():
1)      mutex.lock()
2)      if muziCnt <= 0:
3)          zenyCnt += 1
4)          mutex.unlock()
5)          zenyQ.wait()
6)
7)
8)      muziQ.signal()
9)      tancuj()
10)     rendez.wait()
11)
```

```
Muz():
1)      mutex.lock()
2)      if zenyCnt <= 0:
3)          muziCnt += 1
4)          mutex.unlock()
5)          muziQ.wait()
6)
7)
8)      zenyQ.signal()
9)      tancuj()
10)     rendez.wait()
11)     mutex.unlock()
```

Tanecne pary

Init():
muziQ ← Semaphore(0), zenyQ ← Semaphore(0)
rendezvous ← Rendezvous()
mutex ← Mutex()
muziCnt ← 0, zenyCnt ← 0

```
Zena():
1)      mutex.lock()
2)      if muziCnt <= 0:
3)          zenyCnt += 1
4)          mutex.unlock()
5)          zenyQ.wait()
6)      else:
7)
8)      muziQ.signal()
9)      tancuj()
10)     rendez.wait()
11)
```

```
Muz():
1)      mutex.lock()
2)      if zenyCnt <= 0:
3)          muziCnt += 1
4)          mutex.unlock()
5)          muziQ.wait()
6)      else:
7)
8)      zenyQ.signal()
9)      tancuj()
10)     rendez.wait()
11)     mutex.unlock()
```

Tanecne pary

Init():
muziQ ← Semaphore(0), zenyQ ← Semaphore(0)
rendezvous ← Rendezvous()
mutex ← Mutex()
muziCnt ← 0, zenyCnt ← 0

```
Zena():
1)      mutex.lock()
2)      if muziCnt <= 0:
3)          zenyCnt += 1
4)          mutex.unlock()
5)          zenyQ.wait()
6)      else:
7)          muziCnt -= 1
8)      muziQ.signal()
9)      tancuj()
10)     rendez.wait()
11)
```

```
Muz():
1)      mutex.lock()
2)      if zenyCnt <= 0:
3)          muziCnt += 1
4)          mutex.unlock()
5)          muziQ.wait()
6)      else:
7)          zenyCnt -= 1
8)      zenyQ.signal()
9)      tancuj()
10)     rendez.wait()
11)     mutex.unlock()
```

Tanecne pary

Init():
muziQ ← Semaphore(0), zenyQ ← Semaphore(0)
rendezvous ← Rendezvous()
mutex ← Mutex()
muziCnt ← 0, zenyCnt ← 0

```
Zena():
1)      mutex.lock()
2)      if muziCnt <= 0:
3)          zenyCnt += 1
4)          mutex.unlock()
5)          zenyQ.wait()
6)      else:
7)          muziCnt -= 1
8)          muziQ.signal()
9)      tancuj()
10)     rendez.wait()
11)
```

```
Muz():
1)      mutex.lock()
2)      if zenyCnt <= 0:
3)          muziCnt += 1
4)          mutex.unlock()
5)          muziQ.wait()
6)      else:
7)          zenyCnt -= 1
8)          zenyQ.signal()
9)      tancuj()
10)     rendez.wait()
11)     mutex.unlock()
```

Tanecne pary

Init():
muziQ ← Semaphore(0), zenyQ ← Semaphore(0)
rendezvous ← Rendezvous()
mutex ← Mutex()
muziCnt ← 0, zenyCnt ← 0

```
Zena():
1)      mutex.lock()
2)      if muziCnt <= 0:
3)          zenyCnt += 1
4)          mutex.unlock()
5)          zenyQ.wait()
6)      else:
7)          muziCnt -= 1
8)          muziQ.signal()
9)      tancuj()
10)     rendez.wait()
11)
```

```
Muz():
1)      mutex.lock()
2)      if zenyCnt <= 0:
3)          muziCnt += 1
4)          mutex.unlock()
5)          muziQ.wait()
6)      else:
7)          zenyCnt -= 1
8)          zenyQ.signal()
9)      tancuj()
10)     rendez.wait()
11)     mutex.unlock()
```

Tanecne pary

Init():
muziQ ← Semaphore(0), zenyQ ← Semaphore(0)
rendezvous ← Rendezvous()
mutex ← Mutex()
muziCnt ← 0, zenyCnt ← 0

```
Zena():
1)      mutex.lock()
2)      if muziCnt <= 0:
3)          zenyCnt += 1
4)          mutex.unlock()
5)          zenyQ.wait()
6)      else:
7)          muziCnt -= 1
8)          muziQ.signal()
9)      tancuj()
10)     rendez.wait()
11)
```

```
Muz():
1)      mutex.lock()
2)      if zenyCnt <= 0:
3)          muziCnt += 1
4)          mutex.unlock()
5)          muziQ.wait()
6)      else:
7)          zenyCnt -= 1
8)          zenyQ.signal()
9)      tancuj()
10)     rendez.wait()
11)     mutex.unlock()
```

Tanecne pary

Init():
muziQ ← Semaphore(0), zenyQ ← Semaphore(0)
rendezvous ← Rendezvous()
mutex ← Mutex()
muziCnt ← 0, zenyCnt ← 0

```
Zena():
1)      mutex.lock()
2)      if muziCnt <= 0:
3)          zenyCnt += 1
4)          mutex.unlock()
5)          zenyQ.wait()
6)      else:
7)          muziCnt -= 1
8)          muziQ.signal()
9)      tancuj()
10)     rendez.wait()
11)
```

```
Muz():
1)      mutex.lock()
2)      if zenyCnt <= 0:
3)          muziCnt += 1
4)          mutex.unlock()
5)          muziQ.wait()
6)      else:
7)          zenyCnt -= 1
8)          zenyQ.signal()
9)      tancuj()
10)     rendez.wait()
11)     mutex.unlock()
```

Producent - konzument

- Velmi casta schema v aplikacnom programovani
  - Obsluha spojeni
  - Udalostami riadeny system
- Zdielany objekt: sklad (buffer)

Producent - konzument

- Konzistencia skladu (vkladanie/vyberanie poloziek)

- Ak je sklad prazdny, konzument musi cakat
- Ak je sklad plny, producent musi cakat

P-K

P():

1) item = produceItem()

2) buffer.add(item)

K():

1) item = buffer.get()

2) item.process()

P-K

Init():
mutex ← Mutex()
items ← Semaphore(0)

P():

1) item = produceItem()

2) buffer.add(item)

K():

1) item = buffer.get()

2) item.process()

P-K

Init():
mutex ← Mutex()
items ← Semaphore(0)

P():	K():
1) item = produceItem()	1) items.wait()
2) buffer.add(item)	2) item = buffer.get()
	3) item.process()

P-K

Init():
mutex ← Mutex()
items ← Semaphore(0)

P():	K():
1) item = produceItem()	1) items.wait()
2) buffer.add(item)	2) item = buffer.get()
3) items.signal()	3) item.process()

P-K

Init():
mutex ← Mutex()
items ← Semaphore(0)

P():	K():
1) item = produceItem()	1) items.wait()
2) mutex.lock()	2) mutex.lock()
3) buffer.add(item)	3) item = buffer.get()
4) items.signal()	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 moznosti, kam do P() vlozit free.wait():
  - Pred R1
  - Medzi R1-R2
  - Medzi R2-R3
- Po R3 uz nema zmysel riesit

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()

- Kolko moznosti je spravnych?
- Kolko je optimalnych?
- Ktore to su a preco?

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()

- Koliko moznosti je spravnych?
- Kolko je optimalnych?
- Ktore to su a preco?

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()

Vylucenie kategorii procesov

- Napriklad problem spolocneho WC
  - WC s 3 kabinkami
  - Bud ho pouzivaju muzi alebo zeny, ale nie sucasne
- Problem citatelov a zapisovatelov
  - Menit udaje (zapisovat) moze iba 1, vtedy sa nesmie citat
  - Citat mozu viaceri, ale sa nemoze zapisovat

Vylucenie kategorii procesov

- Semafor
  - Signalizuje obsadenost oblasti (miestnosti)
  - Napr. ak je WC volne, vojde zena a obsadi tak miestnost, muzi nemozu vojst; a opacne
- Ako zabezpecit, aby iba jeden proces signalizoval obsadenie miestnosti?

Vylucenie kategorii procesov

- Signalizacia obsadenosti
  - Kto prvy vchadza, signalizuje obsadenost
  - Kto posledny vychadza, signalizuje volnost
  - Napr. cedulka na WC
- Ako vieme, kto je prvy a kto posledny??

Vylucenie kategorii procesov

- Ako zistime, kto je prvy a kto je posledny?
  - Pocitadlo!
  - Ale ked mame pocitadlo, a k nemu pristup viacerych vlakien…
- Staci nam pocitat pritomnych v miestnosti, alebo potrebujeme vediet odlisit kategoriu?

Problem citatelia-zapisovatelia

Init():

mutex ← Mutex()

readers ← 0

roomEmpty ← Semaphore(1)

R-W

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

R-W

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

R():

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

W():

1)
2)
3)
4)
5)
6)
7)
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)
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():
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)     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 riesenia

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 riesenia
- Kto prvy vchadza, rozsvieti
- Kto posledny odchadza, zhasne
- Vypinac (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()
```

- Vlastnosti riesenia
- Kto prvy vchadza, rozsvieti
- Kto posledny odchadza, zhasne
- Vypinac (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()
```

- Vlastnosti riesenia
  - Iba jeden citatel caka na semafore roomEmpty
  - Viaceri zapisovatelia mozu cakat na semafore roomEmpty
  - Miestnost je prazdna, ked citatel vyvolava 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 riesenia
  - Iba jeden citatel caka na semafore roomEmpty
  - Viaceri zapisovatelia mozu cakat na semafore roomEmpty
  - Miestnost je prazdna, ked citatel vyvolava roomEmpty.signal()

ADT Lightswitch

- Interny stav
  - mutex
  - pocitadlo
- Metody
  - lock(sem)
  - unlock(sem)

ADT Lightswitch

- Interny stav
  - mutex
  - pocitadlo
- Metody
  - lock(sem)
  - unlock(sem)

Init():

mutex ← Mutex()

counter ← 0

ADT Lightswitch

Init():

mutex ← Mutex()

counter ← 0

ADT Lightswitch

Init():
mutex ← Mutex()
counter ← 0

lock(sem):

unlock(sem):

ADT Lightswitch

Init():
mutex ← Mutex()
counter ← 0

lock(sem):
1)
2)
3)
4)
5)

unlock(sem):
1)
2)
3)
4)
5)

ADT Lightswitch

Init():
mutex ← Mutex()
counter ← 0

lock(sem):
1)
2)
3)
4) sem.wait()
5)

unlock(sem):
1)
2)
3)
4) sem.signal()
5)

ADT Lightswitch

Init():
mutex ← Mutex()
counter ← 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 ← Mutex()
counter ← 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 ← Mutex()
counter ← 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()
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 pomocout LS

Init():

readLS ← LS()

roomE ←Sem(1)

R-W pomocout LS

Init():
readLS ← LS()
roomE ←Sem(1)

R():

1)

2)

3)

W():

1)

2)

3)

R-W pomocout LS

Init():
readLS ← LS()
roomE ←Sem(1)

R():

1)

2) // kod citat.

3)

W():

1)

2) // kod zapis.

3)

R-W pomocout 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()

Vyhladovenie (Starvation)

Vyhladovenie (Starvation)

- Nie je to uviaznutie (deadlock)

- Problem, ked sa nejaky proces nikdy “nedostane k slovu”
  - Napr. WC s 3 misami v krcme
  - 2 zeny, kazda pije 1 kavu za 2 hodinkY
  - 37 chlapov, kazdy vypije 5-13 piv za 10 min ;)

Vyhladovenie (Starvation)

- Velmi casto nastava pri rieseni synchronizacnych problemov, kde sa vyuziva LS
- Napriklad aj v R-W moze byt problem, v zavislosti od konfiguracie modelu
  - Radovy nepomer vlakien
  - Rozna doba citania a zapisu vzhladom na pocty vlakien

Vyhladovenie

Init():
readLS ← LS()
roomE ←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)

- Riesenie turniketom
- Napriklad R-W
  - Citatelia chodia cez turniket ako na kolotoci
  - Zapisovatel ma moznost zablokovat turniket, takze uz neprejdu cez turniket dalsi citatelia

Vyhladovenie

Init():

readLS ← LS()

roomE ←Sem(1)

turn ← Sem(1)

Vyhladovenie

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)
```

Vyhladovenie

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)

Vyhladovenie

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()

Vyhladovenie

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()

Vyhladovenie

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()

Vyhladovenie

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

Vyhladovenie

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()

Vyhladovenie

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()

Vyhladovenie

- Na cvičení implementujte obe verzie riešenia problému vyhladovenia zapisovateľov
- Experimentujte s nastaveniami systému

- Je rozdiel medzi oboma riešeniami? Aký? Ktoré je lepšie? Za akych podmienok?